Universidad de las Ciencias Informáticas

FACULTAD 6

Título:

Componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora:

Yanet Ennis Bouly

Tutores:

MSc. Nara Lidia Pérez Solá

Ing. Aldis Joan Abreu Medina

La Habana, junio de 2013

"Año 55 de la Revolución"

"Nadie ha hecho tanto en tan poco tiempo".

Fidel Castro

DECLARACIÓN DE AUTORÍA

Declaro ser la única autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yanet Ennis Bouly

_____

Firma de la Autora

MSc. Nara Lidia Pérez Solá

_____

Firma del Tutor

Ing. Aldis Joan Abreu Medina

_____

Firma del Tutor

DATOS DE CONTACTO

Tutora:

MsC. Nara Lidia Pérez Solá

Universidad de las Ciencias Informáticas, La Habana, Cuba

e-mail: nara@uci.cu

Tutor:

Ing. Aldis Joan Abreu Medina

Universidad de las Ciencias Informáticas, La Habana, Cuba

e-mail: ajabreu@uci.cu

AGRADECIMIENTOS

Hoy es el día más importante de mi vida, he llegado donde me he propuesto llegar, hoy se gradúa Ennis, ¿quién lo iba a decir?, ni yo misma me creo haber podido llegar tan lejos, por eso aprovecho esta oportunidad para agradecer a todos los que de una forma u otra han aportado su granito de arena para hacer realidad mi sueño.

A Fidel Castro y a la Revolución por la increíble oportunidad de estudiar en esta universidad de excelencia.

A mi tía Mary aunque se haya ido y hoy no esté aquí con nosotros, donde quiera que estés espero que esté orgullosa de tu sobrina, sé que aunque no te pueda ver estas hoy aquí con nosotros, cuando nos dejaste fuiste mi inspiración para terminar la carrera, me esforcé para cumplir tu sueño espero no haberte defraudado.

A mi tía Eva sin tu apoyo, sin tu ayuda ni siquiera hubiera entrado a la escuela, muchas gracias por atenderme como si fuera tu hija, por apoyarme y ayudarme a conseguir mis metas.

A mis tías, Iris, Mariselis, Nereida, y en especial a mi tío Beto gracias por estar siempre que te necesité.

A mi amiga Annelys, gracias por tu ayuda, por tu apoyo incondicional.

A Yudelkis por creer en mí cuando nadie lo hacía, incluso cuando yo no daba motivos para que lo hicieran, gracias por estar siempre pendiente de mí y de mis problemas.

A Annia Sucel ,Salvador, Marisel, Yurima, Miulkenia, Rosy, Aliosmy, Leidanys, Katia, Omar Mar , Garnache, las chicas del 29, gracias por su ayuda, por su apoyo,por sus consejos y por tratar de encaminarme.

A mi tutor Aldis por su ayuda en la elaboración del trabajo de diploma, por enseñarme a ser más responsable e independiente, a mi querida tutora Nara por la dedicación, preocupación, por comprender mis momentos difíciles, estar siempre dispuesta a ayudarme y sobre todo por la paciencia que ha tenido conmigo en todo este tiempo, no cualquiera podría haberlo logrado. Gracias por todo.

Agradecimientos

A todos los que me ayudaron en la elaboración de la tesis, omarito, Eddie, Fernan, Aurelio y sobre todo a ese programador anónimo que muchos llaman Google, gracias por compartir líneas de código y conocimientos que fueron de gran ayuda en el desarrollo de la tesis.

A todos los que han compartido conmigo momentos inolvidables tanto de estudio como de fiestas, gracias por hacer más confortable estos 5 años en la universidad, Potin, Darlon, Sandy, Julio, Yailema, Yuned, Gorda, Katiuska La Cabocla, asaroza , La Flaca, Bernal Boby, Liandry, Jorge Ernesto, Marvel, Ernesto, Fernan, Gleivis si los menciono a todos me ocuparían las 80 páginas de la tesis.

A aquellos que no pudieron terminar con nosotros la carrera pero formaron parte en estos 5 años Helen, Gisela, Soto, Nancita a pesar de que te fuiste siempre desde lejos estuviste pendiente de mí.

A Grey por tus consejos, por tus críticas, por estar ahí en los momentos más difíciles, gracias por aceptarme tal y como soy, por apoyarme cuando tomé unas de las decisiones más difíciles de mi vida, por no darme la espalda cuando más te necesité, eres la mejor amiga del mundo y aunque no te lo haya dicho antes quiero que sepas que te quiero mucho.

A Yosbel por regañarme cada vez que hacía algo mal que generalmente es todo el tiempo, por estar siempre encima de mí para que definiera bien mis prioridades, organizando campañas como "Mi Tesis es lo 1ro" inspiradas en mí, gracias por estar ahí siempre que te necesité, siempre en el momento indicado, aunque tuvieras que dejar de hacer tus cosas para atenderme, disculpa si en algún momento abusé de tu bondad.

A Haymee esa persona tan especial e importante para mí, has sido mucho más que una amiga, puedo decir que te considero como una madre, siempre pendiente de mí, tratando de formarme como mejor persona , has sido mi guía a seguir, estas líneas no alcanzan para decirte lo mucho que te agradezco por estar todo este tiempo a mi lado en los momentos buenos y malos, lo has hecho muy bien moti, has sido mi motor impulsor sin ti no estuviera parada aquí ahora mismo. Te amo mucho mi moti.

Yanet Ennis Bouly.

DEDICATORIA

A mi tia Mary, yo sé que te hubiera encantado estar aquí en estos momentos y verme convertida en una ingeniera.

A mi Familia de amigos Grey, Yosbel y en especial a Haymee, por estar ahí siempre que los necesitaba, ustedes son el verdadero motivo por el cual he llegado tan lejos.

Yanet Ennis Bouly.

RESUMEN

El sistema Generador Dinámico de Reporte (GDR) es un producto del Centro de Tecnologías y Gestión de Datos cuyo principal objetivo es consultar los datos existentes en una base de datos, permitiendo la formulación de reportes en diferentes formatos y modelos personalizables. GDR está compuesto por un conjunto de módulos que brindan las funcionalidades necesarias para la creación de estos reportes con un mínimo de conocimientos y esfuerzo, entre ellos se encuentra: el Diseñador de Modelos, encargado específicamente de la gestión de los orígenes de datos.

Actualmente el sistema extrae y presenta información proveniente de los Sistemas Gestores de Bases de Datos (SGBD): PostgreSQL, MySQL, Microsoft SQL Server, SQLITE y Oracle 10g, pero aún existen instituciones que usan otros muy antiguos como Microsoft Access o muy avanzados como Oracle 11g, provocando que no se pueda aprovechar el uso de esta herramienta.

