Universidad de las Ciencias Informáticas

FACULTAD 6

Título: "Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Sureny Orihuela Sánchez

Liandry Monteslier López

Tutores:

Ing. Miguel Lezcano Ramos

Ing. Julio César Brito Rodríguez

La Habana, Junio del 2013

"Año 55 de la Revolución"

"Los sacrificios que se hagan no deben mostrarse como tarjeta de identificación, sino como el cumplimiento del deber."

Ernesto Guevara de la Serna

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Sureny Orihuela Sánchez

_____

Firma del Autor

Ing. Julio César Brito Rodríguez

_____

Firma del Tutor

Liandry Monteslier López

_____

Firma del Autor

Ing. Miguel Lezcano Ramos

_____

Firma del Tutor

DATOS DE CONTACTO

Autores:

Sureny Orihuela Sánchez
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: sorihuela@estudiantes.uci.cu

Liandry Monteslier López
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: lmonteslierl@estudiantes.uci.cu

Tutores:
Ing. Miguel Lezcano Ramos
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: mlezcano@uci.cu

Ing. Julio César Brito Rodríguez
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: jcbrito@uci.cu

AGRADECIMIENTOS

Hoy celebro el fin de una etapa importante en mi vida donde me toca despedirme y agradecerles a muchas personas que a lo largo de estos años, han contribuido a que hoy yo esté cumpliendo mi sueño, personas que han llegado para quedarse siempre en mis recuerdos.

A mi compañero de tesis Liandry, por ayudarme a hacer realidad mi sueño, aguantarme todas mis malcriadeces y peleas, por estar siempre presente cuando más lo necesitaba y porque "sin ti no hubiera sido posible esto".

A mis padres por ser las personas más especiales en mi vida y guiarme por el camino correcto, por darme todo su amor y cariño, por apoyarme en todas mis decisiones y acompañarme en los momentos buenos y malos que he tenido. Gracias por ser más que padres y confiar en mí, como verdaderos amigos, haciéndome sonreír antes los problemas.

A mi familia que son las personas más importantes para mí, mi padrastro, mis tíos, mis primos, en fin toda la familia que se han sacrificado siempre por verme echa toda una mujer. En especial a mis abuelitos del alma Aldo, Mercedes y Eva, mis viejitos preferidos que sin ellos nada de esto habría pasado, gracias por enseñarme a ser fuerte en los momentos más duros de la vida y tener mucha fe ante las dificultades.

A una personita más que especial para mí, mi novio Ariam. Gracias por tener paciencia y comprenderme todo este tiempo, sé que no fue fácil pero lo hiciste muy bien. Para ti mil gracias por amarme tanto y por ser la persona más especial que he conocido, TE AMO.

A mi suegra, mi cuñada y mi abuelita Blanquita, gracias por quererme como una más de la familia y por darme tanto apoyo incondicional. Marietta gracias por ser la hermana que nunca tuve y por ser tan cariñosa conmigo.

A mis tutores Julio César Brito y Miguel Lezcano por apoyarme, guiarme y soportar mis malcriadeces. Gracias por su grandísimo esfuerzo, por toda la preocupación y por todos los regaños cuando fueron necesarios. Gracias por ser tan admirables para mí.

A mi tribunal y a mi oponente por estar dispuestos siempre a ayudarme en todo lo que necesité y por los buenos consejos me dieron siempre.

A mis amigos de estos 5 años los que están hoy aquí y los que no, pues se convirtieron en mi familia de la UCI, gracias por todos los buenos momentos que pasamos juntos y los recuerdos que quedaron grabados para siempre. En especial amigos como Yuned, Yailema, Gleivis, Daneilis, Siana, Ana del Carmen, Yudit, Yanet, Popi, Migue, Osman, Dasley, Poti, Sandy, Darlon, Julio, Jorge Antonio, Yosbel, Jorge Ernesto, Reisel y muchos más, tantos, que no cabrían en otras 80 páginas.

A mis amigas del apartamento Liena, Rosita, Ailin, Yari y Yasmany gracias por todas las cosas que compartimos siempre, por todas las peleas y por todo el cariño que me demostraron.

AGRADECIMIENTOS

A profesores que nunca olvidaré como Haymee, Arodys, Lacoste, Oscar, Yudelkys, entre otros que me ayudaron mucho.

A mis amistades del comedor de la dieta que me acompañaron todo este tiempo y me hicieron vivir momentos de alegrías, gracias a Pedro, Alfredo, Arelis y Rudibel.

En general, agradezco a todos aquellos que están presentes y que pusieron su granito de arena para que un día como hoy, pudiera dedicarle estas palabras. A todos muchas gracias.

Sureny Orihuela Sánchez

En estos cinco años de estudio, muchas han sido las personas que de una forma u otra contribuyeron a lograr este resultado por lo que creo que es el momento de agradecerles.

Primeramente quiero agradecer a mi mamá Nancy López Rodríguez y a mi papá Luis Monteslier Rodríguez por su amor, cariño, apoyo y por la crianza que me dieron, y sobre todo por su sacrificio, sin ellos nada de esto hubiese sido posible.

A mi primo Yosvani Vázquez, a mi abuela Caruca, a Gloria y al chino por todo su amor, cariño y admiración. En general a toda mi familia por su apoyo.

A mi novia Yuliet Fernández, por su amor y su entrega incondicional en cada momento.

A mi compañera de tesis Sureny por su trabajo, su entrega y esmero en este trabajo pues de otra manera no hubiésemos podido alcanzar esta meta.

A mis tutores Julio Cesar Brito y Miguel Lezcano por estar siempre pendiente de todos los detalles y guiándonos en todo el desarrollo de este trabajo.

A mi compañero de aula René Leandro Cruz por ayudarme en todo momento que lo molesté para que me explicara alguna de mis tantas dudas, gracias mi hermano.

A mi amiga Yaily Leyva por su enseñanza desde el momento que llegue a la UCI y por los incontables favores que me ha hecho.

A mis compañeros de aula, del deporte y de las fiestas Julio German, Sandy, Ernesto, Jorge Ernesto, Marvel, Darlon, Alejandro, Osvaldo, Luis Angel, Yunet, Osman, Jeovani (El Potin) y Rolando.

A mi tribunal y a mi oponente, por corregir cada detalle y estar dispuestos a ayudarme.

A todos los compañeros que estuvieron estos años conmigo en los buenos y en los malos momentos y a todos los que de alguna forma tuvieron algo que ver en este trabajo.

Liandry Monteslier López

DEDICATORIA

A la mujer más linda y especial en mi vida, mi mamá Idania, por todo lo que ha hecho por mí, por aconsejarme cuando me equivocaba, por celebrar conmigo cada victoria, por acompañarme en cada una de mis tristezas y sobre todo por hacer de mí una gran persona. Te quiero mucho.

A mi papá José Miguel por darme su apoyo, cariño y confianza. Por enseñarme a ser fuerte y luchar por lo que quiero.

A mi abuelito Aldo por su amor, sus años de sacrificios, por criarme, educarme y guiarme por el mejor camino y sobre todo por ser el mejor abuelito del mundo.

A mi abuelita Mercedes por ser mi especial protectora, por cuidarme mucho y luchar siempre por mi bienestar. Por ser como mí segunda mamá y llevarme por el camino correcto, por todos sus consejos. Te adoro.

A mi padrastro David por ser tan cariñoso conmigo y acogerme como su hija.

A mi familia completa por confiar plenamente en mí y apoyarme en todas mis decisiones.

A mi novio Ariam que lo ha dado todo por mí y me ha enseñado lo que es el verdadero amor. Por amarme tanto y convertirme en una mujer, por estar presente cuando más lo he necesitado y por estar siempre a mi lado. Te amo

Sureny Orihuela Sánchez

A mi mamá y a mi papá por ser mis ejemplos y hacer de mi lo que soy hoy.
A mis hermanos por quererme tanto, espero ser un ejemplo para ellos.
A mis abuelos, a mi primo y a mi novia.
A toda mi familia por confiar y esperar siempre lo mejor de mí.

Liandry Monteslier López

RESUMEN

La presente investigación surgió en el marco de trabajo de la herramienta Generador Dinámico de Reportes (GDR), en el departamento de Integración de Soluciones del Centro de Tecnologías de Gestión de Datos (DATEC). Dicha herramienta tiene la finalidad de mostrar la información almacenada en una base de datos en forma de reportes. Este trabajo se desarrolló con el objetivo de ampliar el campo de conexión del GDR, aumentar su interoperabilidad con otras aplicaciones, independientemente de la plataforma donde fueron desarrolladas, y brindar más seguridad a los clientes en el proceso de conexión a las diferentes fuentes de datos. Para el desarrollo del trabajo se utilizó un conjunto de herramientas y tecnologías que permitieron diseñar e implementar las clases del componente y se siguieron las pautas que propone la metodología OpenUP para construir software con buenas prácticas. También se aplicaron pruebas funcionales, de rendimiento y de integración que validaron el correcto funcionamiento del componente implementado. Se obtuvo como resultado un mecanismo de comunicación entre el GDR y los orígenes de datos PostgreSQL y MySQL; lo que permitió obtener la información de forma indirecta, utilizando las potencialidades de los servicios web. Además brinda al cliente la posibilidad de crear su propio servicio web en el entorno de desarrollo que posea; debido a que se facilita la descripción de la interfaz para la integración a GDR. Finalmente se logró una conexión mediante un servicio web con una implementación de forma genérica, la cual puede ser utilizada por otros sistemas.

PALABRAS CLAVES: generadores de reportes, origen de datos, servicio web.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Actualmente la sociedad se encuentra inmersa en las nuevas tecnologías, donde la informática juega un papel fundamental en la mayoría de las esferas de la sociedad. Siendo difícil concebir un área que no utilice, de alguna forma, el apoyo de la misma; cubriendo desde la más sencilla organización hasta la compleja empresa. La informática tiene asegurado en sus diversas ramas un papel protagónico para el futuro, ya que es una disciplina formada por un conjunto de técnicas y conocimientos que hacen posible el tratamiento automático de la información por medio de computadoras.

Las Tecnologías de la Información y las Comunicaciones (TIC) incrementan su protagonismo dentro de los cambios y transformaciones de la sociedad actual, pues se han convertido en un medio que facilita el éxito de los objetivos de cualquier organización. Estas agrupan elementos y técnicas usadas en el tratamiento y transmisión de la información, permitiendo una interacción activa de sus usuarios. El desarrollo acelerado de las mismas ha logrado un incremento a nivel mundial de la informatización de las sociedades, lo que ha traído consigo el aumento de la capacidad de almacenamiento de la información, permitiéndole a la organización tener almacenados y organizados los datos que posee para cualquier gestión o cambio.

Cuba busca nuevas alternativas para colocarse dentro del ámbito de la industria tecnológica. Una de ellas fue la creación de la Universidad de las Ciencias Informáticas (UCI) destinada a la formación de ingenieros en ciencias informáticas altamente calificados en dicha rama y comprometidos con la revolución. Sus estudiantes son capaces de producir software y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación. La UCI se encuentra inmersa en el proceso de desarrollo de software y para ello cuenta con diversos centros productivos que aportan beneficios tanto al servicio de la universidad como del país.

El Centro de Tecnologías de Gestión de Datos (DATEC), es uno de los centros de desarrollo de software de la UCI. Su objetivo principal es proveer soluciones integrales, brindar soportes y consultorías relacionadas con tecnologías de bases de datos y análisis de información. Dicho centro contiene un conjunto de proyectos que tributan al desarrollo de herramientas y servicios informáticos, especializados tanto en el almacenamiento como en el análisis de datos. Además cuenta con herramientas que brindan la posibilidad de mostrar dichos datos, como son los generadores de reportes.

INTRODUCCIÓN

El Generador Dinámico de Reportes es una herramienta desarrollada por DATEC con la finalidad de mostrar la información almacenada en base de datos en forma de reporte. Su fácil integración y capacidad de extraer dichos datos lo convierten en una herramienta útil. Sin embargo, el GDR necesita establecer una conexión directa con las fuentes de datos, donde se le especifica una dirección única e irrepetible con la cual se identifica la computadora conectada a la red y el puerto del servidor de bases de datos, así como las credenciales para la conexión; por lo que puede ser considerado como una vulnerabilidad de seguridad. Además existen entidades como las Fuerzas Armadas Revolucionarias (FAR) y el Ministerio del Interior (MININT) que dentro de sus políticas de seguridad, no admiten que aplicaciones externas puedan consultar directamente sus bases de datos. Lo que trae como consecuencia que se pierdan oportunidades de negocio con este tipo de instituciones. De igual manera, el GDR necesita aumentar la interoperabilidad con otras aplicaciones de software independientemente de las propiedades que posean o de las plataformas sobre las que estén instaladas.

A partir de lo antes expuesto surge como **problema de la investigación**: ¿Cómo consultar los datos almacenados sin acceder directamente a las fuentes de datos para la generación de reportes desde el GDR?

Se define como **objeto de estudio**: Componentes de conexión en los sistemas generadores de reportes. Enmarcado en el **campo de acción**: Componente de conexión para el GDR mediante servicios web.

Para solucionar el problema planteado se define como **objetivo general**: Desarrollar un componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes.

Para cumplir este objetivo se desglosan los siguientes **objetivos específicos**:

- Definir conceptos, herramientas, tecnologías y metodología necesaria para el desarrollo del componente de conexión.
- Realizar el análisis y diseño del componente de conexión.
- Implementar el componente de conexión y realizar pruebas que validen su correcto funcionamiento.

Para dar cumplimiento a los objetivos se realizaron las siguientes **tareas de la investigación**:

- Análisis de los conceptos fundamentales para el desarrollo del componente de conexión basado en un servicio web.

- Estudio sobre el desarrollo de los servicios web.
- Selección de las herramientas y tecnologías para el desarrollo del componente de conexión.
- Identificación de los requerimientos del servicio web.
- Diseño del contrato del servicio web para la conexión.
- Implementación del servicio web de acceso a datos.
- Prueba del servicio web para satisfacer los Requisitos Funcionales (RF) y Requisitos No Funcionales (RNF) del componente de conexión.
- Despliegue del servicio web para su puesta en uso.
- Integración del componente de conexión al GDR.
- Diseño de las clases que conformarán las respuestas del servidor.
- Diseño de los casos de pruebas para determinar el correcto funcionamiento de los requisitos.
- Realización de pruebas al componente de conexión para validar su calidad.

El documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, anexos y glosario de términos.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS DEL COMPONENTE DE CONEXIÓN

En este capítulo se abordan los diferentes generadores de reportes, sus características y formas de conexión a sus fuentes de datos. Además se estudian varios conceptos y definiciones existentes sobre los servicios web, así como funciones, ventajas y desventajas de su uso. También se abordan aspectos importantes a tener en cuenta durante el desarrollo del proceso de la aplicación y se fundamenta el uso de tecnologías y herramientas así como la metodología que se utilizarán para dar solución a los objetivos planteados.

CAPÍTULO 2: DISEÑO DEL COMPONENTE

En este capítulo se realiza el diseño del componente, donde se generan los artefactos correspondientes al flujo de trabajo diseño. Se incluye además la realización de modelos y diagramas que permiten mostrar la estructura estática del sistema y la interacción de sus clases. También se profundiza en las actividades específicas del componente, permitiendo así asegurar una organización del software para alcanzar resultados satisfactorios.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL COMPONENTE

En este capítulo se generan los artefactos referentes a la fase de implementación y prueba del software. Se modela el sistema en términos de componentes y se dan a conocer las técnicas usadas en la solución del problema. También se especifican los casos de pruebas realizadas al componente para validar su correcto funcionamiento.

1 CAPÍTULO I: FUNDAMENTOS TEÓRICOS DEL COMPONENTE DE CONEXIÓN

Introducción

En este capítulo se hace un análisis de los generadores de reportes existentes, haciendo énfasis en los diferentes conectores que posibilitan la realización de dichos reportes. También se aborda sobre el uso de los servicios web y su utilización en los generadores de reportes. Además se definen las distintas herramientas, tecnologías y metodología empleada, con el objetivo de explotar al máximo sus potencialidades.

1.1 Definición de Reporte

Un reporte es un informe o una noticia que puede ser impreso, digital o audiovisual, pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos, persuasivos y específicamente en el ámbito de la informática los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y fácil de interpretar por los usuarios. Los reportes tienen diversos niveles de complejidad, desde una lista o enumeración hasta gráficos mucho más desarrollados. Según la herramienta informática y la base de datos en cuestión, los reportes permiten la creación de etiquetas y la elaboración de facturas. (Navarro Rodríguez, 2011).

Los reportes son de gran importancia pues permiten al usuario realizar de forma transparente, consultas a las bases de datos y obtener información de ellas en un determinado formato, con una mayor rapidez y un mayor nivel de detalle y flexibilidad.

1.2 Sistemas generadores de reportes

Los generadores de reportes son herramientas complementarias de los sistemas de información incluidos en la mayoría de los productos de software empresariales. Proveen una forma transparente al usuario para realizar consultas a las bases de datos y obtener información de ella en forma de reporte. Además, tienen la capacidad de interactuar con los resultados obtenidos basándose en los datos para tomar sus propias decisiones. (Silva Hernández, 2003).

1.2.1 iReport

iReport es un generador de informes visuales y un diseñador para Jasper Reports[1]. Herramienta que posibilita entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV y XML. Está escrito en el lenguaje de desarrollo Java, con una interfaz gráfica intuitiva y fácil de usar.

Esta herramienta permite que los usuarios corrijan visualmente informes complejos imágenes y subinformes. iReport está además integrado con JFreeChart, marco de software OpenSource para el lenguaje de programación Java, el cual permite la creación de gráficos complejos de forma simple. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, JavaBeans, XML. (Herrera , 2005).

Características

- Es gratuito y soporta conexiones a Hibernate mediante Spring, framework de desarrollo de código abierto de aplicaciones para la plataforma Java.
- Maneja el 98% de las etiquetas de Jasper Reports.
- Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de textos, cartas y subreportes.
- Soporta JDBC, API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

Sistema Operativo: Windows 2000, NT y XP.