Este trabajo se centra en el desarrollo de componentes que le permitan al GDR, la realización de reportes dinámicos sobre bases de datos en Access y en Oracle 11g. Para el desarrollo de la solución, se emplea PHP 5.3 como lenguaje de desarrollo, NetBeans 7.3 como entorno de desarrollo, Symfony 1.1 para atender las peticiones del servidor y Ext-JS 2.2 como tecnología del lado del cliente. Además, se elaboran casos de pruebas para validar las funcionalidades de los componentes implementados, garantizando así, la calidad que requiere el proceso de desarrollo de los componentes de conexión.

PALABRAS CLAVES:
Diseñador de Modelos, Generador Dinámico de Reportes, Gestores de Bases de Datos, reportes.

TABLA DE CONTENIDOS

Tabla de Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) constituyen un elemento clave para el desarrollo y avance de toda institución. Con la ayuda de estas, las organizaciones han logrado grandes beneficios como: la optimización de sus recursos, la mejora de sus operaciones, conocer las necesidades de los clientes para ofrecerles un buen servicio, llegar a nuevos mercados y mejorar la comunicación entre sus empleados con clientes y proveedores, logrando así aumentar la productividad de la empresa.

Estos avances tecnológicos han propiciado una alta disponibilidad de información y datos, trayendo consigo además, los problemas asociados a ella: volumen de información abundante a procesar y cuellos de botella en el sistema. Para solucionar estas complicaciones se hace necesario el uso de grandes repositorios de datos, así como, técnicas innovadoras para su almacenamiento, visualización, interpretación y análisis, siendo estas últimas las que más influyen en el proceso de tomas de decisiones por parte de los directivos de la empresa.

Una de las herramientas de apoyo, durante el proceso de análisis, son los generadores de reportes, ya que permiten realizar consultas a las base de datos a través de un lenguaje transparente al usuario y obtener la información de manera estructurada y/o resumida (en forma de reporte), en un diseño atractivo y que sea fácil de interpretar por los usuarios. El reporte, de esta forma, confiere una mayor utilidad a los datos, ya que no es lo mismo trabajar con una planilla de cálculos con 10.000 campos que con un dibujo descriptivo y detallado, que presenta dichos campos de manera gráfica.

En Cuba, muchas empresas y/o proyectos realizan sus análisis a través del Generador Dinámico de Reportes (GDR), implementado en la Universidad de las Ciencias Informáticas (UCI), como parte del fortalecimiento de la gestión de las entidades y la informatización de la sociedad cubana. Dicha herramienta permite la obtención de datos existentes en bases de datos en diferentes formatos y modelos personalizables, facilitando así la toma de decisiones por parte de las entidades.

En una primera versión el GDR, enfocó su desarrollo en la capacidad de extraer y presentar información proveniente de los sistemas gestores de base de datos (SGBD*):* PostgreSQL y MySQL. Fruto del éxito de esta primera experiencia, se extendieron sus funcionalidades hasta permitir conectarlo con otros SGBD como: Microsoft SQL Server, SQLITE y Oracle 10g.

A pesar de todas las ventajas que representa para una empresa trabajar con estos potentes gestores, hay algunas instituciones que usan otros muy antiguos como es el caso del Microsoft Access, gestor no soportado en la versión actual del GDR y que imposibilita su uso en dichos centros. De esta forma el sistema pierde cualidades y afecta tanto a los usuarios, como al producto, llegando a impedir su comercialización.

Por otro lado, existen muchas instituciones que renuevan y actualizan sus tecnologías a versiones más actuales, como es el caso de Oracle que ha actualizado su versión a 11g. En esta nueva versión de Oracle se incluyen los procedimientos almacenados dentro de su catálogo, permitiendo de esta manera obtener esos datos de forma dinámica, por lo que se hace necesaria una actualización del componente de conexión que utiliza el GDR para consultar los datos de una base de datos en Oracle 11g.

Teniendo en cuenta la situación problemática existente, se plantea como **problema de la investigación**: ¿Cómo consultar los datos almacenados en los gestores de bases de datos Access y Oracle 11g para la realización de reportes dinámicos desde el GDR?

Se define como **Objeto de Estudio**: El proceso de visualización de información. Enmarcado en **el campo de acción**: Visualización de la información almacenada en los gestores Microsoft Access y Oracle 11g.

Para solucionar el problema planteado se define como **objetivo general**: Desarrollar componentes para el GDR que permitan la realización de reportes dinámicos sobre bases de datos en Access y Oracle 11g.

Para facilitar su cumplimiento, ha sido desglosado en los siguientes **objetivos específicos**:

1. Realizar el análisis y diseño de los componentes de conexión.
2. Implementar los componentes de conexión.
3. Realizar pruebas que validen el correcto funcionamiento de los componentes de conexión.

Para darle cumplimiento a los objetivos específicos se definieron las siguientes tareas de la investigación:

1. Revisión bibliográfica del marco conceptual referente a los diferentes sistemas gestores de bases de datos y su comunicación con aplicaciones externas.
2. Revisión bibliográfica sobre las herramientas, metodología y XML.

3. Revisión bibliográfica de los mecanismos de conexión entre el GDR y los diferentes sistemas gestores de bases de datos que este soporta.
4. Caracterización de los mecanismos de obtención de metadatos de las bases de datos en Access y Oracle 11g.
5. Manipulación e interpretación de los metadatos obtenidos desde los distintos gestores.
6. Implementación de la conexión con los orígenes de datos.
7. Diseño de los modelos de datos.
8. Salva de los modelos en formato XML en la base del GDR.
9. Realización de pruebas de caja negra para validar el correcto funcionamiento de los componentes.

El presente trabajo está estructurado en un volumen de 78 páginas, compuesto por varias secciones que forman tres capítulos, además de la introducción, conclusiones, referencias bibliográficas, recomendaciones y anexos.

Capítulo 1. Fundamentación Teórica: En este capítulo se realiza una revisión bibliográfica del estado actual de la temática en estudio, con el objetivo de caracterizar y profundizar las herramientas, tecnologías y metodología que se requieren para dar cumplimiento al objetivo de la presente investigación.

Capítulo 2. Análisis y Diseño del sistema: En este capítulo se describen los materiales y métodos usados, así como, la arquitectura propuesta para el sistema a desarrollar, generando todos los artefactos necesarios (según la metodología utilizada): el diagrama de despliegue y el modelo de clases del diseño, este último imprescindible para comprender la lógica del negocio.

Capítulo 3. Implementación y pruebas: En este capítulo se abordan los resultados obtenidos en el desarrollo de la solución y las consideraciones finales mediante la realización de todas las pruebas pertinentes realizadas al software.

CAPÍTULO 1:

Fundamentación teórica

1.1 Introducción

El objetivo principal de este capítulo es abordar los temas relacionados con la generación de reportes, haciendo énfasis en el GDR. Se define la arquitectura del sistema, así como, las herramientas, tecnologías y metodología empleadas para darle cumplimiento al objetivo de este trabajo.

1.2 Generadores de Reportes

Los generadores de reportes son herramientas complementarias de los sistemas de información. Utilizan un lenguaje transparente para el usuario por medio del cual se realizan consultas a la base de datos del sistema para obtener información de ella en forma de reporte.

Actualmente estas herramientas son las más usadas a la hora de realizar análisis sobre la información almacenada en una base de datos, sin ser, necesariamente, especialistas del tema. Estos sistemas además de realizar consultas a las base de datos a través de un lenguaje transparente al usuario, permiten visualizar la información de manera estructurada y/o resumida en un diseño atractivo y fácil de interpretar; y permiten también acoplarse a otras técnicas de análisis.

En el uso de los generadores de reportes, siempre se tienen en cuenta los sistemas gestores de bases de datos soportados (portabilidad del sistema), y la forma en que se acoplaría a los sistemas y/o aplicaciones que lo usan (integración con aplicaciones externas)

1.2.1 Generador Dinámico de Reportes (GDR)

Con el objetivo de desarrollar un Sistema de Gestión de Reportes Dinámicos (SGRD), que permitiese cubrir las necesidades de reportes de cualquier empresa o institución, se concibe el sistema GDR.
Herramienta que brinda la posibilidad de controlar el funcionamiento periódico de una o varias entidades, mediante el diseño de cualquier tipo de reportes, incluyendo los tabulares (con gráficos incluidos), tabla pivote y cruzada, desde los gestores de bases de datos: SQL Server, SQLite, Oracle 10g, MySQL y PostgreSQL.Proporciona a los usuarios la ventaja de agilizar el proceso de la toma de decisiones.

Características:

- Aplicación desarrollada sobre el marco de trabajo Symfony.
- Escrita en el lenguaje de desarrollo PHP.
- Multiplataforma.
- Soporta imágenes y gráficas.
- Soporta varios orígenes de datos.

La última versión estable de este sistema es GDR v1.8. La misma cubre el ciclo básico de la generación de reportes y soluciona el problema de obtener los diferentes informes en los sistemas de gestión de la información que se desarrollan en cualquier entorno empresarial, incluyendo la UCI. A manera de resumen se puede argumentar que el GDR v1.8 es capaz de realizar estudios de investigación o consultas de información de cualquier negocio con el objetivo de tomar decisiones.

Puede ser utilizado tanto en el ámbito de la universidad como en cualquier otra entidad que maneje un proceso de negocio. Aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, el usuario aún debe poseer conocimientos básicos para emplear esta herramienta. (4)

1.2.2 Arquitectura del GDR

La arquitectura del software ha pasado a ser un elemento esencial dentro del proceso de desarrollo de cualquier sistema informático, debido a sus múltiples usos. Pero, llegar a construir una arquitectura para el software que sea a la vez apropiada, para dar cabida a las exigencias de las distintas partes interesadas, y buena en términos absolutos, es una tarea que dista muchísimo de ser sencilla.

El GDR está diseñado con una arquitectura modular y distribuida que ayuda a obtener tanto escalabilidad como flexibilidad. Los procesos se distribuyen entre varios componentes que se pueden ampliar e integrar en soluciones personalizadas. Una típica aplicación para la generación de informes atraviesa por las tres etapas del ciclo de vida de los reportes: creación, administración y entrega; GDR ofrece todas las herramientas necesarias para llevar a cabo estos procesos.

Para soportar los procesos de creación de informes se implementaron varios módulos. El ciclo se inicia con el Diseñador de Modelos, creando una abstracción de la fuente de datos denominada modelo

semántico, el cual describe la estructura de los datos. Por su parte, el Diseñador de reportes se encarga de la conformación del informe, teniendo como entrada el modelo semántico y generando como salida la definición del reporte. En la fase de administración, el Administrador de reportes permite ejecutar tareas para la gestión de los modelos semánticos y las definiciones de los reportes. La última etapa en el ciclo de vida de los reportes es la entrega, el GDR implementa este proceso mediante el Administrador de reportes, el cual posibilita la visualización y exportación del informe a diferentes formatos. (4)

1.2.3 Mecanismo de conexión utilizado por el GDR

La generación de los reportes se basa en su capacidad de extraer la información desde diferentes bases de datos y enmarcarla en el contexto del reporte encargado, a su vez, formatearlo y ajustarlo a las necesidades del cliente final. GDR, para la interconexión con los diferentes gestores de base de datos que este soporta, hace uso de un grupo de clases.

Esta implementación se hace a través de los Objetos de Datos de PHP (PDO por sus siglas en inglés). En la clase gdrPDO se configuran los parámetros necesarios, para garantizar la conexión con cada gestor. A su vez, se implementa una clase por cada uno, con los métodos necesarios, para obtener los metadatos y datos provenientes de estos gestores.

1.3 Sistemas Gestores de Bases de Datos

Un sistema gestor de bases de datos (SGBD) es un software o conjunto de programas que permiten definir, construir y manipular bases de datos para diversas aplicaciones. Éste actúa como interfaz entre los programas de aplicación y el sistema operativo para propiciar un entorno eficiente a la hora de almacenar y recuperar la información de la base de datos. (1)

Uno de los SGBD más usados en entidades cubanas es: Access. A pesar de estar entre los más antiguos, el GDR no soporta la conexión hacia él. Por otro lado, Oracle (v11g) que constituye uno de los gestores más prometedores de la actualidad por las mejoras que ofrece, se ha convertido en una opción optada por muchos, por lo que se hace necesario actualizar el componente de conexión que actualmente tiene implementado el GDR para este gestor.

1.3.1 Microsoft Office Access

Microsoft Access es un sistema interactivo de administración de bases de datos para Windows. Access tiene la capacidad de organizar, buscar y presentar la información resultante del manejo de sus bases de datos. En Access, el motor de base de datos se denomina JET, el cual proporciona flexibilidad en el diseño, almacenamiento, recuperación de datos y producción de información. Este gestor, además del motor de bases de datos, incorpora herramientas para generar aplicaciones y poder ejecutarlas y dispone de macros y módulos de Visual Basic que permiten programar y automatizar tareas.

Entre sus principales características se encuentran:

- Es gráfico, por lo que aprovecha al máximo la potencia gráfica de Windows, ofreciendo métodos usuales de acceso a los datos y proporcionando procedimientos simples y directos de trabajar con la información.
- Facilita la administración de datos, ya que sus posibilidades de consulta y conexión le ayudan a encontrar rápidamente la información deseada, cualquiera que sea su formato o lugar de almacenamiento.
- Es posible producir formularios e informes sofisticados y efectivos, así como gráficos y combinaciones de informes en un solo documento.
- Permite lograr un considerable aumento en la productividad mediante el uso de los asistentes y las macros. Estos permiten automatizar fácilmente muchas tareas sin necesidad de programar.