Licencia: iReport cuenta con licencia GPL (Licencia Pública General) y se licencia como Freeware[2] para el sistema operativo Windows.

1.2.2 Crystal Reports

Crystal Reports es un producto de inteligencia empresarial, con alta tecnología para la creación e integración de reportes, con datos provenientes de múltiples SGBD (Sistemas Gestores de Base de

[1] **Jasper Reports:** es una herramienta libre de Java para la creación de informes.

[2] **Freeware**: tipo de software que se distribuye sin coste, disponible para su uso y por tiempo ilimitado.

Datos), entre los que se encuentran: PostgreSQL, MySQL, Oracle, DB2, MSSQL, Informix, InterBase, Sybase y Frontbase. Permite crear contenido interactivo con calidad de presentación en la plataforma .NET[3], debido a que es la herramienta de elaboración de informes estándar para Visual Studio .NET, lo que ha supuesto una ventaja fundamental para Crystal Reports durante años. (Navarro Rodríguez, 2011).

Características

- Permite diseñar informes ilimitados para uso en aplicaciones de Visual Studio con el diseñador de Crystal Reports integrado.
- Admite integrar el motor de Crystal Reports en aplicaciones de Microsoft Windows y aplicaciones web.

Sistema Operativo: Windows en todas sus versiones, Linux y Mac OS.

Licencia: Crystal Reports dispone de licencias por usuario designado para el diseño de reportes, independientemente de la edición de Crystal Reports adquirida. Se distribuye bajo los términos de la EULA, licencia por la cual el uso de un producto sólo está permitido para un único usuario, el comprador. Es software privativo y de uso restringido mediante el pago de patente.

1.2.3 Pentaho Reporting

La unidad de reportes de Pentaho (Pentaho Reporting) permite a las organizaciones acceder, dar formato y distribuir fácilmente la información a empleados, clientes y asociados. Pentaho provee acceso a fuentes de datos relacionales y basadas en XML, además de ofrecer varios formatos de salida como PDF, HTML, EXCEL o hasta texto plano. También permite llevar esta información a los usuarios finales vía web, correo, portales corporativos o aplicaciones propias.

Pentaho Reporting permite ir incrementando la plataforma de reportes a medida que las necesidades crecen. Es una herramienta independiente que forma parte de la unidad que simplifica el proceso de generación de reportes, permitiendo a los diseñadores de reportes crear rápidamente informes sofisticados y ricos visualmente basados en el proyecto de reportes de Pentaho JFreeReports[4].

[3] **.NET**: framework de Microsoft que permite un rápido desarrollo de aplicaciones.

[4] **JFreeReports**: es una librería de código abierto para la generación de reportes. Está desarrollada en el lenguaje Java.

Características

- Diseñador gráfico basado en “arrastrar y soltar” (drag & drop) que provee completo control de acceso a los datos, agrupaciones, cálculos, gráficas y formato para reportes de alta resolución.
- Plantillas de reportes aceleran el proceso de generación, proporcionando un aspecto consistente y atractivo.
- Opciones de salida flexibles incluyendo los populares formatos PDF, HTML, Excel.

Sistema Operativo: No existe dependencia de ningún sistema operativo.

Licencia: El software es gratuito y puede ser apoyado y desarrollado.

1.2.4 Eclipse BIRT

Eclipse BIRT, herramienta para el reporte y la inteligencia de negocio es un proyecto de software de código abierto que proporciona capacidades de creación de informes para aplicaciones web, especialmente aquellas basadas en Java y Java EE. Está conformado por dos componentes principales: un diseñador de informes visuales dentro de Eclipse IDE para crear informes BIRT, y un componente de rutina para generar informes que pueden ser puestos en uso en cualquier entorno Java. El proyecto BIRT también incluye un motor de gráficos que está integrado en el diseñador de informes y además puede ser usado por separado para incluir gráficas en una aplicación. Sus diseños de informes se hacen en XML. (Brito Rodríguez, 2012).

Sistema Operativo: Eclipse BIRT se encuentra disponible para los sistemas operativos Windows, Linux, Solaris, AIX, HP-UX y Mac OSX.

Licencia: Bajo licencia EPL (Licencia Pública Eclipse).

1.2.5 Generador Dinámico de Reportes

Generador Dinámico de Reportes (GDR) es una aplicación web que tiene como objetivo generar reportes de forma rápida, interactiva y con una amplia gama de alternativas para los usuarios. Permite a sus clientes consultar las bases de datos de sus organizaciones y poder generar reportes con la información que estos manejan, independientemente del SGBD que utilicen ya sea MySQL, Oracle o PostgreSQL. El desarrollo de dicho sistema está basado en tecnología web, lo cual hace del sistema una gran ventaja para las organizaciones puesto que lo mantiene disponible desde cualquier estación de trabajo, las 24 horas del día. Tiene una interfaz gráfica amigable y está orientado al usuario final. Se encuentra en constante

perfeccionamiento con el objetivo de que cada día, logre satisfacer las expectativas del mercado. Durante su implementación se emplearon tecnologías novedosas en el desarrollo web actual como son: PHP, JavaScript, Symfony, Propel, Ext-JS, OpenJacob, XML, PostgreSQL. Es una aplicación multiplataforma y soporta imágenes, gráficas y varios orígenes de datos. (Rodríguez Durán, 2011).

Desde el punto de vista arquitectónico ofrece las siguientes ventajas:

- La aplicación del estilo multicapas atribuye una lógica organizacional que conserva la alta cohesión y el bajo acoplamiento entre los componentes del sistema.
- La aplicación del estilo Modelo-Vista-Controlador facilita el mantenimiento del sistema al desacoplar los componentes.
- El uso de marcos de trabajo en cada una de las capas fomenta la reutilización, acelera el tiempo de desarrollo y provee una estructura al código fuente que facilita su mantenimiento.
- Los diferentes entornos de despliegue del sistema le aportan flexibilidad y escalabilidad a la arquitectura del sistema, una vez que puede ser adaptado a las necesidades particulares de una empresa u organización que requiera los servicios del generador de reportes.

Sistema Operativo: El GDR encuentra disponible para los sistemas operativos Windows y Linux.

Licencia: Está bajo licencia LGPL (Licencia Pública General Reducida).

1.2.6 Comparación entre las herramientas analizadas

Después de haber analizado estas herramientas de generación de reportes se puede apreciar que poseen diversos aspectos que determinan sus potencialidades; aspectos importantes a tener en cuenta a la hora de realizar una comparación, algunos de ellos se recopilan en las siguientes tablas (ver tablas 1 y 2).

Herramientas de reportes	**iReport**	**Crystal Reports**	**Pentaho Reporting**	**Eclipse BIRT**	**GDR**
Sistemas Operativos					
Windows	**Si**	**Si**	**Si**	**Si**	**Si**
Linux	**No**	**Si**	**Si**	**Si**	**Si**

Mac OS	**No**	**Si**	**Si**	**Si**	**Si**

Tabla 1. Comparación de los sistemas operativos

Licencia					
GPL	**Si**	**No**	**Si**	**No**	**No**
EPL	**No**	**No**	**No**	**Si**	**No**
EULA	**No**	**Si**	**No**	**No**	**No**
LGPL	**No**	**No**	**No**	**No**	**Si**

Tabla 2. Comparación de la licencia

El estudio realizado a los diferentes generadores de reportes en cuanto a los sistemas operativos donde se desarrollan demostró que son multiplataforma, exceptuando el iReport que solo es utilizable en Windows. Además se analizaron los tipos de licencia donde se evidenció que hay generadores tantos privativos como públicos, así como el GDR que posee una licencia pública general reducida, lo que no ocasionará ningún inconveniente para desarrollar el componente de conexión.

1.3 Diseñador de modelos

Los modelos de datos son herramientas conceptuales que describen la información de las bases de datos, las relaciones entre los datos de la misma, su semántica y sus limitantes. Representan la información de forma sencilla y global. El nivel de diseño de un modelo de datos es aquel que se le presenta al usuario final y que puede tener combinaciones o relaciones entre los objetos que conforman la base de datos global. (Brito Rodríguez, 2012).

Definir o crear un informe consiste en indicar al Diseñador de Modelos de dónde tiene que obtener los datos necesarios, cómo tiene que transformarlos y por último, cómo debe presentar cada elemento de datos sobre el documento final.

Origen de datos

Es la unidad que contiene la información necesaria para obtener acceso a las fuentes de datos. Este término se emplea para referirse al objeto utilizado para establecer las conexiones con las bases de datos. Un origen de datos especifica un proveedor de datos y la configuración del resto de las propiedades de la cadena de conexión utilizada para obtener el acceso a su información. (Brito Rodríguez, 2012).

Modelo semántico

Es la forma de representar una base de datos de manera conceptual. Permiten captar mejor el significado o la semántica de la información contenida en la base de datos. Un objeto semántico es una representación de algunos elementos identificables en el ambiente de trabajo de los usuarios. El modelo semántico es un conjunto de atributos que describen con eficacia una identidad bien determinada. (Brito Rodríguez, 2012).

1.3.1 Diseñador de Modelos del iReports

Los datos a visualizar pueden ser recogidos de diferentes maneras, incluyendo conexiones a múltiples JDBC, (API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java).

Para establecer una conexión mediante iReports con una base de datos, se debe proporcionar el JDBC correspondiente, hay versiones como la 0.5.1 que ya lo contiene para establecer conexiones con bases de datos como MySQL y Access. Una vez que se configura el origen de datos, en caso que no esté configurada, se procede a configurar el iReports para establecer la conexión y tener acceso a los datos, para esto se debe ir llenando todos los campos necesarios como se observa en la Figura 1.

Figura 1 . Propiedades de la conexión

Figura 2. Comprobación de la conexión

Una vez cumplidos estos requisitos se comprueba la conexión y si esta fue exitosa se salvan los datos y se muestra un mensaje de confirmación (ver Figura 2).

1.3.2 Diseñador de Modelos del Crystal Reports

Cuando se trabaja con Visual Basic .Net y Crystal Report, se conecta a SQL Server u Oracle como fuente de datos, debido a que tienen integración con el IDE de Visual Studio, para facilitar la creación de los mismos. Pero cuando se habla de motores como PostgreSQL y MySQL, no se tiene la integración del IDE con el servidor de base de datos, esto hace que no se pueda crear tan sencillamente los reportes como se hace con los otros motores, pero no lo hace imposible, sólo se debe seguir unos pasos distintos, que también sirve para los dos primeros motores mencionados anteriormente. (devTroce, 2010).

El primer paso a la hora de diseñar el informe es indicarle al asistente dónde están y cuáles son los datos que se van a utilizar en el informe. Aquí se tiene que indicar:

a) La tecnología a utilizar para conectarse al origen de datos. En este caso, tratándose de una base de datos SQL Server, se debe indicar **Crear nueva conexión | OLE DB (ADO)** y luego elegir el proveedor '**Microsoft OLE DB Provider for SQL Server**'.

b) El nombre del servidor, la base de datos, el usuario y la contraseña necesarios para establecer la conexión. En este caso, se utilizará como nombre de servidor **.\SQLExpress** ('.' significa la máquina local, y 'SQLExpress' es el nombre de la instancia en que se instala SQL Server Express de forma predeterminada). De momento se utilizará la autenticación integrada (marcando la casilla correspondiente), aunque la autenticación de usuario/contraseña es más utilizada en la vida real, sobre todo en aplicaciones web.

Una vez indicados el servidor y la base de datos, estos pasan a la lista de los orígenes de datos disponibles y llega el momento de indicar la(s) tabla(s), vista(s) o procedimiento almacenado de la base de datos de donde se deberá obtener la información y seguido de esto seleccionar cuáles de las columnas (campos) que componen la tabla elegida se quiere mostrar en el listado. Por último, el asistente permite elegir entre un conjunto de plantillas de estilos disponibles para establecer las características visuales y así genera el informe que se ve reflejado en el área de trabajo de Visual Studio 2005.

1.3.3 Diseñador de Modelos del Pentaho Reporting

La parte más importante cuando se está creando un informe es la definición del DataSet, pues va a determinar los datos que se van a poder utilizar dentro del informe. Pentaho Report Design permite trabajar con varios tipos de orígenes de datos, como son JDBC, Pentaho Data Integration, origen de datos OLAP, fichero XML. A continuación se muestra un ejemplo sencillo utilizando JDBC como origen de datos.

En primer lugar, se configura el origen de datos al que se va a conectar utilizando JDBC. Seguidamente se definen las consultas que van a determinar los datos devueltos desde el origen de datos, tal y como se observa en la Figura 3.

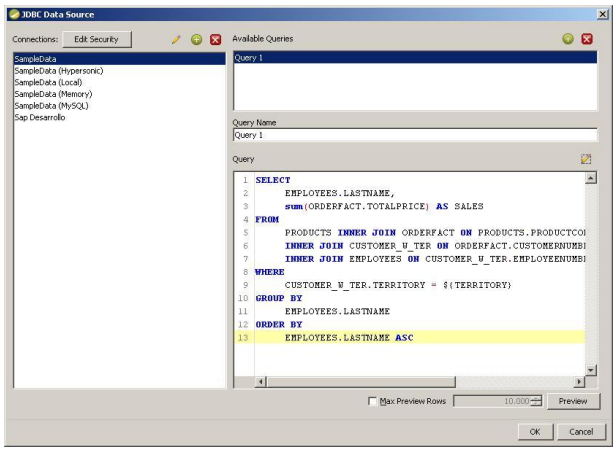


Figura 3. Configuración de DataSet Origen JDBC

Para la construcción de las consultas, se tiene un asistente que permite ver las tablas disponibles e ir construyendo las diferentes secciones de la sentencia SQL. También se dispone de la opción “Preview” para ir visualizando los resultados que devolvería la ejecución de la consulta (ver Figura 4).

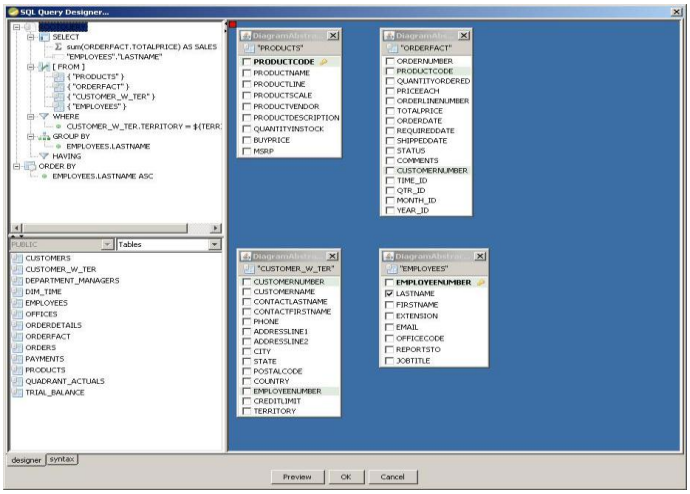


Figura 4. Definición de la consulta

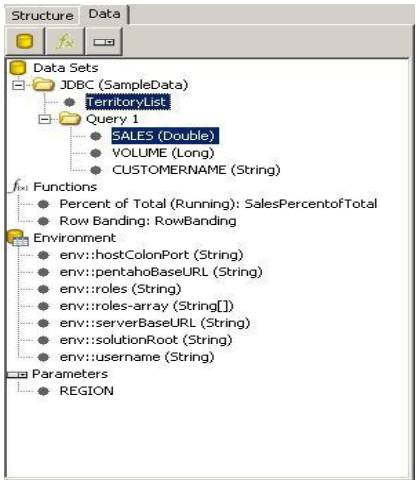


Figura 5. Consultas definidas

Una vez concluida la definición de la consulta, aparece en la parte derecha de la aplicación, en la sección Data (tal y como se ve en la Figura 5). Además se tienen todas las consultas que se hayan definido y la lista

de campos que devuelve la consulta, para poder ser utilizados en las diferentes secciones y controles del reporte.

1.3.4 Diseñador de Modelos del Eclipse BIRT

A la hora de hacer la conexión para el Eclipse BIRT lo primero que se hace es añadir la vista de Data Source donde se configurará la conexión con la base de datos (ver Figura 6).

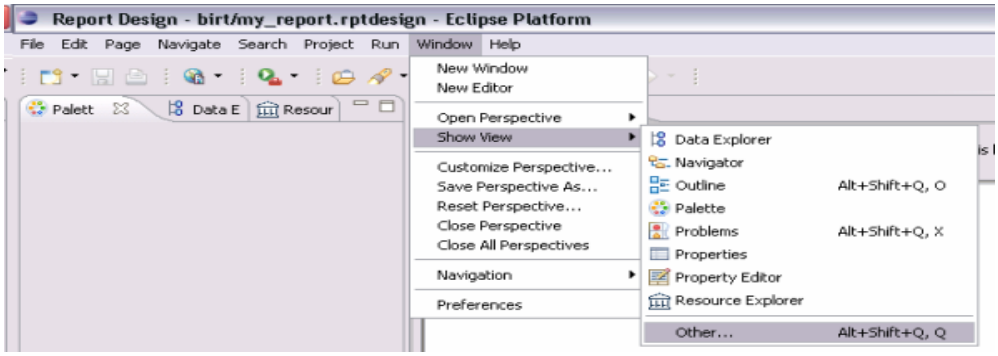


Figura 6. Configuración de la Base de Datos

Luego se elige la vista Data Source Explorer que aparece dentro de "Connectivity" (ver Figura 7).

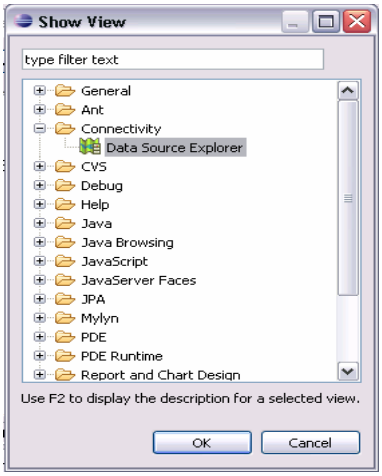


Figura 7. Selección de la vista

Así se crea el JDBC Data Source que da paso a la ficha con la información requerida para hacer la conexión, esto se observa en la Figura 8.