Las desventajas más importantes son:

- No crea un compilado real (un .EXE). Siempre necesita tener instalado Access para su funcionamiento.
- No es multiplataforma, constituyendo esto su mayor inconveniente, pues sólo está disponible para sistemas operativos de Microsoft, su uso es inadecuado para grandes proyectos de software que requieren tiempos de respuesta críticos (2).

1.3.2 Oracle (v11g)

Oracle es un gestor de base de datos relacional, ha probado ser rápido, confiable, seguro y fácil de gestionar, y a diferencia de Access, es multiplataforma. Es un sistema gestor de datos relacional de última generación que hace uso de los recursos de los sistemas informáticos en todas las arquitecturas de

hardware, lo que permite garantizar su aprovechamiento en ambientes cargados de información, por su capacidad de almacenar y acudir a los datos de forma recurrente. Soporta unos 17 idiomas, funciona automáticamente en más de 80 arquitecturas de hardware y software distintos sin tener la necesidad de cambiar una sola línea de código; esto se debe a que más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de sistemas operativos. Entre las particularidades básicas que ofrece se encuentra, las de permitir trabajar en entornos multiusuario.

Oracle 11g proporciona nuevas e innovadoras funcionalidades que garantizan alto rendimiento, alta escalabilidad, fiabilidad y seguridad mediante el uso de plataformas grid, asegurando altos niveles de calidad de servicio e incrementos de la flexibilidad de negocio reduciendo además, los costos de explotación. Incorpora Secure Files que permite la gestión de todo tipo de datos, incluyendo imágenes, ficheros de texto o tipos avanzados de datos soportados de manera nativa, como XML, imágenes médicas y objetos en 3D. Es el primer gestor de base de datos del mundo en incluir funcionalidades que permiten hacer pruebas de cambios en aplicaciones simulando las cargas reales generadas por los usuarios en los entornos de producción.

Esta nueva versión centra sus cambios en los siguientes grupos:

- Nuevas características en el SQL.
- Nuevas características en el soporte del lenguaje.
- Nuevas características en el PL/SQL.
- Nuevas características en el DBA.
- Nuevas características y mejoras en el Oracle Real Application Clusters
- Nuevas características en el Rendimiento
- Nuevas características de la Seguridad.
- Nuevas características en el Enterprise Manager.(3)

La versión 11g incorpora muchas ventajas, siendo el catálogo una de las más significativa. El catálogo de Oracle es un conjunto de tablas y vistas que contienen la definición de la estructura lógica y física de la base de datos. Describe el contenido de la base de datos y permite obtener directamente los metadatos encargados de describir la estructura y los datos contenidos en una base de datos. (3)

1.3.3 Comunicación con aplicaciones externas

Los SGBD se encargan de proveer interfaces de comunicación con las aplicaciones clientes, estas interfaces son las encargadas de abstraer al SGBD de la tecnología con la que se desarrollan los demás elementos de la aplicación informática. Comúnmente esto se realiza a través de la implementación de algún estándar *SQL* (del inglés Structured Query Language).

Desde la perspectiva de la aplicación que hace uso de la base de datos, Access usa el estándar abierto de conexión a bases de datos: ODBC (del inglés Open Database Connectivity), este es un protocolo que se puede usar para conectar cualquier base de datos en Windows, en el caso de Access es el drivers utilizado para garantizar la conexión con la aplicación, utilizando un DSN (del inglés Data Source Name). En los DSN se especifican los datos que se necesita para conectarse con una base de datos, como el nombre del servidor, el origen de datos y la cadena de conexión.

Por otra parte para interactuar con Oracle, aunque se puede utilizar nuevamente ODBC, los lenguajes de programación modernos implementan conectores específicos para él.

1.4 Herramientas y metodologías

Para el desarrollo de cualquier aplicación informática, la buena selección de las herramientas y la metodología, constituye un paso primordial y básico.

1.4.1 Framework de desarrollo

Un framework es un conjunto de bibliotecas y normas a seguir que ayudan a desarrollar aplicaciones. Este está formado por varios componentes que interactúan entre sí. Permiten implementar de manera eficaz las necesidades del proyecto, convirtiéndose además, en indispensables en los proyectos de un desarrollo de gran escala. (5)

1.4.1.1 Symfony (v1.1)

Symfony es un framework PHP basado en arquitectura Modelo-Vista-Controlador (MVC del inglés Model-View-Controller). Fue escrito para ser utilizado sobre la versión 5 de PHP, ya que hace un amplio uso de la orientación a objetos que caracteriza a esta versión, por lo que se necesita mínimamente PHP 5.3.3 para utilizarlo. A esto se le puede adicionar que fue diseñado para optimizar el desarrollo de aplicaciones

web, proporcionando herramientas para agilizar esta actividad y guiando al desarrollador al orden y buenas prácticas dentro del proyecto (reutiliza conceptos y desarrollos exitosos de terceros y los integra como librerías para ser utilizados). (6)

1.4.1.2 Ext JS (v2.2)

Ext JS es una librería JavaScript que permite construir aplicaciones complejas en internet además de flexibilizar el manejo de componentes de la página como el DOM, peticiones AJAX y DHTML; tiene la funcionalidad de crear interfaces de usuario funcionales. Esta librería incluye:

- Componentes UI del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias de código abierto (GPL) y comerciales

Cuenta con un conjunto de componentes (widgets) para incluir dentro de una aplicación web, como:

- Cuadros y áreas de texto.
- Campos para fechas.
- Campos numéricos.
- Combos.
- Cuadros de selección simple y múltiple.
- Editor HTML.
- Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- Árbol de datos.
- Pestañas.
- Barra de herramientas.
- Menús al estilo de Windows.
- Paneles divisibles en secciones.

Varios de estos componentes están dotados de comunicación con el servidor usando AJAX. También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como:

cuadros de diálogo y Quicktips para mostrar mensajes de validación e información sobre campos individuales. (7)

1.4.2 Entornos de Desarrollo

Un entorno de desarrollo integrado (IDE de sus siglas en inglés Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDE´s pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Los IDE´s proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Basic, Object Pascal, entre otros.

Un IDE presenta las siguientes características:

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis
- Extensiones y Componentes para el IDE
- Integración con Framework
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda (8)

1.4.2.1 NetBeans (v7.3)

NetBeans es un entorno de desarrollo integrado de código abierto, escrito completamente en Java pero puede servir para cualquier otro lenguaje de programación. Una herramienta pensada para escribir, compilar, depurar y ejecutar programas, donde existe además, un número importante de módulos para extenderla.

Este IDE es apoyado por una gran comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación. Además, ofrece todas las herramientas necesarias para crear aplicaciones de escritorio, empresariales, web y móviles haciendo uso de los lenguajes: Java, JavaFX, C++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. Fácil de instalar y se puede ejecutar tanto en Windows, Linux, Mac OS X y Solaris. (9)

Se decidió usar como IDE de desarrollo NetBeans debido a las características anteriormente mencionadas. También se tuvo en cuenta que esta es la herramienta de código abierto usada por la Universidad y el centro de DATEC para la implementación de sus productos.

1.4.3 Lenguajes de programación

Se le llama lenguaje de programación al idioma artificial diseñado para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Un lenguaje de programación está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. Existen distintos niveles de programación, que se engloban en dos grandes categorías:

- Bajo nivel: Mediante el cual se accede al hardware directamente (lenguaje de máquina). Fue el primer lenguaje utilizado en la programación de computadoras.
- Alto nivel: También denominados lenguajes evolucionados, persiguen en primer lugar la independencia de las máquinas, de forma tal, que un mismo programa se puede utilizar en diferentes ordenadores si se dispone de un programa traductor (intérprete o compilador). (10)

1.4.2.2 PHP (v5.3)

PHP (de sus siglas en inglés Hypertext Pre-processor) es un lenguaje interpretado de alto nivel, sencillo, de sintaxis cómoda y dispone de muchas librerías que facilitan en gran medida el desarrollo de las aplicaciones. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas.

Características:

- Dispone de una conexión propia a varios sistemas de base de datos como: MySQL, PostgreSQL y Oracle.
- Incorpora bibliotecas con un conjunto de funciones integradas para realizar útiles tareas relacionadas con la Web.
- Es un producto de código abierto, soportado por una gran comunidad de desarrolladores.
- Es un lenguaje multiplataforma.
- Permite las técnicas de Programación Orientada a Objetos.
- Soporte a XML. (11)

1.4.2.3 Java Script

Es un lenguaje de programación que se utiliza principalmente para crear páginas web. A diferencia del lenguaje HTML, permite la creación de páginas dinámicas, y es muy fácil de aprender. Es técnicamente un lenguaje de programación interpretado por lo que no es necesario compilar los programas para ejecutarlos. Permite integrarlo directamente en páginas HTML. Los programas escritos en este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Algunas de las principales características son las siguientes:

- Es interpretado por el navegador del cliente.
- Está basado en objetos, por lo que hereda todas las ventajas de la Programación Orientada a Objetos (POO).
- Su código se integra en las páginas HTML, incluido en las propias páginas.
- No es necesario declarar los tipos de variables que van a utilizarse.
- Las referencias a objetos se comprueban en tiempo de ejecución. (12)

1.4.4 Lenguajes de Marcado

Un lenguaje de marcado es un conjunto de reglas que establecen qué tipo de marcas han de ser utilizadas, de qué modo se distinguirán las marcas del texto del documento, cómo se insertarán estas y cuáles son las marcas permitidas en cada una de las partes del texto.

Los lenguajes de marcado suelen confundirse con lenguajes de programación. Sin embargo, no son lo mismo, ya que el lenguaje de marcado no tiene funciones aritméticas o variables, como sí poseen los

lenguajes de programación. Históricamente, el marcado se usaba y se usa en la industria editorial y de la comunicación, así como entre autores, editores e impresores. Para cada lenguaje de marcado, los desarrolladores de software pueden construir una aplicación para leer los documentos escritos en ese lenguaje. (13)

1.4.2.4 XML

El lenguaje de marcado extensible XML (de sus siglas en inglés de Extensible Markup Language) no es tan solo un lenguaje de marcado de documentos, es un metalenguaje; es decir un lenguaje para definir un lenguaje de marcado. Los documentos escritos en XML pueden leerse por medio de aplicaciones personalizadas utilizando diferentes objetos de análisis gramatical o pueden combinarse con el lenguaje de estilo extensible (XLS- Extensible Stylesheet Language) para poder mostrarse en un navegador. Para crear un lenguaje de marcado es necesario definir un DTD (Document Type Definition) que es la gramática del lenguaje de marcado, es decir, especifica qué etiquetas van a ser válidas. Todo documento XML está asociado a un DTD, en donde se habrán declarado las etiquetas que use el documento.

Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier aplicación imaginable. (14)

1.4.5 Metodología de desarrollo

Una metodología de desarrollo, es una recopilación de técnicas y procedimientos estructurados en fases para la producción de software. Ésta abarca todo el ciclo de vida del mismo y se fundamentan sobre tres pilares básicos: qué hacer y en qué orden, cómo deben realizarse las tareas y con qué pueden llevarse a cabo. En cada pilar se definen etapas, actividades y tareas que se deben acometer, técnicas que deben emplearse para realizar estas actividades y cuáles son las herramientas de software a utilizar en cada caso. (15)

1.4.2.5 Open UP

Open Unified Process (OpenUP) es un proceso de desarrollo unificado que está basado en Rational Unified Process (RUP). Mantiene las mismas características de RUP, pues está dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. El ciclo de vida de un proyecto según la metodología OpenUP se divide en cuatro fases fundamentales:

- Inicio: se determina la visión del proyecto.
- Elaboración: el objetivo es determinar la arquitectura óptima.
- Construcción: en esta etapa, el objetivo es obtener la capacidad operacional inicial.
- Transición: se obtiene la integración del proyecto.

OpenUP propone 6 flujos de trabajo:

- Requerimientos: en este flujo de trabajo se realizan entrevistas con el cliente para comprender el problema a resolver y se definen los requerimientos.
- Análisis y Diseño: se realiza el diseño de los requisitos que serán después implementados.
- Implementación: en este flujo se realiza la implementación del sistema basándose en el diseño realizado.
- Prueba: busca los defectos a lo largo del ciclo de vida.
- Gestión del Proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Gestión de Configuración y Cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización, actualización y control de versiones.

Características fundamentales de OpenUp:

- Colaboración para unificar intereses y compartir conocimientos.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Enfoque en la articulación de la arquitectura.
- Desarrollo continúo para obtener realimentación y realizar las mejoras respectivas.

Roles presente en la metodología OpenUp:

- Gerente de Proyecto
- Analista
- Diseñador
- Arquitecto de software
- Desarrollador

- Stakeholder (interesados)
- Tester (Probador) (15)

1.4.6 Herramienta de Modelado

CASE, (acrónimo de Computer Aided Software Engineering) es el tipo de herramienta destinadas a aumentar la productividad en el desarrollo de software, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza la construcción de aplicaciones con calidad y a un menor costo. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como la reducción del costo de las mismas en términos de tiempo y dinero. 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 costos y hasta la implementación de parte del código, automáticamente a partir de un diseño.

1.4.6.1 Visual Paradigm (v8.0)

Es una herramienta de ingeniería de software multiplataforma, que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, generación del código fuente de los programas y la documentación de los mismos. Su propósito general es: soportar el ciclo de vida del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.