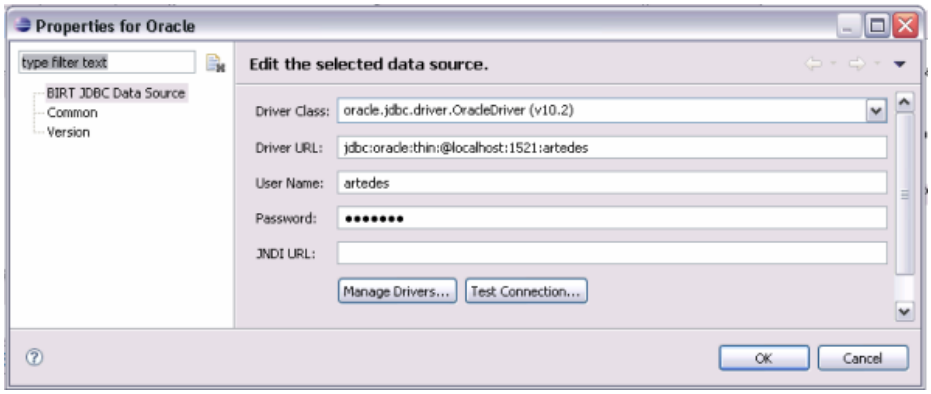


Figura 8. Propiedades para la conexión

Una vez hecha la conexión se obtiene la información necesaria para confeccionar el informe.

1.3.5 Diseñador de Modelos del GDR

El GDR v1.8 contiene un módulo Diseñador de Modelos el cual es el encargado de conectarse a un origen de datos, utilizando gestores de base de datos PostgreSQL, Oracle, SQLite, MySQL o SQL Server para luego diseñar los modelos semánticos que serán utilizados en la confección de los reportes. El Diseñador de Modelos permite visualizar, adicionar, modificar y eliminar los orígenes de datos. Además, muestra los modelos previamente creados y los elementos que los componen (tablas, vistas y rutinas) y cuenta con una API que garantiza entre otras funcionalidades la exportación de los reportes y la obtención del catálogo de reportes, aspectos esenciales para su integración con otros sistemas.

Actualmente el diseñador de modelos del GDR v1.8 brinda soporte a los siguientes gestores: Microsoft SQL Server (mssql), MySQL Server (mysql), Oracle Server (oracle), PostgreSQL (postgresql) y SQLite (sqlite). Los parámetros requeridos para crear un nuevo origen de datos son: nombre con que se registra, tipo de gestor, dirección IP[5] del servidor de base de datos, puerto que usa el servidor, usuario para conectarse al servidor, clave de dicho usuario y la base de datos de la cual se extraerá la información (ver Figura 9).

[5] **Dirección IP**: dirección única e irrepetible con la cual se identifica la computadora conectada a la red.

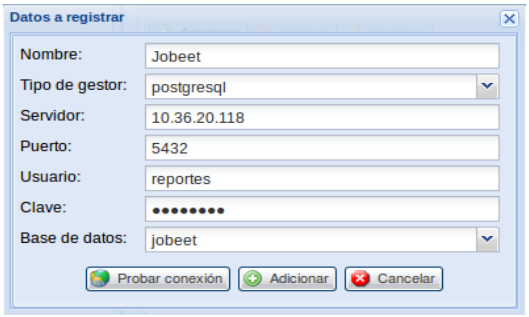


Figura 9. Datos a registrar

1.3.6 Comparación entre las herramientas analizadas

Herramientas de reportes	iReports	Crystal Reports	Pentaho Reporting	Eclipse BIRT	GDR
Propiedades					
Múltiples origen de datos	**Si**	**Si**	**No**	**Si**	**No**
Orígenes de datos combinados	**No**	**Si**	**No**	**Si**	**No**
Transformaciones de datos	**Si**	**Si**	**No**	**Si**	**Si**

Tabla 3. Comparación de las propiedades

Fuentes de datos					
JDBC	**Si**	**Si**	**Si**	**Si**	**Si**
XML	**Si**	**Si**	**Si**	**Si**	**Si**
Servicio web	**No**	**No**	**No**	**Si**	**No**
Hibernate	**Si**	**Si**	**Si**	**No**	**Si**

Tabla 4. Comparación de las fuentes de datos

Tras realizar un estudio de los componentes que intervienen en el proceso de diseño de modelos se evidenció que aportan información para la concepción del nuevo componente, desde la perspectiva en que gestionan diversos orígenes de datos. Además se puede observar que todos poseen una serie de

mecanismos para conectarse a las distintas fuentes de datos, donde solo el Eclipse BIRT soporta conexiones mediante servicios web no siendo así en los demás generadores.

1.4 Servicio web

La necesidad de estandarizar la comunicación entre diversas plataformas y lenguajes de programación provoca el surgimiento de los servicios web. Estos tienen múltiples definiciones, lo que demuestra su complejidad para brindar una adecuada definición que englobe todo lo que son e implican.

Una definición del W3C[6] dice que: "Los servicios web son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer una serie de servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la web. A su ves proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario."

Los servicios web surgen en la tentativa de encontrar una mejor integración entre la arquitectura del Internet y la arquitectura de las aplicaciones web. Actualmente la mayoría se basan en programas que operan en entornos de servidores de aplicaciones como Apache. Los servicios web se acercaron al concepto de interfaces y comunicaciones definidas en XML, para finalmente unir aplicaciones de cualquier tipo. Además proporcionan la libertad de redefinir su implementación tras peticiones de nuevos requerimientos. Los servicios web se distinguen de otras tecnologías por la versatilidad de XML. Lo que permite separar la estructura gramatical (sintaxis) del significado gramatical (semántica). Los objetos pueden ser definidos como servicios que se comunican con otros servicios en la gramática definida por XML, donde cada uno traduce y analiza el mensaje de acuerdo con la implementación local y el entorno. Debido a que una aplicación conectada en red puede efectivamente estar compuesta por varias entidades con construcciones y diseños diferentes, a condición de que cumplan con las reglas definidas por su arquitectura orientada a servicios.

[6] **W3C:** World Wide Web Consortium, es un consorcio internacional que produce recomendaciones para la World Wide Web.

El uso de los servicios web en el desarrollo de este trabajo permite, en la aplicación a desplegar, un ambiente diferente, que pueda intercambiar información con los orígenes de datos existentes, sin importar las plataformas en que estos se encuentren o sus propiedades.

Los servicios web son considerados herramientas altamente populares debido a una serie de ventajas que poseen, entre ellas:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

No son aplicaciones con una interfaz gráfica con la que las personas puedan interactuar, sino que son software accesible en internet por otras aplicaciones. De esta forma podemos desarrollar aplicaciones que hagan uso de otras aplicaciones que estén disponibles en internet interactuando con ellas. Básicamente los servicios web permiten que diferentes aplicaciones, realizadas con diferentes tecnologías, y ejecutándose en toda una variedad de entornos, puedan comunicarse e integrarse.

1.4.1 Estándares para servicios web

Un punto clave en los servicios web es la interoperabilidad donde las distintas aplicaciones, en lenguajes de programación diferente y ejecutada sobre cualquier plataforma, puedan utilizar estos para intercambiar datos. Debido a que se asientan sobre protocolos y estándares abiertos ya existentes y muy difundidos (HTTP, XML, SOAP, WSDL).

Los principales estándares para el desarrollo de servicios web son los siguientes, (ver Figura 10):

- SOAP (*Simple Object Access Protocol*)
- WSDL (*Web Services Description Language*)
- UDDI (*Universal Description, Discovery and Integration*)

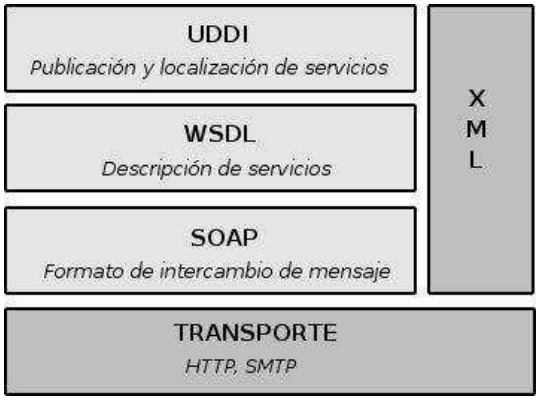


Figura 10. Arquitectura de los servicios web

SOAP

SOAP es un protocolo de mensajería XML extensible que forma la base de los servicios web. Proporciona un mecanismo simple y consistente que permite a una aplicación enviar mensajes XML a otra aplicación. Un mensaje SOAP es una transmisión desde un emisor SOAP a un receptor, y cualquier aplicación puede participar en este intercambio como emisor o receptor. Los mensajes SOAP se pueden combinar para soportar muchos comportamientos de comunicación, incluyendo, solicitud/respuesta, respuesta solicitada, mensajería asíncrona de una vía, o incluso notificación.

Es un protocolo de alto nivel que sólo define la estructura del mensaje y unas pocas reglas para su procesamiento. Es completamente independiente del protocolo de transporte subyacente, por eso los mensajes SOAP se pueden intercambiar sobre HTTP o protocolos de transporte de correo. Dentro del paradigma orientado a objetos, usar un servicio web es igual que usar cualquier otra clase. Y esto significa instanciarlo, y llamar a sus métodos, pasándoles los parámetros que sean necesarios, y obteniendo a su vez el resultado que se retornen. SOAP define precisamente cómo se debe codificar las llamadas a los métodos de un servicio web, y cómo debe el servicio web codificar el resultado para que se pueda interpretar. Estos mensajes son los que transportarán los protocolos de transporte, por lo general, HTTP.

Ventajas de SOAP:

- No está asociado con ningún lenguaje.
- No se encuentra fuertemente asociado a ningún protocolo de transporte.
- No está atado a ninguna infraestructura de objeto distribuido.

- Aprovecha los estándares existentes en la industria.
- Permite la interoperabilidad entre múltiples entornos.

WSDL

WSDL es un fichero XML usado para describir la interfaz de un servicio web como un conjunto de puntos finales de comunicación (métodos) capaces de intercambiar mensajes, es decir recibir llamadas con sus parámetros correspondientes y generar respuesta con el resultado que le corresponda. Este proporciona toda la información necesaria para acceder y utilizar un servicio web. Describe los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo.

Un documento WSDL describe qué hace el servicio web, cómo se comunica, y dónde reside. Se considera parte integral de UDDI, que es un registro de servicio web XML. Es el lenguaje usado por UDDI para describir a los servicios web; donde las operaciones y los mensajes que soporta se describen en abstracto, se ligan después al protocolo concreto de red y al formato del mensaje.

UDDI

Es un directorio de servicio web distribuido y basado en web que permite que se listen, busquen y descubran este tipo de software. Se podría comparar con las típicas páginas amarillas. El registro en el directorio se hace en XML. UDDI es una iniciativa industrial abierta entroncada en el contexto de los servicios web. El registro de un negocio en UDDI tiene tres partes: Páginas blancas - dirección, contacto y otros identificadores conocidos. Páginas amarillas - categorización industrial basada en taxonomías. Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

UDDI es uno de los estándares básicos de los servicios web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios web del catálogo de registros. (Sánchez Góngora, 2009).

1.4.2 Ciclo de vida de los servicios web

Para desarrollar un servicio web hay que tener en cuenta el ciclo de vida por el que se rige, siguiendo cada una de sus fases para lograr con éxito su puesta en uso. El primer paso es identificar los requerimientos que manejará el servicio web, aprobado el servicio se procede al diseño del contrato para la conexión y posteriormente a la implementación. Luego de implementado se realizan las pruebas para satisfacer los

Requisitos Funcionales (RF) y los Requisitos No Funcionales (RNF) del componente. Una vez probado se realiza el despliegue para su puesta en uso. En caso de que el servicio requiera de nuevos cambios ya sea nuevas funcionalidades o se quiera una nueva versión se comienza desde la etapa de diseño. Esta explicación se puede ver claramente en la Figura 11.

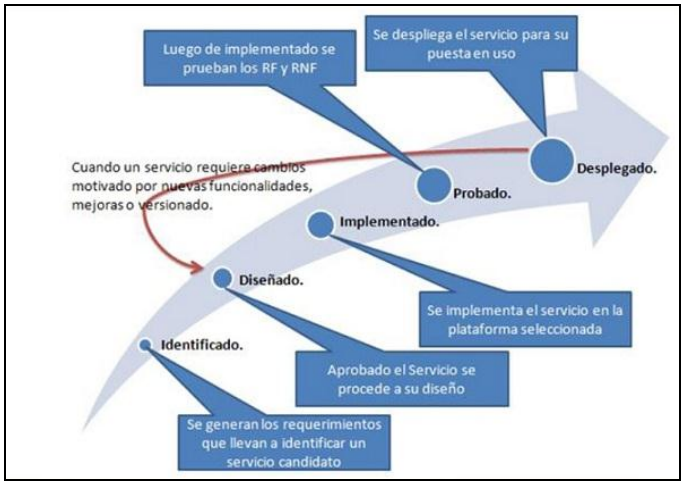


Figura 11. Ciclo de vida de un servicio web

1.4.3 Usos de los servicios web

El uso de los servicios web como protocolos de comunicación representa la capacidad de la SOA (Arquitectura Orientada a Servicios). SOA describe un sistema entero de servicios que buscan dinámicamente los unos a los otros, se unen para realizar alguna aplicación y se recombinan de varias formas. Ese modelo fomenta la reutilización de la tecnología y de software, lo que produce una evolución en la forma de diseñar, desarrollar y poner en uso las aplicaciones. En ese nivel, los desarrolladores de software necesitan pensar en el modelo SOA y diseñar su aplicación distribuida con ese modelo. Ese nivel se caracteriza por el uso de tecnologías para permitir las comunicaciones distribuidas de los servicios.

1.5 Ambiente de desarrollo

Las herramientas y tecnologías son programas o aplicaciones que se crean y se diseñan para efectuar una o varias funciones determinadas. Estas se utilizan en el desarrollo de aplicaciones web, debido a que desempeñan un papel fundamental en su evolución. Con el objetivo de definir ventajas, oportunidades y con la disposición de potenciar al máximo las prestaciones de las mismas se realiza el siguiente estudio.

1.5.1 Lenguajes de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; permitiendo al desarrollador comunicarse con los dispositivos de hardware y software existentes. Los programas creados pueden ser usados para controlar el comportamiento físico y lógico de una máquina.

PHP

PHP 5.3 (Procesador de Hipertexto) es un lenguaje interpretado de propósito general ampliamente utilizado en el mundo, puede ser incrustado dentro de código HTML (Lenguaje de Marcado de Hipertexto). Este lenguaje está diseñado especialmente para incrementar el dinamismo de las páginas web, aprovechando los recursos de las redes informáticas y los escasos requerimientos de hardware que solicita el lenguaje para que sus aplicaciones funcionen correctamente.

Es un lenguaje que no requiere definición de tipos de variables aunque se pueden evaluar por el tipo de datos que estén manejando en tiempo de ejecución. El código fuente escrito es invisible al navegador ya que es el servidor el encargado de ejecutar el código y enviar su resultado HTML al cliente, logrando una programación segura y confiable. Además permite aplicar técnicas de programación orientada a objetos y al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. (Hernández Hernández, 2009).

JavaScript

JavaScript es un lenguaje basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet. Además es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Los programas JavaScript van incrustados en los documentos XHMTL o en un fichero ".js", y se encargan de realizar acciones en el cliente, como puede ser: pedir datos y/o confirmaciones, mostrar mensajes, crear animaciones y comprobar campos. (Brito Rodríguez, 2012).

Lenguaje para la transferencia de información JSON

JSON, (Notación de Objetos de JavaScript), genera un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso del lenguaje de marcas extensible (XML). Es muy sencillo de usar, especialmente como alternativa a XML en AJAX. Una de sus ventajas sobre XML como formato de intercambio de datos es que prevalece un formato mucho más sencillo

a la hora de escribir un analizador semántico del mismo y los datos en JSON ocupan mucho menos que en XML. Además de que es fácil de leer para los humanos y máquinas y su procesamiento por parte de los ordenadores es rápido, pues necesita librerías muy pequeñas para trabajar con él. (json.org, 2012).

Lenguaje para la representación del modelo de datos XML

XML (Lenguaje de Etiquetado Extensible), conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades y contiene tres características muy importantes que son: extensibilidad, estructura y validación. Este lenguaje facilita la integración desde fuentes de datos heterogéneas, por ejemplo, páginas web y distintas bases de datos; permite proporcionar diferentes vistas sobre los datos (HTML, PDF), dependiendo de quién sea el cliente. (O´Really XML, 2010).

1.5.2 Entorno de Desarrollo

NetBeans

NetBeans IDE 7.1 es un entorno de desarrollo integrado y distribuido por SUN Microsystems bajo licencia dual GPL (Licencia Pública General) y CDDL (Licencia Común de Desarrollo y Distribución). Permite a los programadores escribir, compilar, depurar y ejecutar programas. Provee la implementación del patrón de arquitectura Modelo-Vista-Controlador, además de estar enfocado para el desarrollo del lenguaje dinámico de programación PHP. Es fácil de instalar y configurar en la mayoría de las plataformas. Brinda soporte para el framework Symfony, el cual es utilizado en la aplicación debido a que brinda la posibilidad de facilitar y automatizar una gran cantidad de funcionalidades que resultan comunes en las aplicaciones web. (Henández Carvajal, 2012).

1.5.3 Servidor de Aplicaciones

Apache

Apache 2.2 constituye una tecnología gratuita de código abierto. Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Permite la creación de sitios web dinámicos mediante el uso de Server Side Includes (SSI), de lenguajes de scripting como PHP y JavaScript. Se ejecuta en varios sistemas operativos. Posee gran cantidad de extensiones para diversas tecnologías, además de una amplia documentación, es libre, modular y multiplataforma. Se señala como desventaja que no posee interfaz gráfica que facilite su configuración. (Lobo, 2009).