Esta herramienta genera diseños centrado en casos de uso y enfocados al negocio contribuyendo así a un software de mayor calidad. Usa un lenguaje estándar y común a todo el equipo de desarrollo que facilita la comunicación. Mantiene capacidades de ingeniería directa e inversa y posibilita que el modelo y el código permanezcan sincronizados en todo el ciclo de desarrollo. (17)

Se decidió usar Visual Paradigm debido a las características antes mencionadas y que es una herramienta CASE profesional. También se tuvo en cuenta que esta es la herramienta usada por la Universidad y DATEC. La misma posee una licencia comercial y la UCI cuenta con la misma para su uso.

1.4.7 Lenguaje de Modelado

El modelado de sistemas de software es una técnica que ayuda al ingeniero de software a visualizar el sistema que desea construir. De ahí que los modelos pueden utilizarse para la comunicación con el cliente, grupo de desarrollo y otros factores que intervienen en el proceso de desarrollo.

1.4.7.1 UML

Lenguaje Unificado de Modelado (UML por sus siglas en inglés, Unifield Modeling Languaje) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Es una exigencia de la gran mayoría de instituciones dentro de su Plan Informático estratégico, que los desarrollos de software sean bajo una arquitectura en capas, se formalicen con un lenguaje estándar y unificado. Es decir, se requiere que cada una de las partes que comprende el desarrollo de todo software de diseño orientado a objetos, se visualice, especifique y documente con lenguaje común.

Este lenguaje cumple con los procesos de negocios y funciones del sistema, cuenta con una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos.

Los principales beneficios de UML son:

- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos. (16)

1.5 Conclusiones Parciales

En este capítulo se realizó un estudio de las herramientas, tecnologías y metodología empleadas en la investigación. Se seleccionó PHP 5.3 como lenguaje de desarrollo para incrementar el dinamismo de la aplicación y utilizar las potencialidades de la red. Se utilizará NetBeans 7.3 como interfaz de desarrollo por ser libre y tener gran integración con PHP 5.3. Se seleccionó como marco de trabajo Symfony 1.1, para que atienda las peticiones del servidor y Ext-JS 2.2 como tecnología del lado del cliente para la reutilización de componentes y la validación de interfaces. Para llevar a cabo el modelado del sistema se

empleará Visual Paradigm 8.0 de conjunto con el Lenguaje Unificado de Modelado UML y se seguirán las pautas que plantea la metodología de desarrollo OpenUp para construir software con buenas prácticas.

CAPÍTULO 2:

Análisis y Diseño del Sistema

2.1 Introducción

En el presente capítulo se desarrollan los artefactos del flujo de trabajo análisis y diseño, correspondiente a la metodología de trabajo OpenUp. En este sentido se realizan los diagramas de clases del diseño, mediante los cuales se muestra la estructura estática del sistema y se describen las clases más relevantes. Se desarrollan los diagramas de interacción, en los cuales se muestran los aspectos dinámicos que poseerá el sistema, así como, los requisitos funcionales y no funcionales del software, y los patrones de diseño usados en la solución del mismo.

2.2 Modelo de Dominio

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Se crea con el objetivo de documentar los conceptos dominantes y el vocabulario del sistema. En él, no solo se determinan conceptos, métodos y cualidades importantes del negocio, sino, se identifican las relaciones entre ellos.

A continuación se presenta el Modelo de Dominio (ver figura 1) y la descripción de sus clases.

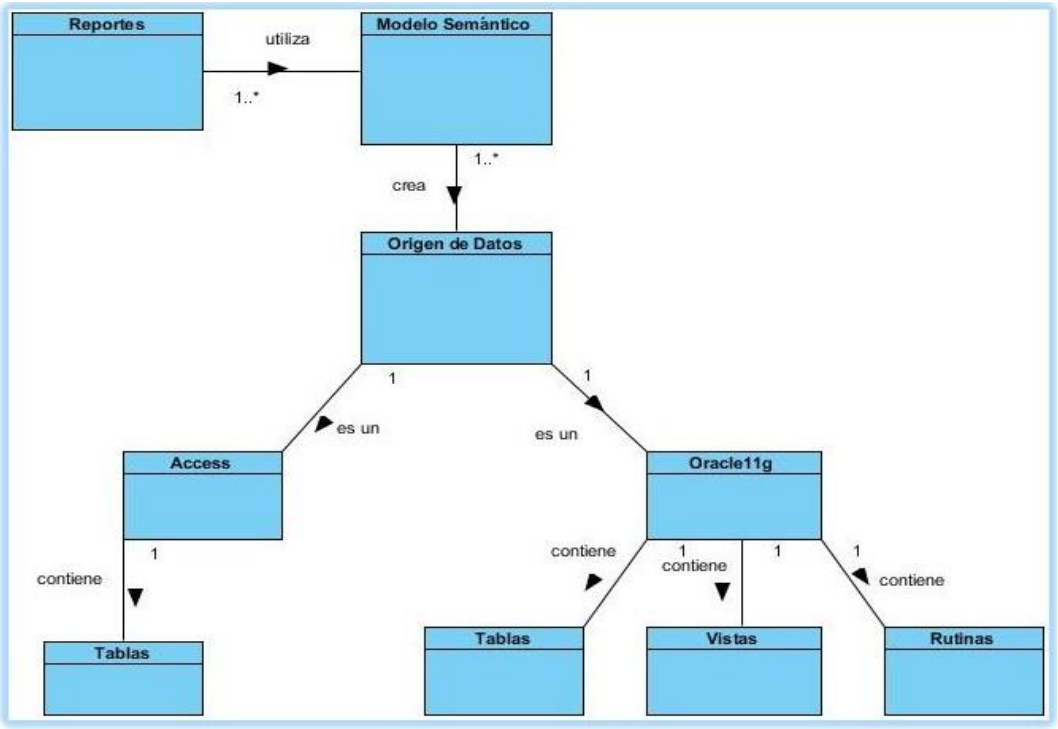


Figura 1. Modelo de Dominio

En la figura 1 se muestra la clase Reportes, para crear estos reportes, es imprescindible tener previamente creado un modelo semántico, es importante destacar que en el proceso de generación de un modelo, este está asociado a un único origen de datos, es decir que necesita de un origen de datos que lo provea, por el contrario de un único origen de datos se pueden crear varios modelos. Estos orígenes de datos pueden ser creados de diferentes gestores de base de datos que soporta el GDR, de los cuales se obtienen sus entidades, en este caso se obtienen de el gestor de base de datos Access que solo se obtendrán sus tablas y Oracle11g sus tablas, vistas y rutinas.

Descripción de las clases del dominio:

Reportes: 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 y CSV.

Modelo Semántico: Es un modelo de datos usado para almacenar en forma de fichero 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.

Origen de Datos: Contiene los datos que permiten conectar al GDR con los SGBD para obtener sus metadatos y posteriormente crear un modelo semántico.

Access: Gestor de base de datos desde donde se obtienen los metadatos.

Oracle11g: Gestor de base de datos desde donde se obtienen los metadatos.

Tablas, Rutinas, Vistas: Componentes de la base de datos necesarios para crear reportes.

2.3 Requisitos del sistema

Los requisitos se definen como la cualidad necesaria de un sistema, una declaración que identifique una capacidad, una característica, o un factor de calidad de un sistema y que proporcione valor y utilidad al cliente o usuario. Es la descripción de los servicios y restricciones de un sistema de software, es decir, lo que el software debe hacer y bajo qué circunstancias debe hacerlo.

2.3.1 Requisitos funcionales

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactarlos. A continuación, se muestra en la tabla 1, la descripción de los requisitos funcionales, así como las entradas y salidas necesarias para le correcta ejecución de los mismos:

Nº	Funcionalidad	Entrada	Salida	Descripción
RF1	Listar las bases de datos	Recibe los parámetros para tener acceso a las bases de datos en el caso de Oracle11g (servidor, tipo de	Lista las bases de datos utilizables.	El sistema debe permitir mostrar las bases de datos disponibles.

		gestor, puerto, usuario y contraseña) y en el caso de Access los parámetros opcionales usuario y clave, junto a los parámetros obligatorios base de datos y nombre.		
RF2	Adicionar Origen de datos.	Se especifican los detalles del nuevo origen de datos que se va a adicionar, en el caso de Oracle11g (servidor, tipo de gestor, puerto, usuario y contraseña) y en el caso de Access los parámetros opcionales usuario y clave junto a los parámetros obligatorios base de datos y nombre.	Muestra una lista con todos los orígenes de datos existentes.	Se adiciona en el sistema un nuevo origen de datos.
RF3	Probar Conexión a un origen de datos.	Se especifican los detalles del nuevo origen de datos que se va a adicionar, en el caso de Oracle11g (servidor, tipo de gestor, puerto, usuario y contraseña) y en el	Mensaje de validación si se estableció, bien o no, la conexión.	Se prueba establecer la conexión a un origen de datos, para verificar que funcione correctamente.

		caso de Access los parámetros opcionales usuario y clave junto a los parámetros obligatorios base de datos y nombre.		
RF4	Listar Tablas	Recibe como parámetro el id de la base de datos seleccionada para poder tener acceso a sus tablas.	Muestra en una lista las tablas de la base de datos seleccionada.	El sistema debe brindar la funcionalidad de obtener las tablas de la base de datos seleccionada.
RF5	Listar tablas relacionadas.	Recibe como parámetro el id de la base de datos seleccionada para poder tener acceso a las tablas relacionadas.	Muestra una lista de tablas relacionadas.	El sistema debe brindar la funcionalidad de obtener las tablas que se relacionan con las seleccionadas.
RF6	Listar atributos de tablas.	Recibe como parámetro el id de la base de datos seleccionada para poder tener acceso a los atributos de las tablas de esa base de datos.	Muestra una lista con los atributos de las tablas.	El sistema debe permitir obtener los atributos de las tablas.
RF7	Listar Vistas	Recibe como parámetro el id de la base de datos	Muestra una lista con todas las	El sistema debe brindar la funcionalidad de obtener las vistas de la

		seleccionada, para poder tener acceso a sus vistas.	vistas que posee la base datos seleccionada.	base de datos seleccionada.
RF8	Listar atributos de Vistas	Recibe como parámetros el id del modelo, el nombre de la vista y el esquema al que pertenece para así obtener los atributos de la vista.	Muestra una lista con los atributos de las vistas.	El sistema debe permitir obtener los atributos de las vistas.
RF9	Listar Rutinas	Recibe como parámetro el id de la base de datos seleccionada para poder tener acceso a sus rutinas.	Muestra una lista con las rutinas de la base de datos seleccionada.	El sistema debe brindar la funcionalidad de obtener las rutinas de la base de datos seleccionada.
RF10	Listar atributos de rutinas	Recibe como parámetros el id del modelo, el nombre de la rutina y el esquema al que pertenece para así obtener los atributos de la rutina.	Muestra una lista con los atributos de las rutinas.	El sistema debe permitir obtener los atributos de las rutinas.
RF11	Exportar Reporte	Recibe como parámetros el id del reporte, el límite (cantidad de	Se exporta el reporte seleccionado al formato	Se obtiene los datos para la construcción del reporte.

		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.	predefinido.	

Tabla 1. Requisitos Funcionales

2.3.2 Requisitos no funcionales

Los requisitos no funcionales describen atributos del sistema o del ambiente del mismo, que no responden a funciones del sistema, sino que son necesarios para que este funcione correctamente.

Los requisitos no funcionales se refieren específicamente a las propiedades emergentes de un sistema, tales como la fiabilidad, el tiempo de respuesta o precisión, capacidad de almacenamiento o tipo de plataforma (lenguajes de programación y/o sistemas operativos, etc.).

Interfaz:

- El sistema tendrá una apariencia agradable y ajustada a la interfaz del GDR.
- Las interfaces de usuario serán diseñadas a modo de aplicaciones *RIA* (*Rich Internet Application*) lo que permite a los usuarios contar con aplicaciones web con una experiencia de usuario similar a la de las aplicaciones de escritorio. Para lograr este fin se usará la librería *javascript ExtJS*, la cual conjuga una serie de componentes visuales que proveen funcionalidades, que ayudan al diseño de este tipo de aplicaciones web con apariencia de escritorio.

Software:

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

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian GNU/Linux 4.0 o superior.

- Paquetes: apache2, php5, libapache2-modphp5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-xsl, php5-gd, php5-odbc,php5-pear,libaio1,libpq-dev,oci8, PDO_OCI.
- Navegador Firefox en su versión 4.0 o superior.
- Usuario con privilegios de administración.

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

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian GNU/Linux 4.0 o superior.
- PostgreSQL versión 8.3 o superior.
- PGAdmin III 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

El servidor donde se instalará la aplicación 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.

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.

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 como mínimo.
- 20 GB de espacio en disco duro.

2.4 Casos de Uso del sistema

El caso de uso es una estructura para describir la forma en que un sistema lucirá para los usuarios potenciales. Es una colección de escenarios iniciados por una entidad llamada actor (una persona, un

componente de hardware). Un caso de uso debería dar por resultado algo de valor ya sea para el actor que lo inicia o para otra entidad.

2.4.1 Diagrama de casos de uso del sistema (DCUS)

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 (CU) engloban los requisitos funcionales de un sistema, representando las funciones que la aplicación puede ejecutar.

A continuación se presenta el DCUS (ver figura 2) de la presenten investigación.