1.5.4 Herramientas de Base de Datos

PostgreSQL

PostgreSQL 9.0.3 es un gestor de bases de datos muy específico dedicado a servir una interfaz entre las bases de datos, los usuarios y las aplicaciones que lo utilizan. Su propósito general es el de manejar de una manera más clara, sencilla y ordenada los conjuntos de datos que se convierten en información relevante para una organización. En esta nueva versión incorpora nuevas características y cuenta con un enfoque que se centra en la administración y vigilancia. Proporcionan una notable mejora en el tiempo de ejecución y hace más sencillo el análisis de datos avanzados. Es multiplataforma y está diseñado para ambientes de alto volumen. (Brito Rodríguez, 2012).

PgAdmin III

PgAdmin III es una herramienta para la administración del gestor de base de datos PostgreSQL. Esta permite crear simples consultas SQL o consultas de una alta complejidad de una manera más fácil, debido a que se le ha incorporado una nueva interfaz gráfica, la cual resulta de gran ayuda en el momento de gestionar las bases de datos. Es una herramienta de código abierto respaldada por una amplia comunidad de desarrolladores que encaminan sus esfuerzos al perfeccionamiento de la misma. Esta aplicación incluye un editor de resaltado de sintaxis SQL, una interfaz administrativa gráfica y una herramienta de consulta SQL. La conexión al servidor se puede hacer a través de protocolo TCP/IP. No se requieren controladores adicionales para comunicarse con el servidor de base de datos. (Pgadmin, 2011).

1.5.5 AJAX

AJAX (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de estas que trabajan conjuntamente (XHTML o HTML, CSS, DOM, XMLHttpRequest[7] y XML). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. (Yanes León, 2011).

1.5.6 Marcos de trabajo

[7] **XMLHttpRequest** objeto empleado para intercambiar datos de forma asíncrona con el servidor web.

Symfony

Symfony 1.1.7 es un marco de trabajo que facilita y automatiza una gran cantidad de funcionalidades que resultan comunes a la hora de realizar algunos tipos de software. Permite optimizar el desarrollo de las aplicaciones web y está desarrollado completamente con PHP5, siendo sencillo de usar pero lo suficientemente flexible como para adaptarse a los casos más complejos. Es una herramienta fácil de instalar y configurar en la mayoría de las plataformas ya que se puede ejecutar tanto en sistemas Windows y Unix estándares. Independiente del sistema gestor de bases de datos, su capa de abstracción y el uso de Propel, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto. (Gazquez Martínez, 2011).

Ext-JS

Ext-JS 2.2 es una librería JavaScript para la creación de aplicaciones enriquecidas del lado del cliente haciendo un uso intensivo de las tecnologías AJAX, XHTML, DHTML y DOM. Sus características principales son: gran desempeño, componentes de interfaz de usuario personalizables, con buen diseño y documentación. Sirve de puente entre las librerías JS más usadas (Prototype, JQuery, YUI), debido a que se inició como una extensión de YUI (Yahoo User Interface) esta presenta una cierta ventaja de compatibilidad respecto a las otras dos. (Ruiz Durán, y otros, 2010).

1.5.7 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado o UML 2.1 es un lenguaje para especificar, construir, visualizar y documentar los artefactos, información que se utiliza o produce mediante un proceso de software. Este lenguaje de modelado no es una guía para realizar el análisis y diseño orientado a objeto, es decir, no es un proceso, es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos informáticos, de arquitectura o de cualquier otra rama. Además ayuda a una rápida construcción de aplicaciones con mejor calidad y promueve la reutilización. (Larman, 1999).

1.5.8 Herramienta CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadoras), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costos de las mismas en términos de tiempo y de dinero. Estas herramientas son de mucha ayuda en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto,

cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores.

Visual Paradigm 8.0 es una herramienta de modelado profesional que hace uso del Lenguaje Unificado de Modelado. Una característica esencial de Visual Paradigm es que soporta el ciclo de vida completo del desarrollo de software. Además ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (Visual Paradigm for UML, 2011).

Esta herramienta se utilizará para visualizar y diseñar los elementos de nuestra aplicación debido a que es multiplataforma. Se puede intercambiar diagramas UML y modelos con otras herramientas. Posee facilidades para el diseño de los diagramas necesarios y su documentación, además de que la UCI cuenta con la licencia para su uso.

1.5.9 Metodología de desarrollo de software

Una metodología de desarrollo no es más que una colección de documentación formal referente a procesos, políticas y procedimientos que intervienen en el desarrollo del software. Es una guía que muestra paso a paso las tareas y actividades que se deben ir realizando para obtener el producto con buena calidad, así como el papel fundamental que debe enfrentar cada integrante en el transcurso del proyecto.

La finalidad de una metodología de desarrollo es garantizar la eficacia, cumpliendo los requisitos iniciales y la eficiencia, minimizando las pérdidas de tiempo en el proceso de generación de software. Existen diversas metodologías, se podrían nombrar algunas como: RUP, que es completa, XP (Extreme Programming), que es una metodología para proyectos de corto plazo y OpenUP.

1.5.10 OpenUP

OpenUP es un proceso que mantiene las mismas características de RUP, que contiene el desarrollo iterativo, casos de uso y escenarios de conducción de desarrollo, gestión de riesgos y el enfoque centrado en la arquitectura. El ciclo de vida del proyecto provee a los interesados un mecanismo de supervisión y dirección para controlar los fundamentos del proyecto, su ámbito, la exposición a los riesgos, el aumento de valor y otros aspectos.

Esta metodología estructura el ciclo de vida de un proyecto en cuatro fases:

1. **Concepción**: primera fase en el proyecto del ciclo de vida, acerca del entendimiento del propósito y objetivos obteniendo suficiente información para confirmar que el proyecto debe hacer. El objetivo de ésta fase es capturar las necesidades de los stakeholder (clientes) en los objetivos del ciclo de vida para el proyecto.

2. **Elaboración**: se trata los riesgos significativos para la arquitectura. El propósito de esta fase es establecer la base la elaboración de la arquitectura del sistema.

3. **Construcción**: esta fase está enfocada al diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito de esta fase es completar el desarrollo del sistema basado en la arquitectura definida.

4. **Transición**: es la última fase, cuyo propósito es asegurar que el sistema es entregado a los usuarios, evalúa la funcionalidad y rendimiento del último entregable de la fase de construcción.

1.6 Conclusiones Parciales

En este capítulo se analizaron los diferentes tipos de generadores de reportes, así como los diseñadores de modelos de cada uno de ellos, y los servicios web, como herramientas importantes de comunicación. También se estudiaron las herramientas y tecnologías para el desarrollo del componente de conexión donde se seleccionaron las siguientes: PHP 5.3 como lenguaje de desarrollo, para incrementar el dinamismo de la aplicación y utilizar las potencialidades de la red , el IDE NetBeans 7.1 por ser libre y tener gran integración con PHP y Symfony. Apache 2.2 como servidor flexible, rápido y multiplataforma. Además el marco de trabajo Symfony 1.1.7, para que atienda las peticiones del servidor y Ext-JS 2.2 como tecnología del lado del cliente. Se empleará el Visual Paradigm 8.0 como herramienta de modelado, para documentar y visualizar los resultados se hará uso del lenguaje de modelado UML 2.1. En todo momento se seguirán las pautas que presenta la metodología de desarrollo OpenUP ya que esta propone procesos ágiles y ligeros para construir software con buenas prácticas.

2 CAPÍTULO II: DISEÑO DEL COMPONENTE DE CONEXIÓN

Introducción

En este capítulo se realiza el diseño del componente conexión. También se profundiza en las actividades específicas del proceso, permitiendo así asegurar una organización del software para alcanzar resultados satisfactorios. El capítulo incluye además la realización del modelo de dominio, la definición de requisitos funcionales y no funcionales, el diagrama de casos de uso del sistema, el diagrama de clases del diseño mediante los cuales se muestra la estructura estática del sistema, el diseño del servicio web implementado y los diagramas de interacción donde se representan las clases y sus relaciones. Además, se especifica la estructura física de la solución que se propone mediante el diagrama de despliegue.

2.1 Modelo de Dominio

El Modelo de Dominio es una representación visual del entorno real de los objetos del proyecto o de las clases conceptuales que se centra en una parte del negocio, la relacionada con el ámbito del proyecto. Es un diagrama con los objetos reales que existen, relacionados con el sistema que se va a desarrollar y las relaciones que existe entre ellos. Este modelo se crea para documentar el vocabulario del sistema, que ayuda a comprender los conceptos que utilizan los usuarios, con los que trabajan y con los que deberá trabajar la aplicación. (Brito Rodríguez, 2012).

A continuación se presenta el Modelo de Dominio del componente de conexión y se describen sus clases.

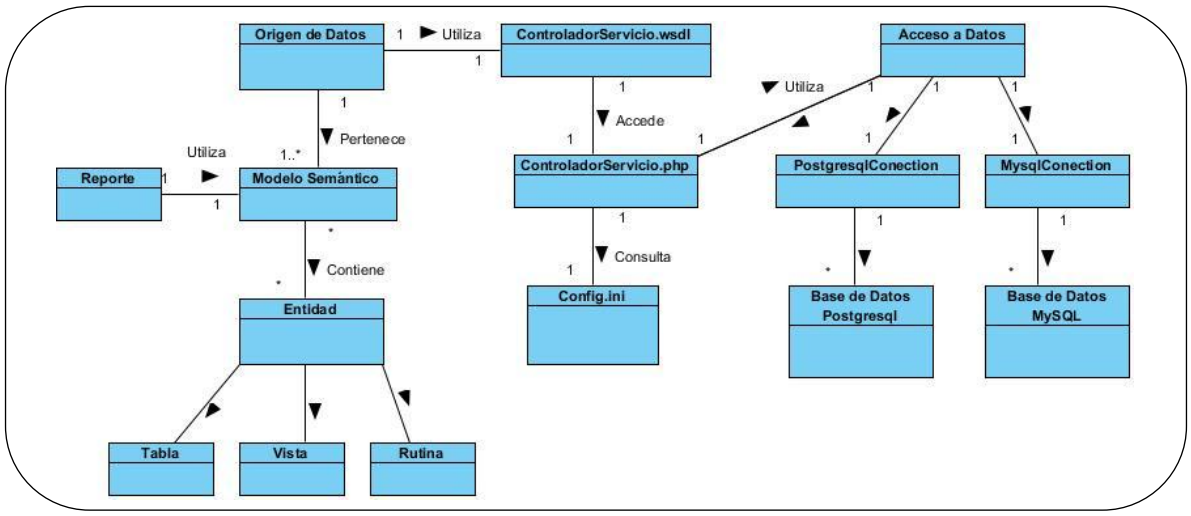


Figura 12. Modelo de Dominio

En este diagrama la clase Reporte, Origen de Datos y Modelo Semántico son recursos del negocio del GDR. La clase Reporte utiliza la información contenida en los modelos semánticos para la confección de los reportes por lo que es imprescindible un modelo semántico para dicha confección. En este contexto, es importante destacar que en el proceso de generación de un modelo se requiere de un origen de datos que lo provea, es decir, que estará asociado a un único origen de datos. Mientras que a partir de un origen de datos se pueden construir varios modelos. Estos últimos contienen los diferentes tipos de entidades: Tabla, Vista y Rutina, que serán utilizadas en el proceso de elaboración de los reportes. Para acceder al origen de datos con alta información sensible se utiliza un servicio web a través de la clase ControladorServicio.wsdl que accede a la clase ControladorServicio.php la cual garantiza una instancia con los parámetros que están en el fichero Config.ini para una mayor seguridad de conexión. Además se tiene la clase de Acceso a Datos como un único punto de acceso para más organización a la hora de conectar las bases de datos, a través de las clases PostgresqlConnection y MysqlConnection que son las encargadas de gestionar toda la información de las base de datos, representadas en este diagrama como Base de Datos PostgreSQL y Base de Datos MySQL.

Descripción de las clases del dominio

Reporte: Utiliza la información contenida en los modelos semánticos para la confección de reportes, los cuales van a ser exportados en formatos como PDF, HTML, EXCEL, CSV.

Origen de Datos: Contiene los datos que permiten conectar al Generador Dinámico de Reportes con los Sistemas Gestores de Bases de Datos. Las variables que maneja son: tipo de gestor de base de datos, dirección IP del servidor, puerto de conexión al servidor, usuario, clave y base de datos.

Modelo Semántico: Es un modelo de datos usado para almacenar en forma XML toda la información de los metadatos de los objetos de la base de datos, formado por entidades tales como: tablas, vistas y rutinas que serán utilizadas en los reportes y previamente cargadas en los orígenes de datos.

Entidad: Representa una clase contenedora de información del negocio, puede ser una tabla, vista o rutina.
Tabla: Tipo de entidad que contiene las tablas de las bases de datos.
Vista: Tipo de entidad que contiene las vistas de las bases de datos.
Rutina: Tipo de entidad que contiene las rutinas de las bases de datos.
ControladorServicio.wsdl: Es la clase que contiene la descripción del servicio web a la cual hará referencia el cliente para acceder a las funcionalidades del servicio.

ControladorServicio.php: Es la clase que garantiza una instancia con los parámetros que están en el fichero Config.ini para una mayor seguridad de conexión.

Config.ini: Este es el fichero que contendrá todos los datos del cliente para conectarse y extraer la información de la base de datos para lograr así una mayor seguridad de conexión.

Acceso a Datos: Es una clase abstracta utilizada como único punto de acceso para más organización de las bases de datos.
PostgresqlConnection: Es la clase que se encarga de gestionar la información en las bases de datos PGSQL.
MysqlConnection.php: Es la clase que se encarga de gestionar la información en las bases de datos MySQL.
Base de Datos PostgreSQL: Esta clase representa el tipo de base de datos con que se va a trabajar que en este caso es Postgresql.
Base de Datos MySQL: Esta clase representa el tipo de base de datos con que se va a trabajar que en este caso es MySQL.

2.2 Requisitos del sistema

Los requisitos son condiciones o capacidades que el sistema debe cumplir, definiendo que es lo que debe hacer. Para esto se identifican las funcionalidades requeridas y las restricciones que se imponen, clasificándose en: funcionales y no funcionales.

2.2.1 Requisitos Funcionales

Los requisitos funcionales (RF) definen el comportamiento interno de un software, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software. El componente de conexión debe cumplir con los requisitos funcionales que a continuación se describen:

- **RF1 Listar las bases de datos**

  Entrada: recibe los parámetros (nombre, servidor, tipo de gestor) para tener acceso a las bases de datos.

  Descripción: el sistema deberá permitir mostrar las bases de datos disponibles.

  Salida: muestra una lista con las bases de datos utilizables.

- **RF2 Adicionar origen de datos**

  Entrada: se especifican los detalles del nuevo origen de datos que se va a adicionar (base de datos, tipo de gestor, identificador del origen de datos, nombre del origen de datos).

  Descripción: se adiciona en el sistema un nuevo origen de datos.

  Salida: muestra una lista con todos los orígenes de datos existentes.

- **RF3 Probar conexión a un origen de datos**

  Entrada: recibe datos necesarios para la conexión (servidor, gestor, base de datos, y se le asigna un nombre al origen de datos).

  Descripción: se prueba establecer la conexión a un origen de datos para verificar que funcione correctamente.

  Salida: mensaje de validación si se estableció bien o no la conexión.

- **RF4 Listar tablas**

Entrada: recibe como parámetro el identificador de la base de datos seleccionada para poder tener acceso a sus tablas.

Descripción: el sistema deberá brindar la funcionalidad de obtener las tablas de la base de datos seleccionada.

Salida: muestra en una lista las tablas de la base de datos seleccionada.

- **RF5 Listar tablas relacionadas**

  Entrada: recibe como parámetro el identificador de la base de datos seleccionada para poder tener acceso a las tablas relacionadas.

  Descripción: El sistema deberá brindar la funcionalidad de obtener las tablas que se relacionan con las seleccionadas.

  Salida: muestra una lista de tablas relacionadas.

- **RF6 Listar atributos de tablas**

  Entrada: recibe como parámetro el identificador de la base de datos seleccionada para poder tener acceso a los atributos de las tablas de esa base de datos.

  Descripción: el sistema deberá permitir obtener los atributos de las tablas.

  Salida: muestra una lista con los atributos de las tablas.

- **RF7 Listar vistas**

  Entrada: recibe como parámetro el identificador de la base de datos seleccionada para poder tener acceso a sus vistas.

  Descripción: el sistema deberá brindar la funcionalidad de obtener las vistas de la base de datos seleccionada.

  Salida: muestra una lista con todas las vistas que posee la base datos seleccionada.

- **RF8 Listar atributos de vistas**

  Entrada: recibe como parámetros el identificador del modelo, el nombre de la vista y el esquema al que pertenece para así obtener los atributos de la vista.

Descripción: el sistema deberá permitir obtener los atributos de las vistas.

Salida: muestra una lista con los atributos de las vistas.

- **RF9 Listar rutinas**

  Entrada: recibe como parámetro el identificador de la base de datos seleccionada para poder tener acceso a sus rutinas.

  Descripción: el sistema deberá brindar la funcionalidad de obtener las rutinas de la base de datos seleccionada.

  Salida: muestra una lista con las rutinas de la base datos seleccionada.

- **RF10 Listar atributos de rutinas**

  Entrada: recibe como parámetros el identificador del modelo, el nombre de la rutina y el esquema al que pertenece para así obtener los atributos de la rutina.

  Descripción: el sistema deberá permitir obtener los atributos de las rutinas.

  Salida: muestra una lista con los atributos de las rutinas.

- **RF11 Exportar reporte**

  Entrada: recibe como parámetros el identificador del reporte, el límite (cantidad de elementos), un arreglo en formato JSON el cual posee un arreglo de valores para cada condición que se desee agregar y un arreglo de parámetros.

  Descripción: se obtiene los datos para la construcción del reporte.

  Salida: se exporta el reporte seleccionado al formato predefinido.

2.2.2 Requisitos No Funcionales

Los requisitos no funcionales especifican criterios que pueden usarse para juzgar las operaciones que realiza un sistema. Constituyen propiedades o cualidades que el producto debe tener, propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

Software

Para el servidor donde se instalará la aplicación se debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4.0 GNU/Linux o superior.
- Para el sistema operativo Windows: wampserver, configurar la variable de entorno y activar las extensiones de php necesarias.
- Paquetes para el sistema operativo GNU/Linux: apache2, php5, libapache2-modphp5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-xsl, php5-gd, php5-soap.
- Extensiones para el sistema operativo Windows: php_intl, php_mbstring, php_mysql, php_mysqli, php_pdo_mssql, php_pdo_mysql, php_pdo_pgsql, php_pdo_pqlite, php_pgsql, php_soap, php-xsl, php_zip.
- Usuario con privilegios de administración.

Para el servidor donde se instalará la base de datos se debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4.
- PostgreSQL versión 8.3 o superior.
- PgAdminIII u otro administrador compatible con PostgreSQL.
- Usuario con privilegios para instalar la base de datos.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP.

Hardware

Para el servidor donde se instalará la aplicación se debe cumplir con los siguientes requisitos:

- Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- Como mínimo requiere 512 MB de memoria RAM.
- Necesita espacio en disco duro igual o superior a 40 GB.

La máquina utilizada para acceder a la aplicación debe cumplir con los siguientes requisitos:

- Procesador Intel Pentium 4 1.7 GHz, o AMD similar.
- 256 MB RAM.

- 20 GB de espacio en disco duro.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos:

- Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- Necesita memoria RAM igual o superior a 512 MB.
- Como mínimo requiere 40 GB de espacio en disco duro.

Restricciones de Diseño e Implementación

- El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.3 o superior.
- Se utilizará el framework de desarrollo Symfony en su versión 1.1.7 y la librería de JavaScript Ext-JS versión 2.2.
- Se empleará la herramienta de desarrollo NetBeans 7.1 y el sistema gestor de base de datos PostgreSQL 9.0.

Interfaz

- La interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma.

Usabilidad

- El sistema debe tener un alto nivel de usabilidad permitiendo que usuarios sin mucha experiencia informática, en aproximadamente 1 mes puedan utilizarlo al 100%.

Eficiencia

- El sistema debe mantener tiempos de respuestas en un marco razonable de diez segundos, permitiendo que existan al menos 10 usuarios conectados de forma simultánea.

2.3 Diagrama de caso de usos del sistema

El diagrama de casos de uso del sistema (DCUS) se utiliza para describir las funcionalidades de un software y documentar su comportamiento. Los casos de uso del sistema (CUS) son operaciones o tareas específicas que se realizan tras una orden de algún agente externo, ya sea desde una petición de un actor o desde la invocación de otro caso de uso. Estos engloban los requisitos funcionales de un sistema, representando las funciones que la aplicación puede ejecutar. A continuación se presenta el DCUS del componente de conexión.

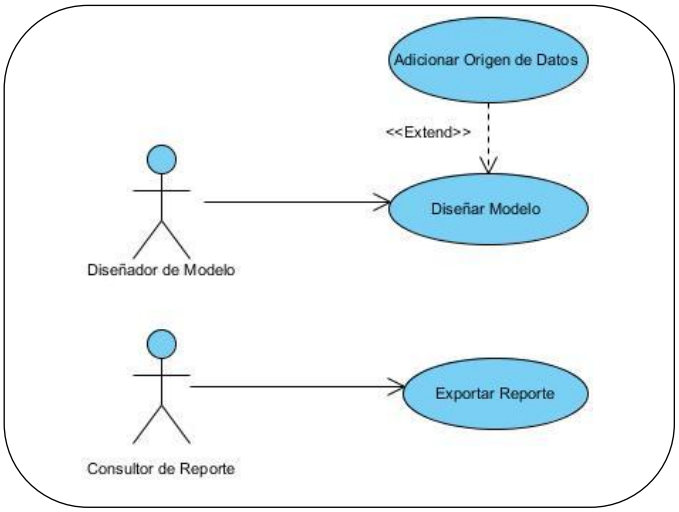


Figura 13. Diagrama de Casos de Uso del Sistema

En este DCUS, el actor Diseñador de Modelos es el encargado de inicializar el caso de uso Diseñar Modelo, considerado crítico por su impacto en la arquitectura del componente. Como función opcional este actor tiene la posibilidad de adicionar orígenes de datos, elementos imprescindibles para el diseño de un modelo semántico. Esta actividad es crítica por el impacto que proporciona en la operatividad del sistema. El actor Consultor de Reporte es el encargado de inicializar el caso de uso Exportar Reporte. En la elaboración de este diagrama se aplicó el patrón de casos de uso “Extensión Concreta” ya que existe una relación extendida entre el caso de uso base Diseñar Modelo y el caso de uso extendido Adicionar Origen de Datos, donde este último no altera el funcionamiento del primero, sino que se incorpora como parte integral del caso de uso base.

Descripción de los casos de uso

Diseñar Modelo: este CUS permite mostrar los modelos y los orígenes de datos ya creados, además de crear nuevos modelos, a partir de un origen de datos seleccionado y buscar un modelo específico en la lista de los modelos previamente creados.

Adicionar Origen de Datos: este CUS permite adicionar un nuevo origen de datos de donde se extrae la información requerida.

Exportar Reporte: en este CUS el actor puede exportar los reportes realizados en diferentes formatos: HTML, PDF, EXCEL, CSV.

2.3.1 Descripción del Caso de Uso Diseñar Modelo.

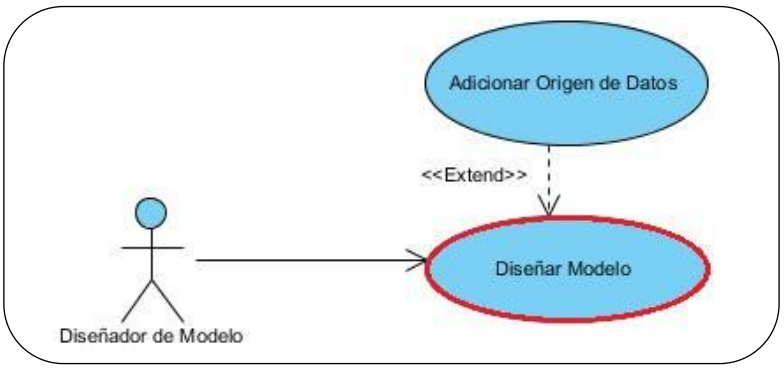


Figura 14. Diagrama de CUS_Diseñar Modelo

Caso de Uso:	Diseñar Modelo.
Actores:	Diseñador de modelo.
Resumen:	El caso muestra los modelos y orígenes de datos que se encuentren creados. Además permite crear nuevos modelos, a partir de un origen de datos seleccionado y buscar un modelo específico en la lista de los modelos previamente creados.
Precondiciones:	1. Tiene que existir al menos un origen de datos creado.
Prioridad	Crítico

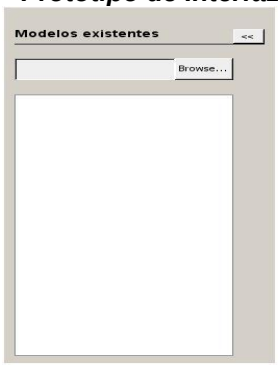
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción *"Diseñador de modelo".*	2. El sistema muestra una interfaz brindándole un listado de modelos, y orígenes de datos previamente creados. Si el actor decide crear un nuevo modelo, ver Sección: "Adicionar modelo".

Prototipo de Interfaz	
	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	Se carga la interfaz de diseñar de modelo.

Sección "Adicionar modelo"	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona un origen de datos.	2. El sistema resalta el origen de datos seleccionado.
3. El actor selecciona la opción "Siguiente".	4. El sistema muestra las entidades pertenecientes al origen de datos, agrupadas por tablas, vistas y rutinas, ver Figura 5: Panel de selección de entidades para asociarlas a un modelo.
5. El actor selecciona las entidades que conformarán el modelo semántico.	6. El sistema habilita la opción "Siguiente". *En caso de que el actor seleccione la opción "Anterior", ver Flujo Alterno 4.1.*
7. El actor selecciona la opción "Siguiente".	8. El sistema muestra los campos asociados a cada entidad previamente seleccionada y

	el botón “Aceptar” para finalizar la construcción del modelo. *En caso de que el actor seleccione la opción “Anterior”, ver Flujo Alterno 8.1.*
9. El actor selecciona la opción “Aceptar”.	10. El sistema muestra una ventana para introducir el nombre del modelo que se desea generar. *En caso de que el actor seleccione la opción “Cancelar”, ver Flujo Alterno 10.1.*
11. El actor introduce un nombre para el nuevo modelo a generar.	12. El sistema habilita la opción “Aceptar”. *En caso de que el actor seleccione la opción “Cancelar”, ver Flujo Alterno 10.1.*
3. El actor selecciona el botón “Aceptar”.	14. El sistema genera el modelo semántico y carga la interfaz con el listado de los orígenes de datos, así como actualiza el listado de modelos semánticos con el nuevo modelo actualizado.

Prototipo de Interfaz



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1. El sistema muestra la lista de orígenes de datos existentes.
	8.1. Retorno al paso 4 del *Flujo Normal de Eventos.*
	10.1. Cierra la ventana donde se debe introducir el nombre del modelo a generar.

Prototipo de Interfaz	
Poscondiciones	Se adiciona a la lista de modelos el modelo creado.

2.4 Descripción del diseño del servicio web

La interfaz ControladorServicio.wsdl perteneciente al paquete "ComponenteConexionServicioWeb", es la encargada de describir el funcionamiento del servicio web implementado, el cual está basado en XML y puntualiza la forma de comunicación. Esta interfaz se desarrolló de forma genérica para que pueda ser utilizada por otros sistemas y la estructura del fichero WSDL se definió de la siguiente manera:

```
<?xml version="1.0" encoding='UTF-8'?>
<definitions>
       <message> ... </message>
       <portType> ... </portType>
       <operation> ... </operation>
       <binding> ... </binding>
</definitions>
```

Donde los elementos del WSDL brindan la información necesaria para interactuar con el servicio web, elementos que se explican a continuación:

- **<?xml version="1.0" encoding='UTF-8'?>:** el documento WSDL es como cualquier documento XML y se basa en los esquemas, por lo que debe comenzar con dicha etiqueta.
- **<definitions>:** con esta etiqueta se define el comienzo del documento y agrupa a todos los demás elementos. Dentro de ella se utilizan atributos como: xmlnsc (espacio de nombre al que pertenece el WSDL, "http://schemas.xmlsoap.org/wsdl/") y name (nombre del propio servicio en el momento de crear el WSDL, "ControladorServicio").
- **<message>:** con esta etiqueta se definen los métodos y parámetros de recepción y entrega de información para realizar las operaciones. Cada message puede consistir en una o más partes (parámetros) y los parámetros se expresan de la siguiente forma, <part name="param1" type="xsd:int">.
- **<portType>:** con este apartado se definen las operaciones permitidas que proporciona el servicio y los mensajes vinculados, (por ejemplo el mensaje de petición y el de respuesta).

- **<operation>:** esta etiqueta es utilizada para indicar si los parámetros son de entrada (input) o de salida (output). La colección de todas las operaciones (métodos) expuestos por el servicio se llama portType y se definen dentro del WSDL con la etiqueta <portType> descrita anteriormente.
- **<binding>:** con esta etiqueta se especifican los protocolos de comunicación usados en el servicio y los formatos de mensaje.
- **<service>:** este elemento define el conjunto de puertos y dirección del servicio. Esta parte final hace referencia a lo aportado por las secciones anteriores y contiene atributos como: name (nombre del servicio) y <port name> (para indicar la dirección y el tipo de acceso del servicio).

Seguidamente se muestra un ejemplo del servicio web implementado con sus diferentes secciones:

```
<message name="listarBaseDatos"></message>
<message name="listarBaseDatosResponse">
    <part name="listarBaseDatosReturn" type="soapenc:Array"></part>
</message>

<message name="listarCamposTabla">
    <part name="database" type="xsd:anyType"></part>
    <part name="table" type="xsd:anyType"></part>
</message>
```

```
<binding name="ControladorServicioBinding" type="typens:ControladorServicioPortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"></soap:binding>
    <operation name="__construct">
        <soap:operation soapAction="urn:ControladorServicioAction"></soap:operation>
        <input>
            <soap:body namespace="urn:ControladorServicio"
            use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"></soap:body>
        </input>
        <output>
            <soap:body namespace="urn:ControladorServicio"
            use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"></soap:body>
        </output>
    </operation>
</binding>
```

2.5 Diagrama de clases del diseño

En la fase de diseño del software se hace un refinamiento del análisis y se centra en cómo van a ser implementados los requerimientos, logrando adaptar el diseño para que se ajuste al entorno de implementación, persiguiendo así el desarrollo de una arquitectura estable y sólida.

El diagrama de clases del diseño (DCD) muestra los atributos y métodos de cada clase, representando de forma sencilla la colaboración y las responsabilidades de cada una de ellas, entorno al sistema que conforman. El siguiente diagrama de clase del diseño contiene las principales clases con sus respectivos métodos y atributos para el caso de uso Diseñar Modelo.

Descripción del diagrama de clases del diseño para el CUS_Diseñar Modelo

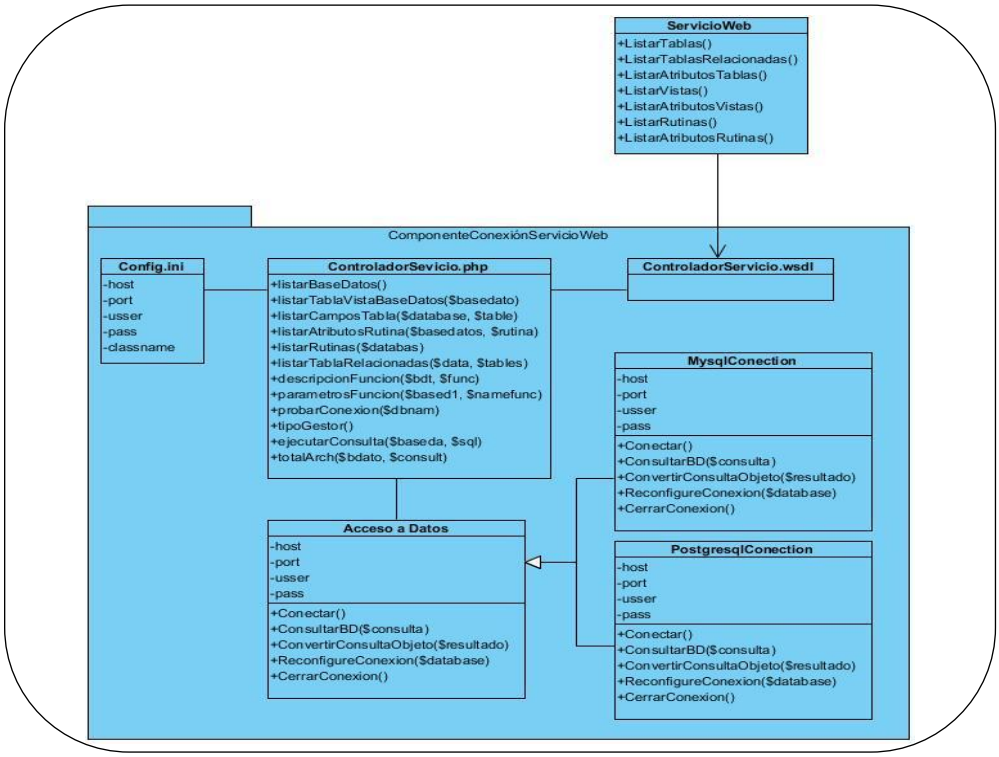


Figura 15. Diagrama de clases del diseño para el CUS_Diseñar Modelo

En este diagrama la clase “Servicio web” contiene las acciones que dan respuestas al caso de uso Diseñar Modelo implementadas en el lenguaje PHP. Dicha clase se relaciona con el paquete “ComponenteConexionServicioWeb” mediante la clase ControladorServicio.wsdl contenida dentro del paquete, este tiene los atributos y métodos a los cuales se hace referencia para acceder a las funcionalidades del servicio. Esta clase se relaciona además con la clase ControladorServicio.php que posee una serie de métodos para garantizar la instancia de los datos del cliente a la hora de conectarse

recogidos en la clase Config.ini y a su vez accede a la clase Acceso a Datos la cual contiene las propiedades y funcionalidades necesarias para gestionar toda la información recogida en las bases de datos que en este caso se trabaja con PostgresqlConnection y MysqlConnection.

Descripción del diagrama de clases del diseño para el CUS_Diseñar Modelo integrado al GDR

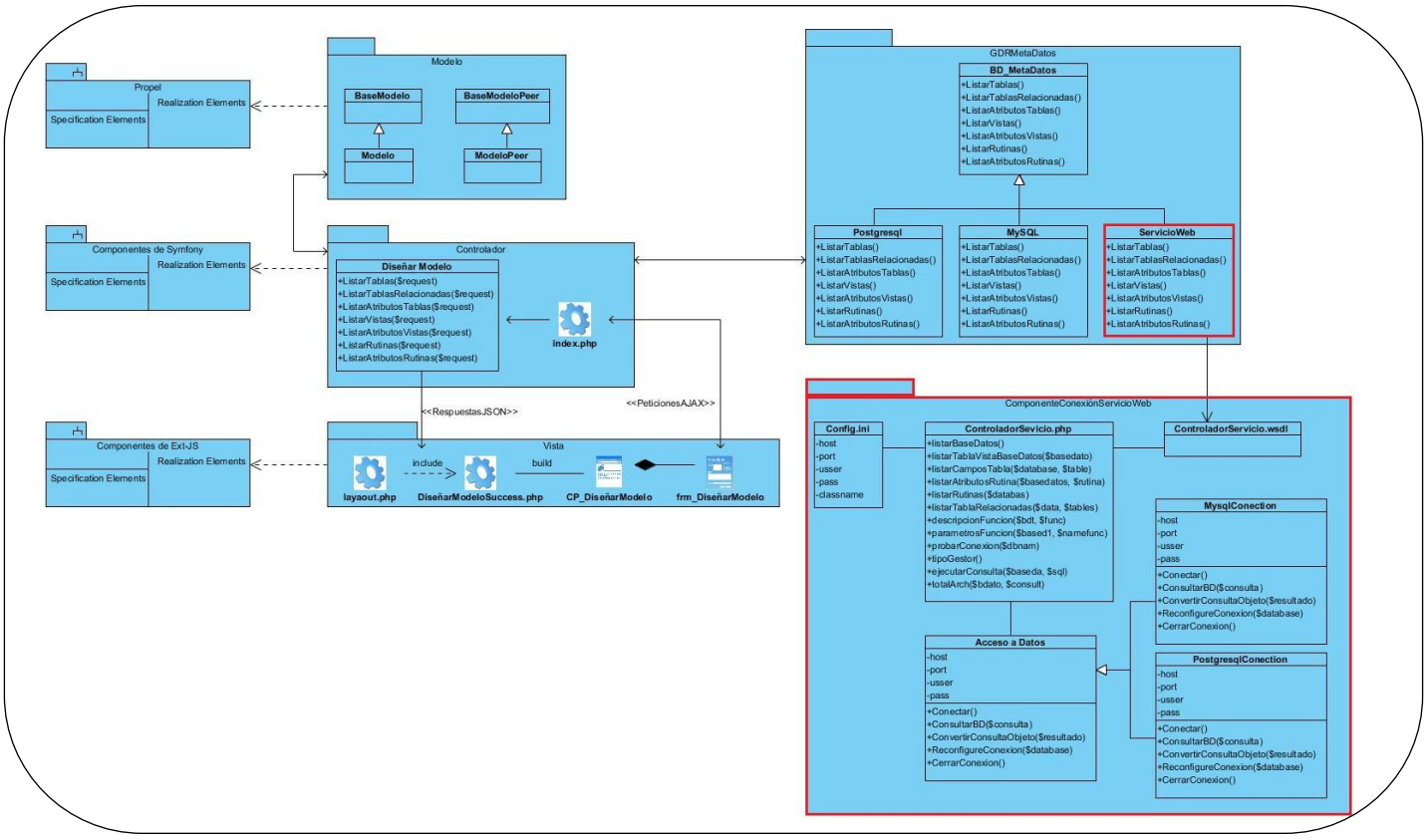


Figura 16. Diagrama de clases del diseño para el CUS_Diseñar Modelo integrado al GDR

En este diagrama se representa la integración del componente con el GDR donde los elementos del paquete “Modelo” corresponden a las clases generadas por la herramienta de mapeo de objeto-relacional Propel modelado por el subsistema de igual nombre. El mismo genera cuatro clases por cada tabla de la base de datos, esto permite refactorizar las funcionalidades sin perder la información contenida hasta ese momento. Las acciones son los elementos que construyen las respuestas del servidor, estas se encuentran agrupadas según el correspondiente caso de uso al que pertenecen e implementadas en el lenguaje PHP, en este caso pertenece al CUS Diseñar Modelo, con la dependencia del subsistema “Componentes de

Symfony" y comprobadas por el controlador frontal "index.php". Se destaca la relación de flujo de información que existe entre el componente caso de uso del cliente y el componente módulo del servidor. Las solicitudes se realizan por medio de la tecnología AJAX y las respuestas son devueltas en formato JSON. Los elementos del lado del cliente corresponden a componentes específicos del negocio como paneles de trabajos asociados al caso de uso y a los componentes genéricos fuertemente reutilizables e implementados en lenguaje JavaScript utilizando los componentes del subsistema "Componentes Ext-JS". El paquete "GDRMetaDatos" contiene todas las funcionalidades que permite obtener la información que se necesita para el funcionamiento del componente de conexión. Para lograr la integración entre el GDR y el componente de conexión se define una clase interfaz en los metadatos para integrar las funciones del componente con la lógica de negocio establecida por el GDR. El componente de conexión está representado por el paquete "ComponenteConexionServicioWeb" el cual contiene las clases con sus respectivas acciones necesarias para realizar la conexión.

2.6 Diagrama de interacción del diseño

Un diagrama de interacción (secuencia o colaboración) explica gráficamente las relaciones existentes entre las instancias y las clases, es decir representa, en respuesta a un determinado evento. En el presente trabajo se utilizan diagramas de secuencia, para organizar los eventos con la sucesión temporal en que se desencadenan. Este tipo de diagrama, muestra los objetos que participan en la interacción mediante las líneas de vida y los mensajes que intercambian. A continuación se representa el escenario Adicionar Modelo perteneciente al caso de uso Diseñar Modelo. (Brito Rodríguez, 2012).

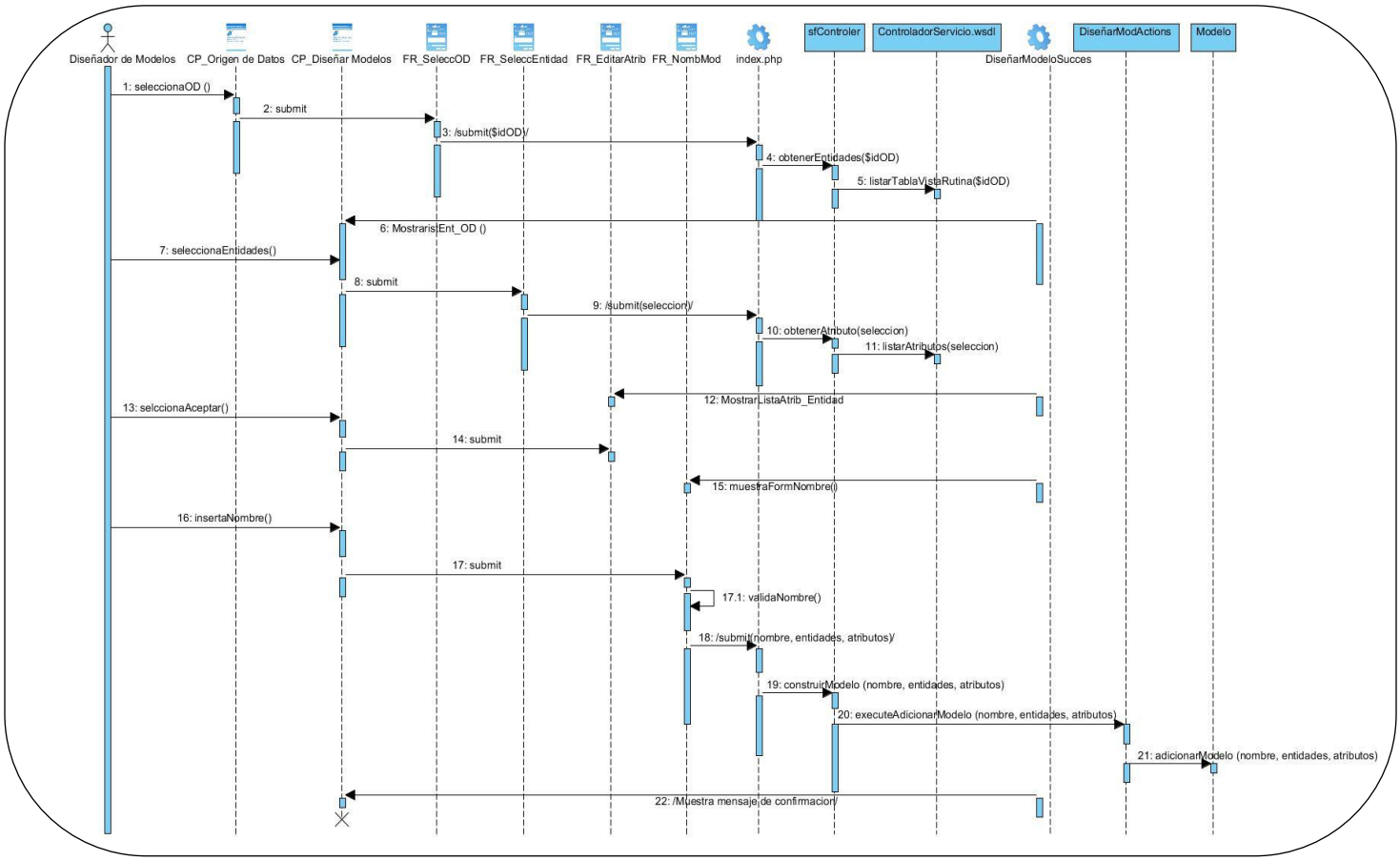


Figura 17. Diagrama de Secuencia del escenario Adicionar Modelo del CUS_Diseñar Modelo.

En este diagrama de secuencia el actor Diseñador de Modelos selecciona un origen de datos en la CP_Origen de Datos el sistema envía el identificador del origen de datos a la clase index.php y con ese identificador se obtienen las tablas, vistas y rutinas, el sistema muestra estas entidades, el diseñador las selecciona, el sistema las envía a la clase index.php, obtiene los atributos de esas entidades y los muestra en el formulario seleccionar atributo, el Diseñador de Modelos selecciona aceptar y el sistema muestra un formulario para introducir el nombre del modelo, el diseñador de modelos introduce el nombre que le dará al modelo y el sistema valida el nombre, envía los datos del modelos, crea el modelo, adiciona el modelo y muestra un mensaje de confirmación.

2.7 Patrones utilizados en la solución

La sociedad requiere sistemas más complejos y más grandes, y los recursos desarrollados son cada vez más escasos, por tanto se hacen imprescindibles los mecanismos de reutilización. Ante la necesidad de

asimilar la reutilización en el desarrollo de software se han popularizado mecanismos como: componentes, framework, objetos distribuidos y patrones de diseño.

Los patrones de diseño de software son soluciones reutilizables de problemas recurrentes que aparecen durante el proceso de diseño de software orientado a objetos. Estos estandarizan buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

A continuación se describen algunos de los patrones de diseño utilizados en la solución planteada.

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En la solución del sistema a través de este patrón se accede, en la clase "ControladorServicio.php" que incluye a la clase "Singleton.php", a la instancia del contexto mediante el método getInstance(), que permite acceder a cualquier clase de conexión.

```
class Singleton {

    private static $instancia = null;

    public function __construct() {

    }
    public static function getInstance() {

        if (self::$instancia == null) {
            $Reader = new ConfigReader('config.ini');

            $config = $Reader->getConfig();
            $config = $config['config'];

            self::$instancia = new $config['clase']($config['host'], $config['port'],

                    $config['user'], $config['pass']);
        }
        return self::$instancia;
    }
```

Figura 18. Patrón Singleton

DAO (Objeto de Acceso a Datos): Este patrón permite contar con diversas fuentes de datos (base de datos, archivos, servicios externos). De tal modo que se encapsula la forma de acceder a la fuente de datos, así garantiza gestionar una diversidad de fuentes de datos y ocultar la forma de acceder a ellos. En la solución se utilizó en la clase "Acceso a Datos" como un adaptador entre la interfaz del servicio web y las fuentes de datos.

```php
<?php

abstract class Conexion {

    protected $host;
    protected $puerto;
    protected $usuario;
    protected $contrasena;
    protected $conexion;

    public function __construct($host, $port, $user, $pass) {
        $this->host = $host;
        $this->puerto = $port;
        $this->usuario = $user;
        $this->contrasena = $pass;
        $this->conexion = null;
    }
    public function dameconexion() {
        return $this->conexion;
    }
    public abstract function Conectar();
    public abstract function ConsultarBD($consulta);
    public abstract function ConvertirConsultaObjeto($resultado);
    public abstract function ReconfigureConexion($database);
    public abstract function CerraConexion();
    public abstract function listarBaseDatos();
    public abstract function listarTablaVistasBaseDatos();
    public abstract function listarCamposTabla($tableName);
    public abstract function listarTablaRelacionadas($tables);
    public abstract function listarRutinas();
    public abstract function listarAtributosRutina($functionName);
```

Figura 19. Patrón DAO

Controlador Frontal: Es un patrón muy utilizado por las aplicaciones web. Describe básicamente como construir un sólo punto de acceso para todas las peticiones de una aplicación web. Escucha las peticiones que vienen desde una URL, y se encarga de llamar al controlador específico, posteriormente llama a la acción deseada del controlador. En la solución del sistema se aplica en la creación de un único punto de acceso para todas las peticiones realizadas al módulo, en este caso el “index.php”.

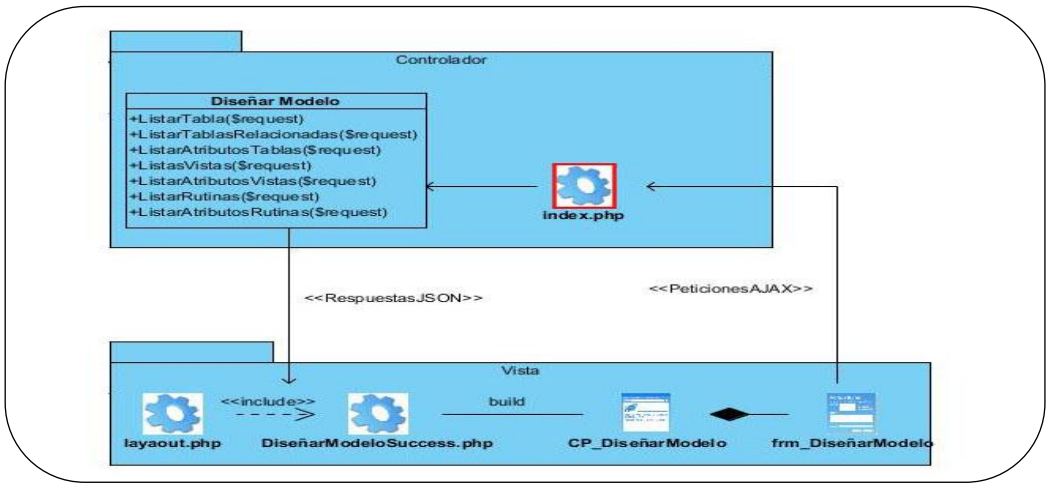


Figura 20. Patrón Controlador Frontal

2.8 Diagrama de Despliegue

Los diagramas de despliegue muestran las relaciones físicas entre los componentes de hardware y software en el sistema final. Están compuestos por nodos, dispositivos y conectores, donde los nodos son elementos de procesamiento, los dispositivos son nodos estereotipados sin capacidad de procesamiento y los conectores expresan el tipo de conector utilizado entre el resto de los elementos del modelo.

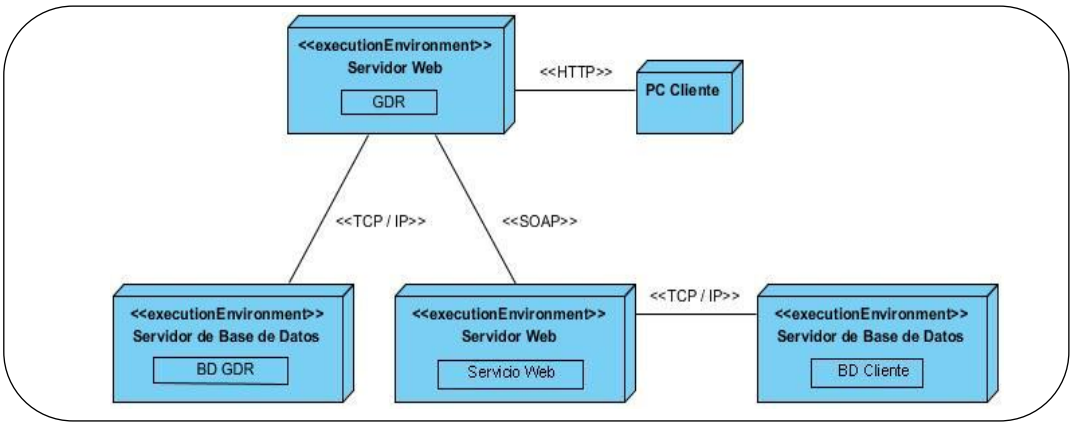


Figura 21. Diagrama de Despliegue

En la solución el diagrama de despliegue se distribuye por una estación de trabajo para el cliente desde la cual se realizarán peticiones por el protocolo HTTP al servidor de aplicaciones web representado por el nodo “Servidor Web”. El cual accederá a los datos de la aplicación almacenada en un Servidor de Base de Datos mediante el protocolo TCP/IP. Para insertar el componente de conexión y que pueda ser utilizado por el cliente se dispone de otro nodo “Servidor Web” donde va a estar el servicio web y la conexión hacia este nodo es mediante el protocolo SOAP. Para acceder a la información del cliente se tiene un Servidor de Base de Datos que contiene los datos utilizados en el diseño de los reportes; la comunicación entre ambos procede de igual forma por el protocolo TCP/IP.

2.9 Conclusiones Parciales

En el desarrollo del presente capítulo se realizó el modelo de dominio donde se describieron las clases conceptuales que ayudarán a comprender los conceptos claves para un mejor dominio del problema. Además se identificaron 11 requisitos funcionales agrupados en 3 casos de uso y 6 requisitos no funcionales que el sistema debe cumplir, basados en software, hardware, interfaz, eficiencia, usabilidad y restricciones de diseño e implementación. También se realizó el diagrama de caso de uso del sistema, donde se definieron 3 casos de uso y el caso de uso Diseñar Modelo como el más crítico por su impacto en

la arquitectura del componente. De igual manera, los diagramas de clases del diseño para mostrar las propiedades y funcionalidades de las 6 clases que conforman el componente de conexión y la relación entre ellas y los diagramas de secuencia que se utilizaron para inspeccionar los aspectos dinámicos del componente. Se modeló el diagrama de despliegue teniendo en cuenta que el sistema es un componente que se utilizará en la solución del GDR y asume sus características y requerimientos tecnológicos, así como la incorporación del servicio web.

3

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL COMPONENTE DE CONEXIÓN

Introducción

Partiendo del resultado del diseño, en el presente capítulo se generan los artefactos referentes a la fase de implementación y prueba del software. Se modela el sistema en términos de componentes y se dan a conocer las técnicas usadas en la solución del problema. También se especifican los casos de pruebas realizadas al componente para validar su correcto funcionamiento.

3.1 Implementación

Esta disciplina explica cómo desarrollar, organizar, realizar pruebas de unidad e integrar los componentes implementados basándose en las especificaciones del diseño. Tiene la finalidad de definir la organización del código, en términos de los subsistemas de implementación, organizados en capas. Además se implementan los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables) y permite probar y desarrollar componentes como unidades, así como integrar los resultados producidos por los implementadores individuales, o equipos en un sistema ejecutable. (Corp, 2006).

El modelo de implementación está compuesto por el diagrama de componentes, artefacto generado en este flujo de trabajo, al describir los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

3.1.1 Diagrama de Componentes

El diagrama de componentes representa cómo un sistema de software se estructura en componentes, señalando la organización y mostrando las dependencias que existen entre ellos; los componentes físicos incluyen: archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables y paquetes. Este diagrama prevalece en el campo de la arquitectura de software, pero puede ser usado para modelar y documentar cualquier arquitectura de sistema; éste es utilizado para modelar la vista estática y dinámica de un sistema. Se pueden agrupar en paquetes, y entre ellos pueden existir relaciones de dependencia como: generalización, asociación, agregación o realización. Además contienen las interfaces que constituyen la

unión entre varios componentes. Los componentes pueden agruparse en paquetes y pueden contener subsistemas. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se modelan por partes; cada diagrama describe un apartado del sistema. En él se sitúan librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Además contienen las interfaces que constituyen la unión entre varios componentes. Uno de los usos principales es que pueden servir para mostrar qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

Diagrama de Componentes del CUS_Diseñar Modelo

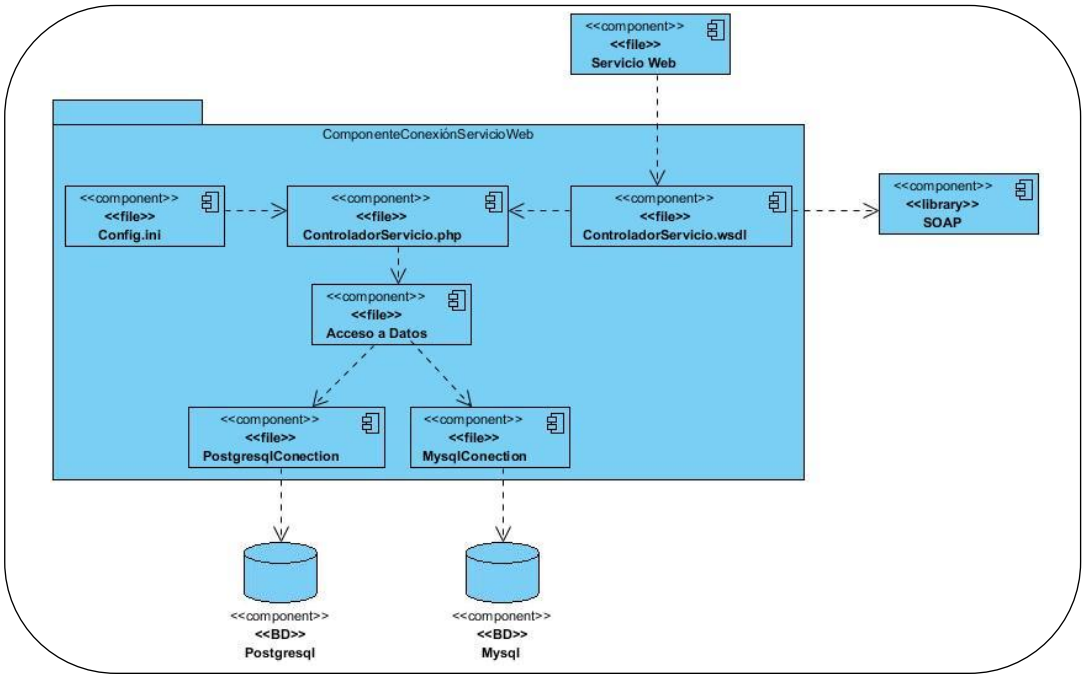


Figura 22. Diagrama de componentes del CUS_Diseñar Modelo

En este diagrama, en el componente “Servicio web” se agrupan todas las acciones que dan respuestas al caso de uso Diseñar Modelo. Los componentes del paquete “ComponenteConexiónServicioWeb” intervienen también en el caso de uso, cada una de estas con su respectivo fichero PHP y su conjunto de acciones que posibilitan el funcionamiento del componente. En el componente “SOAP” se encuentra la librería utilizada para el desarrollo del servicio web y los componentes PostgreSQL y MySQL representan las bases de datos utilizadas.

3.1.2 Implementación del servicio web

El servicio web cuenta con un total de catorce operaciones basadas en las operaciones de “petición – respuesta” que son las operaciones más frecuentes, las cuales se describen a continuación para un mejor entendimiento.

1. **listarBaseDatos** esta operación no recibe ningún parámetro y devuelve el listado de las bases de datos que existen en el servidor.
2. **listarTablaVistaBaseDatos** esta operación recibe por parámetro el nombre de la base de datos de la cual se quieren listar las tablas y las vistas y devuelve un arreglo con las tablas y las vistas de la base de datos.
3. **listarCamposTabla** esta operación recibe dos parámetros: el primero será el nombre de la base de datos y el segundo el nombre de una tabla dentro de la base de datos, arrojando como resultado un arreglo con los campos de la tabla especificada.
4. **listarAtributosRutina** esta operación recibe como parámetros el nombre de la base de datos y el nombre de la rutina de la cual se pretende listar sus atributos, devolviendo un arreglo con cada uno de ellos.
5. **listarRutinas** esta operación recibe por parámetro el nombre de la base de datos y retorna un arreglo con las rutinas de la base de datos.
6. **listarTablaRelacionadas** esta operación recibe dos parámetros: el primero es el nombre de la base de datos y el segundo es un arreglo de las tablas a las que se le buscarán las demás tablas que se relacionan con ellas, como resultado se obtiene un arreglo con todas las tablas relacionadas.
7. **descripcionFuncion** esta operación recibe dos parámetros: la base de datos y el nombre de la función, devuelve la descripción de la función.
8. **llavesForaneasTabla** esta función recibe por parámetro la base de datos y el nombre de la tabla de la cual se desea conocer las llaves foráneas, devuelve un arreglo con las llaves foráneas.
9. **parametrosFuncion** esta operación recibe dos parámetros: la base de datos y el nombre de la función y retorna un arreglo con los parámetros de la función.
10. **probarConexion** esta operación recibe por parámetro la base de datos a la cual se quiere probar si existe conexión o no y retorna un verdadero o falso.

11. **tipoGestor** esta operación no recibe parámetros y devuelve un string con el nombre del gestor al que se está accediendo.
12. **ejecutarConsulta** esta operación recibe dos parámetros: la base de datos y la consulta que se quiere ejecutar en la base de datos y retorna un arreglo con el resultado de la consulta.
13. **totalArch** esta operación recibe por parámetro la base de datos y una consulta y retorna un string con la cantidad de tuplas que devuelve la consulta.
14. **cargarDatos** esta operación recibe tres parámetros: la base de datos, los campos que se desean y el nombre de una tabla de esa base de datos, retorna un arreglo con los datos detallados.

Todas las operaciones garantizan el correcto funcionamiento del Generador Dinámico de Reportes una vez incorporado el uso del servicio web como fuente de datos.

3.2 Estándares de codificación

Los estándares de codificación comprenden todos los aspectos de la generación de código, son reglas que se siguen para la escritura del código fuente. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. La aplicación de un estándar de codificación aporta características siempre deseadas durante la implementación del software ya que eleva la claridad del código, sirve como punto de referencia para los programadores, mantiene un estilo de programación y ayuda a mejorar el proceso de codificación, haciéndolo más eficiente. (Brito Rodríguez, 2012).

Para el desarrollo del componente se definieron varios estándares de codificación los cuales se explican a continuación:

Nombre de identificadores

Se considera como identificador a los nombres de variables, funciones, así como cualquier tipo de dato definido por el usuario (estructura, clase). Dichos identificadores se escribirán en estructura camelCase. En la siguiente figura #24 se evidencia el uso de dicho estándar.

Comentarios

Es conveniente dejar información que pueda ser leída tiempo después por personas que necesitan entender que fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros. Generalmente deben usarse comentarios de una sola línea. En la siguiente figura #24 se evidencia el uso de dicho estándar.

```php
/**
 * este metodo lista las bases de datos de acuerdo al gestor
 * @return array
 */
function listarBaseDatos() {

    $instance = Singleton::getInstance();
    return $instance->listarBaseDatos();
}

/**
 * este metodo lista las tablas de una bases de datos de acuerdo al gesto
 * @param string $basedato
 * @return array
 */
function listarTablaVistaBaseDatos($basedato) {
    Singleton::ReconfigureConexion($basedato);
    $instance = Singleton::getInstance();
    return $instance->listarTablaVistasBaseDatos();
}
```

Figura 23. Estándar Nombre de identificadores

Declaraciones de variables por líneas

Cada variable debe de ser declarada en una línea y el nombre debe de comenzar con letras minúsculas, como se ve en la figura siguiente:

```php
protected $host;
protected $puerto;
protected $usuario;
protected $contrasena;
protected $conexion;
```

Figura 24. Estándar Declaraciones de variables por líneas

3.3 Pruebas del software

Las pruebas del software son el proceso que permite verificar, garantizar y mostrar la calidad de un producto, a través de resultados registrables que proporcionan una evaluación y representa una revisión final de las especificaciones, del diseño y de la codificación. Son empleadas para identificar posibles fallos

durante el proceso de desarrollo. Los casos de prueba son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, permitiendo encontrar y documentar los defectos que puedan afectar la calidad del software. (Brito Rodríguez, 2012).

3.3.1 Estrategia de Prueba

Una estrategia de prueba describe y verifica el enfoque de la misma. Tiene como función fundamental demostrar que diferentes técnicas facilitan la prueba planificada, así como definir las mismas (manual o automática), verificando que el enfoque trabajará, producirá resultados precisos y es apropiado para los recursos disponibles. Además incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada y los casos de prueba diseñados para lograr los objetivos. Delimita los criterios de éxitos y culminación de las pruebas y define consideraciones especiales relacionadas con los recursos necesarios para realizar esta tarea. (Brito Rodríguez, 2012).

Nivel de Prueba

Los niveles de prueba especifican diferentes ángulos para verificar y validar un producto de software. Existen varios niveles de pruebas como: pruebas de desarrollador, independiente, unidad, integración, sistema y aceptación, donde cada nivel contiene una técnica de prueba específica según los atributos de calidad que se deseen verificar. En el desarrollo del componente se aplicarán solamente las pruebas siguientes para comprobar que el sistema da respuesta a los requisitos funcionales definidos anteriormente:

- Nivel de Desarrollador: es el primer nivel de pruebas, diseñadas e implementadas por el equipo de desarrollo, donde el programador es quien revisa su código fuente.
- Nivel de Sistema: en este nivel las pruebas tienen como propósito ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema (hardware, software) y que realizan las funciones adecuadas.
- Nivel de Integración: en este nivel se prueba los componentes combinados para ejecutar un CUS. Además se realiza la prueba para descubrir errores en las especificaciones de las interfaces de las clases.

Tipo de Prueba

Existen diferentes tipos de pruebas que se pueden aplicar para verificar que el componente de conexión cumple con todos los requisitos identificados en el proceso de análisis y comprobar su correcto funcionamiento. A continuación se explican los tipos de pruebas seleccionados:

Pruebas Funcionales: son aquellas que tienen por objetivo demostrar que los sistemas desarrollados, cumplen con las funciones específicas para los cuales han sido creados incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Es común que sea desarrollada por los analistas de pruebas con apoyo de algunos usuarios finales y están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.

Pruebas de Rendimiento (Carga y Estrés): están diseñadas para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. Se realizan basándose en la funcionalidad del sistema bajo cargas pesadas, un gran número de repeticiones, manejo de grandes datos y demasiadas preguntas a bases de datos grandes.

Pruebas de Integración: son un conjunto de pruebas unitarias, funcionales, regresión y aceptación que se realizan para probar el software. Incluye también comprobar que lo programado por los diferentes desarrolladores funciona en un entorno real.

Método de Prueba

Las pruebas funcionales utilizan el método de Caja Negra, que se centra en los requerimientos funcionales del software, o sea, permite demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto.

Este método intenta encontrar errores de las siguientes categorías:

- Errores de interfaz.
- Errores en estructura de datos o en acceso a base de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

La ventaja fundamental del método de Caja Negra es que permite identificar claramente las entradas y salidas además de estudiar las relaciones que existen entre ellas. Además maximiza la eficiencia de los sistemas sin tener que introducirnos en los todos los procesos del software, exceptuando cuando se presentan anomalías en las relaciones de entrada y salida, entonces nos vemos obligados a estudiar ese subsistema de forma más precisa.

3.4 Diseño de Caso de Prueba

Un caso de prueba se diseña a partir de las funcionalidades descritas en los casos de usos y se realizan con el objetivo de comprobar que las funcionalidades han sido implementadas según las peticiones del cliente. Para cada caso de uso debe haber una planilla de caso de prueba donde se recoge la especificación del caso de uso, dividido en secciones y escenarios detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión. A continuación se presentan las tablas de la sección probada para el caso de uso Diseñar Modelo.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1 diseñarModeloAction	SC 1.1: Seleccionar origen de datos.	El diseñador de modelos decide seleccionar un origen de datos y el sistema muestra todas las entidades que existen en el origen de datos.
	SC 1.2: Seleccionar entidades.	El diseñador de modelos escoge las entidades que desea para el reporte y el sistema muestra todos los atributos de las entidades seleccionadas.
	SC 1.3: Editar atributos	El diseñador puede modificar los atributos de las entidades seleccionadas y el sistema muestra un formulario para el nombre del modelo.
	SC 1.4: Nombrar el modelo	El diseñador de modelos introduce el nombre del modelo y el sistema crea y muestra el modelo en la lista de modelos.

Tabla 5. Sección de prueba para el CUS_Diseñar Modelo

A partir de esta descripción se detallan las variables que se encuentran asociadas al caso de uso.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción

1	Origen de datos	Campo de selección	No	Constituye el origen de datos a seleccionar para crear el modelo.
2	Entidad	Campo de selección	No	Representa las entidades del origen de datos (Tablas, Rutinas y Vistas), las cuales deben se seleccionadas para crear el modelo.
3	Alias	Campo de texto	No	Es el alias que va a tener cada Tabla, Rutina o Vista.
4	Buscar entidad	Campo de texto	No	Se escribe el nombre de la entidad que se desea buscar.
5	Nombre	Campo de texto	No	Especifica el nombre que va a tener el modelo creado.

Tabla 6. Descripción de las variables

Esta descripción facilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se registraron los resultados de las pruebas, con el empleo de la técnica de partición de equivalencia. La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.

Matriz de datos

Escenario Seleccionar origen de datos

Escenario	Variables (Enumeradas según descripción de la variable)					Descripción	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	1	2	3	4	5				
EC 1.1Seleccionar origen de datos correctamente.	V	/	/	/	/	En este escenario se selecciona un origen de datos con todos los campos correctos.	El sistema muestra todas las entidades que existen en el origen de datos y habilita la opción “Siguiente”.	Satisfactorio.	1-Diseñador de Modelos y selecciona un origen de datos sobre el cual se requiera crear el modelo y selecciona la opción “Siguiente”.
EC 1.2Seleccionara origen de datos incorrectamente.	I	***/***	***/***	***/***	/	En este escenario se selecciona un origen de datos incorrectamente.	El sistema no habilita la opción “Siguiente” para continuar.	Satisfactorio.	1-Diseñador de Modelos y selecciona incorrectamente un origen de datos sobre el cual se requiera crear el modelo y no puede seleccionar la opción ”Siguiente”.

3.5 Resultados de las pruebas

Caja Negra

Una vez realizadas las pruebas de Caja Negra a través de los casos de prueba asociados a cada CUS, se demostró el correcto funcionamiento del componente, así como la correcta validación de los campos comprobando que sean aceptados todos los caracteres válidos que les corresponde a cada uno. En el proceso de pruebas se detectaron 4 No Conformidades, las cuales fueron corregidas y gestionadas correctamente en cada iteración.

Prueba de rendimiento (carga y estrés)

Para observar el comportamiento de la aplicación bajo una cantidad de peticiones esperadas se realizó las pruebas de rendimiento (carga y estrés) y para ejecutar estas pruebas se empleó la herramienta JMeter donde se logró comprobar la velocidad con la que el sistema ejecuta una tarea. Esta prueba pudo mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación realizadas por un número de usuarios concurrentes haciendo peticiones al mismo tiempo.

En la siguiente figura se muestra un ejemplo de la prueba realizada en la aplicación JMeter a la herramienta donde se obtuvo como resultado: para una muestra de 10 usuarios concurrentes conectados a la herramienta un tiempo de respuesta de 4.8 segundos, que comparado con el tiempo promedio de respuesta, definido en el requisito no funcional de Eficiencia, es menor por lo que se cumple el requisito descrito. Además se puede apreciar que no hubo ningún error en el resultado de las peticiones por HTTP.

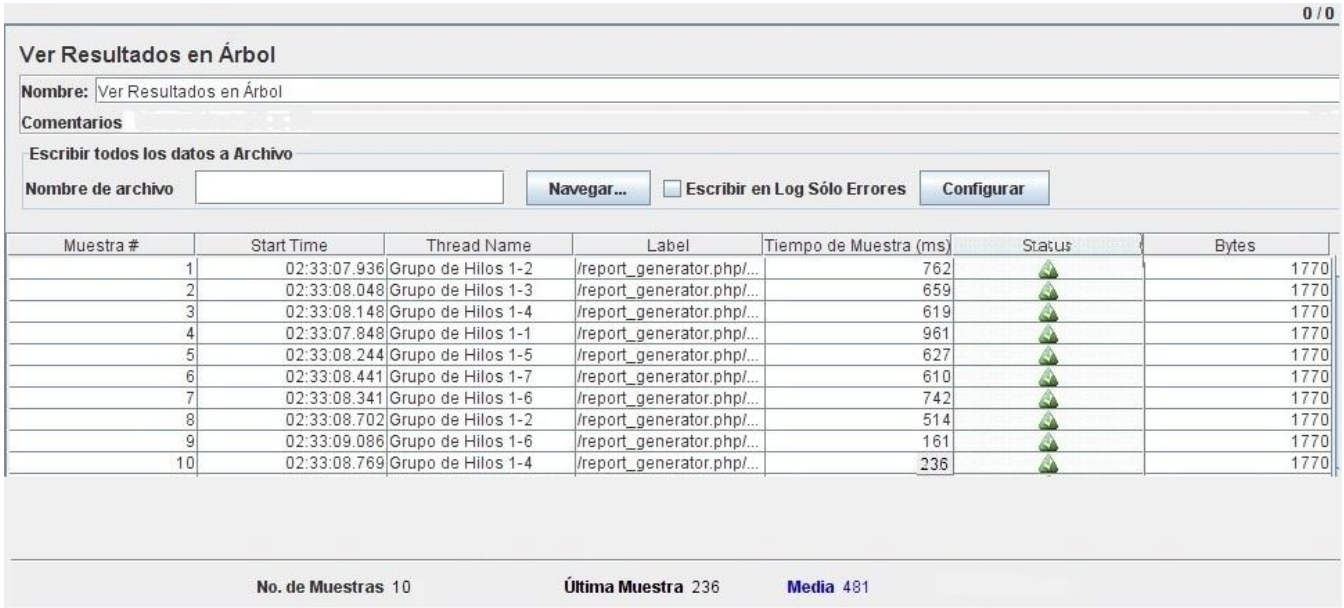


Figura 25. Resultado de la prueba de rendimiento

Prueba de integración

Una vez integrado el componente de conexión al Generador Dinámico de Reportes, se pudo apreciar que funciona correctamente. Cuando el usuario va a diseñar un modelo y adiciona un nuevo origen de datos, el sistema permite seleccionar dentro de los tipos de gestores la opción “web service”, que es la encargada de llamar el servicio web implementado, el cual tendrá la función de obtener toda la información de las bases de datos y devolverlas al sistema de forma tal que continúe el flujo normal de actividades. Además el

usuario después de haber adicionado el origen de datos puede realizar todas las operaciones siguientes como listar las bases de datos, obtener las entidades de las bases de datos, obtener los campos de las entidades hasta lograr tener todos los parámetros necesarios para crear el modelo. Una vez creado el modelo el usuario puede acceder al módulo Diseñador de Reportes para crear el reporte y seguidamente pasar al módulo Visor de Reportes para exportar dicho reporte usando como fuente de datos la información adquirida mediante el servicio web.

3.6 Conclusiones Parciales

En el desarrollo del presente capítulo se realizó la descripción de la implementación del componente de conexión, y se representó la estructura del mismo en el diagrama de componentes, señalando la organización y las dependencias que existen entre ellos. También se definieron los estándares de codificación que fueron utilizados, destacando el uso del estándar camelCase y la utilización de los comentarios para explicar el flujo del código y el propósito de las funciones o variables implementadas. Además se definieron como niveles de pruebas usados el nivel de desarrollador, el de sistema y el de integración, utilizando las pruebas funcionales, de rendimiento con carga y estrés y de integración, empleando el método de caja negra y utilizando la técnica de partición de equivalencia para asegurar la calidad del componente de conexión implementado. Una vez desarrolladas estas pruebas fueron resueltas todas las no conformidades detectadas en cada iteración.

CONCLUSIONES

Culminando el desarrollo de la presente solución se arriba a las siguientes conclusiones:

- Se definió el análisis y el diseño del componente de conexión para el Generador Dinámico de Reportes.
- Se realizó la implementación del componente propuesto, para lo cual se modeló el sistema en términos de componentes.
- Se aplicaron pruebas funcionales de tipo caja negra, descritas en los casos de pruebas realizados, validando con estas el correcto funcionamiento del componente de conexión para el Generador Dinámico de Reportes, además de pruebas de rendimiento a través de carga y estrés y pruebas de integración.
- Se obtuvo como resultado un mecanismo de comunicación entre el GDR y los orígenes de datos PostgreSQL y MySQL, que permite obtener la información de forma indirecta, utilizando las potencialidades de los servicios web.
- Se facilita la descripción de la interfaz para la integración a GDR; logrando una conexión que puede ser utilizada por otros sistemas mediante un servicio web implementado de forma genérica.

RECOMENDACIONES

Al concluir este trabajo se recomienda:

- Utilizar el componente de conexión empleando el servicio web en otras herramientas del departamento.
- Extender la gestión de la conexión del servicio web para los gestores SQLite y Oracle.

REFERENCIAS BIBLIOGRÁFICAS

1. **Brito Rodríguez, Julio César. 2012.** *Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.* La Habana : s.n., 2012.
2. **cab. 2012.** [En línea] 2012. http://www.cab.de/es/marcaje/software-de-etiquetas/database-connector/.
3. **Corp, IBM. 2006.** *Rational Software Architect.* 2006.
4. **devTroce. 2010.** [En línea] 4 de 01 de 2010. Crear Reportes con Visual Basic .Net, PostgreSQL y Crystal Report _ DevTroce.com.htm.
5. **elguille.** [En línea] [Citado el: 4 de abril de 2013.] http://www.elguille.info/colabora/NET2005/giovannyfernandez_EstandarCodificacionNET.htm..
6. **Gazquez Martínez, Orelvi y Serrano García, Yadrian. 2011.** *Diseño e Implementación del módulo Diseñador de Consultas del Generador Dinámico de Reportes.* La Habana : s.n., 2011.
7. **Henández Carvajal, Juan José. 2012.** *Desarrollo del módulo de Administrador de Reportes del Generador Dinámico de Reportes.* La Habana : s.n., 2012.
8. **Hernández Hernández, Yasmany. 2009.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* La Habana : s.n., 2009.
9. **Herrera , Cristhian. 2005.** [En línea] 29 de 04 de 2005. [Citado el: 28 de 11 de 2012.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=iReport.
10. **Larman, Craig. 1999.** *UML y Patrones.* México : s.n., 1999.
11. **Lobo, Armando Robert. 2009.** *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* La Habana : s.n., 2009.
12. **Navarro Rodríguez, Ana Niuska. 2011.** *Asistente de Reportes para el módulo Diseñador de Reportes del Generador Dinámico de Reportes versión 2.0.* La Habana : s.n., 2011.
13. **O´Really XML. 2010.** [En línea] 2010. [Citado el: 5 de 12 de 2012.] http://www.xml.com.
14. **Parte IV_Fase de diseño 1**. *UML.*
15. **Pgadmin, Introduction. 2011.** [En línea] 2011. http://www.pgadmin.org.

16. **Rodríguez Durán, Aurelio. 2011.** *IMPACTO DEL GENERADOR DINÁMICO DE REPORTES EN LOS SISTEMAS DE GESTIÓN DE INFORMACIÓN.* La Habana : s.n., 2011.

17. **Sánchez Góngora, Marcel. 2009.** *Extensión de la capa de servicio web del Gestor de Contenido.* La Habana : s.n., 2009.

18. **2011.** Visual Paradigm for UML. [En línea] 2011. [Citado el: 6 de 12 de 2012.] http://www.visual-paradigm.com/product/vpuml/.

19. **Word Wide Web. 2008.** [En línea] 09 de 01 de 2008. [Citado el: 25 de 11 de 2012.] http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

20. **Yanes León, Vania Elena.** *Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2011.

BIBLIOGRAFÍA

Brito Rodríguez, Julio César. 2012. *Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.* La Habana : s.n., 2012.

cab. 2012. [En línea] 2012. http://www.cab.de/es/marcaje/software-de-etiquetas/database-connector/.

Corp, IBM. 2006. *Rational Software Architect.* 2006.

devTroce. 2010. [En línea] 4 de 01 de 2010. Crear Reportes con Visual Basic .Net, PostgreSQL y Crystal Report _ DevTroce.com.htm.

elguille. [En línea] [Citado el: 4 de abril de 2013.] http://www.elguille.info/colabora/NET2005/giovannyfernandez_EstandarCodificacionNET.htm..

Gazquez Martínez, Orelvi y Serrano García, Yadrian. 2011. *Diseño e Implementación del módulo Diseñador de Consultas del Generador Dinámico de Reportes.* La Habana : s.n., 2011.

Henández Carvajal, Juan José. 2012. *Desarrollo del módulo de Administrador de Reportes del Generador Dinámico de Reportes.* La Habana : s.n., 2012.

Hernández Hernández, Yasmany. 2009. *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* La Habana : s.n., 2009.

Herrera , Cristhian. 2005. [En línea] 29 de 04 de 2005. [Citado el: 28 de 11 de 2012.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=iReport.

Larman, Craig. 1999. *UML y Patrones.* México : s.n., 1999.

Lobo, Armando Robert. 2009. *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* La Habana : s.n., 2009.

Navarro Rodríguez, Ana Niuska. 2011. *Asistente de Reportes para el módulo Diseñador de Reportes del Generador Dinámico de Reportes versión 2.0.* La Habana : s.n., 2011.

O´Really XML. 2010. [En línea] 2010. [Citado el: 5 de 12 de 2012.] http://www.xml.com.

Parte IV_Fase de diseño 1. *UML.*

Pgadmin, Introduction. 2011. [En línea] 2011. http://www.pgadmin.org.

Rodríguez Durán, Aurelio. 2011. *IMPACTO DEL GENERADOR DINÁMICO DE REPORTES EN LOS SISTEMAS DE GESTIÓN DE INFORMACIÓN.* La Habana : s.n., 2011.

Sánchez Góngora, Marcel. 2009. *Extensión de la capa de servicio web del Gestor de Contenido.* La Habana : s.n., 2009.

2011. Visual Paradigm for UML. [En línea] 2011. [Citado el: 6 de 12 de 2012.] http://www.visual-paradigm.com/product/vpuml/.

Word Wide Web. 2008. [En línea] 09 de 01 de 2008. [Citado el: 25 de 11 de 2012.] http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

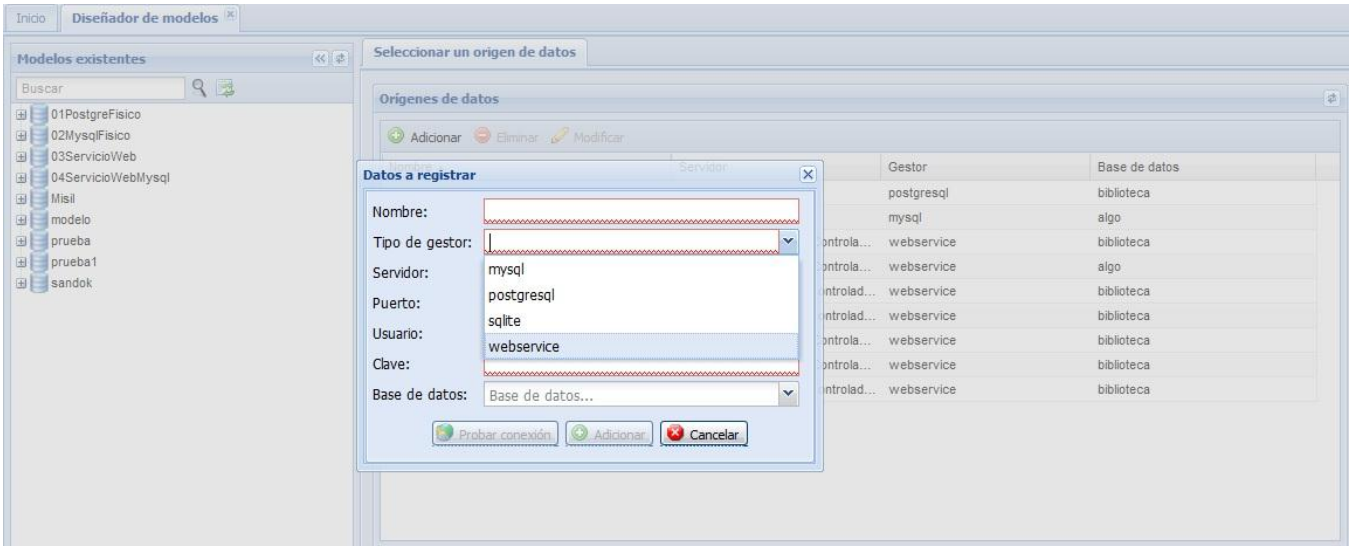
json.org. *json.org* [En línea] [Citado el: 5 de diciembre de 2012]. Disponible en: <http://www.json.org/json-es.html>

netbeans.org. *netbeans.org.* [En línea] [Citado el: 5 de diciembre de 2012]. Disponible en: <www.netbeans.org>

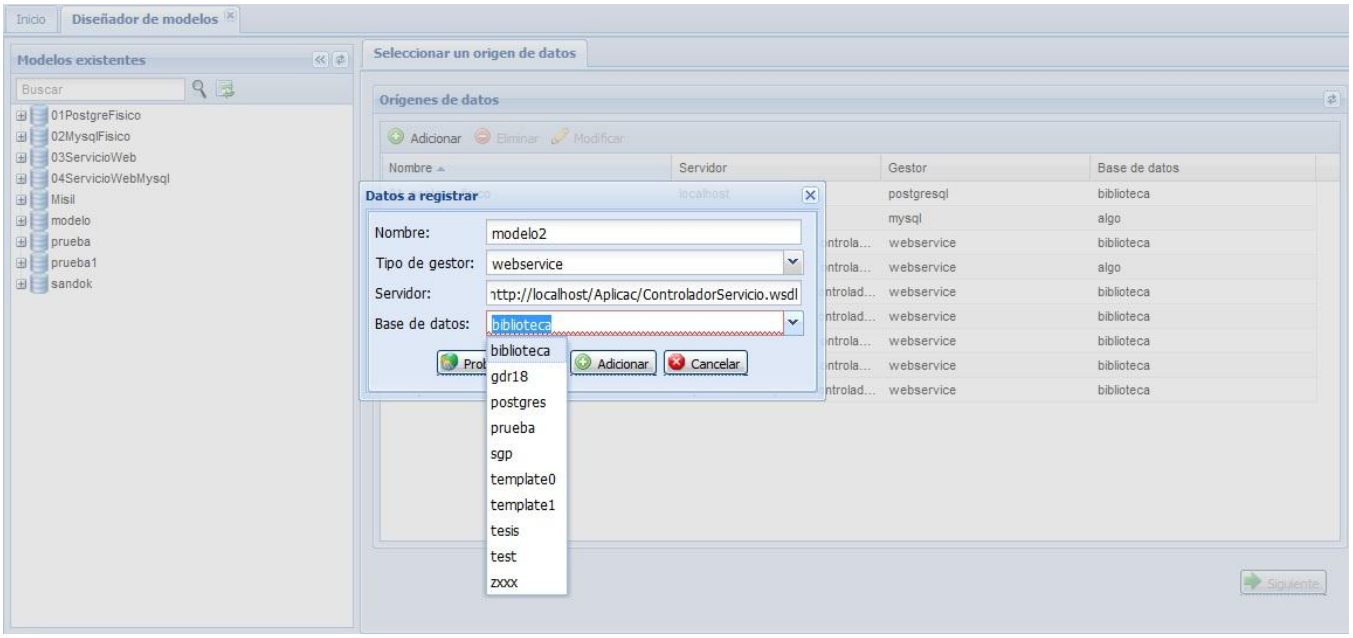
pgadmin.org. *pgadmin.org*. [En línea] [Citado el: 8 de diciembre de 2012]. Disponible en: <http://www.pgadmin.org/index.php>

openerpspain. [En línea] [Citado el: 2 de diciembre de 2012]. Disponible en: <http://openerpspain.com/ck14-articulos/ck20-anexos/iReport/>

ANEXOS



Anexo 3: Integración del servicio web como una forma de conexión



Anexo 4: Parámetros necesarios para hacer la conexión a través del servicio web

GLOSARIO DE TÉRMINOS

API: Interfaz de Programación de Aplicaciones.

CSV: Tipo de documento en formato abierto.

DATEC: Centro de Tecnologías de Gestión de Datos.

Frameworks: Marco de trabajo.

Hibernate: Herramienta de mapeo objeto-relacional que hace mucho más fácil el desarrollo de una aplicación con una base de datos.

HTML: Lenguaje de Marcado de Hipertexto

JavaBeans: Son un modelo de componentes para la construcción de aplicaciones en Java.

JDBC: Java Database Connectivity, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

JFreeChart: Marco de software OpenSource para el lenguaje de programación Java, el cual permite la creación de gráficos complejos de forma simple.

OpenSource: Es el término con el que se conoce al software distribuido y desarrollado libremente.

OpenUP: Proceso Unificado Abierto.

PDF: Formato de Documento Portátil

PHP: Hipertexto Pre-Procesado.

Reporte: Documento caracterizado por contener información u otra materia reflejando el resultado de una investigación adaptado al contexto de una situación.

RUP: Rational Unified Process.

SGBD: Tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje de Modelado Unificado.

World Wide Web: Es un sistema de distribución de información basado en hipertexto enlazados y accesibles a través de Internet.

XLS: Acrónimo de Microsoft Excel Spreadsheet perteneciente a la categoría Extensión de archivo.

XML: Lenguaje de Marcas Extensible.