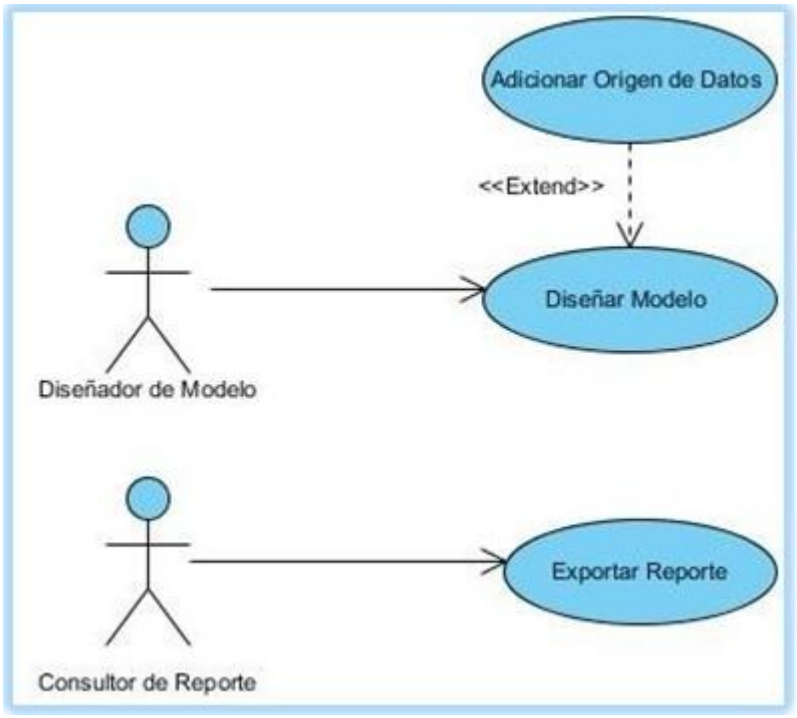


Figura 2. Diagrama de Casos de Uso del Sistema

En el DCUS mostrado en la Figura 2, el actor Diseñador de Modelos es el encargado de iniciar los casos de uso Diseñar Modelo y seguidamente Adicionar un origen de datos, vale destacar que para poder diseñar un modelo tiene que existir el origen de datos desde donde se extraerán los metadatos con los cuales se diseñará dicho modelo. El actor Consultor de Reportes es el encargado de inicializar el caso de uso Exportar Reporte.

- **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, desde donde se extraerá la información necesaria para la realización de los modelos y posteriormente los reportes.

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

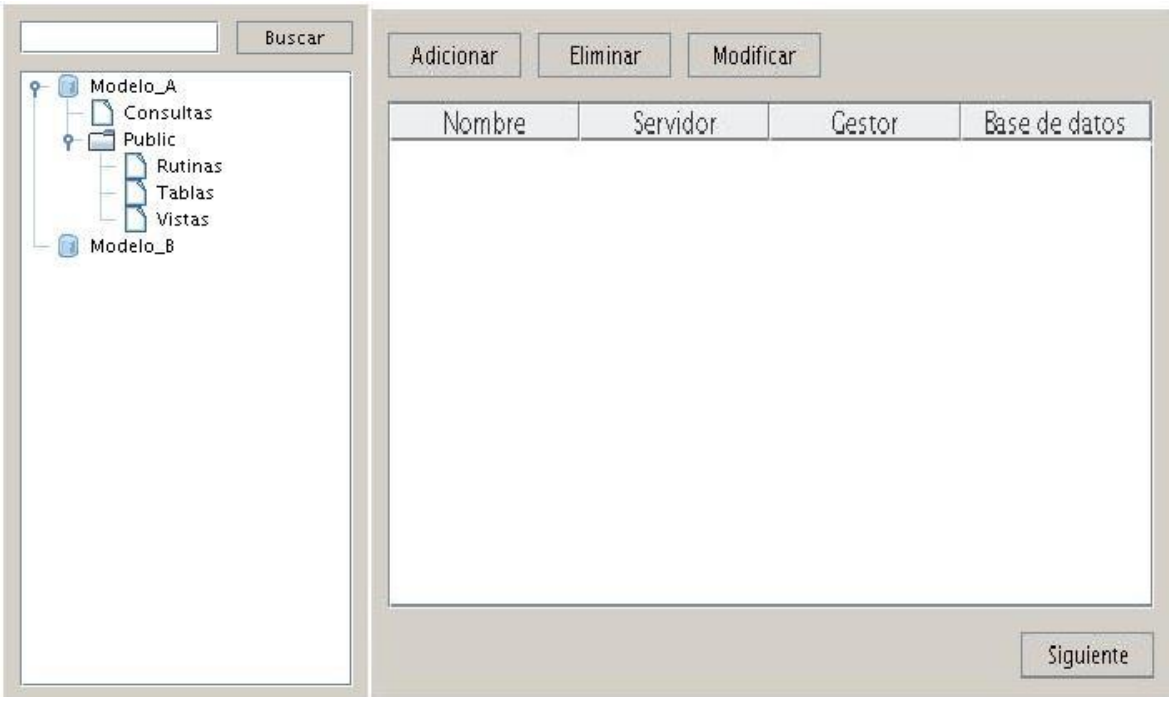
2.4.2 Descripción del Caso de Uso arquitectónicamente significativo del sistema.

Actor	Descripción
Diseñador de Modelo	Actor que interactúa con el módulo Diseñador de Modelos que es el escenario donde se realizarán estos casos de uso.

Tabla 2. Actor del sistema

A continuación se representa la descripción del caso de uso Diseñar Modelo. La descripción del CUS Exportar Reporte, se puede encontrar en el artefacto Modelo del Sistema, del expediente de proyecto del GDR.

Caso de Uso:	Diseñar modelo.
Actores:	Diseñador de Modelo.
Resumen:	El CU, 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.
Referencias:	RF4, RF5, RF6, RF7, RF8, RF9, RF10
Precondiciones:	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 modelos".*	2. El sistema muestra la interfaz "Diseñador de Modelos" brindándole un listado de los modelos existentes y los orígenes de datos previamente creados. Si el actor decide crear un nuevo modelo, ver Sección: "Adicionar modelo". Si el actor decide crear un nuevo origen de datos, ver descripción del CU "Adicionar origen de datos".
Prototipo de Interfaz	
	
Poscondiciones	Se adiciona a la lista de modelos el modelo creado.
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 un mensaje de información y posteriormente muestra las entidades pertenecientes al origen de datos, agrupadas por tablas, vistas y rutinas. *(Ver Anexo 5)*
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 6.1. Si el actor desmarca las entidades previamente seleccionadas para crear el modelo semántico ver flujo Alterno 6.2.*
7. El actor selecciona la opción “Siguiente”.	8. El sistema muestra un mensaje de información y muestra además los campos asociados a cada entidad previamente seleccionada, *Ver Anexo 6,* 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 la interfaz “Generar Modelo” 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, en caso de que el actor posicione el cursor encima del campo marcado en rojo el sistema muestra un mensaje “Campo obligatorio”.*
11. El actor introduce un nombre para el	12. El sistema habilita la opción “Aceptar”. *En*

nuevo modelo a generar.	*caso de que el actor seleccione la opción "Cancelar", ver Flujo Alterno 10.1.*
13. El actor selecciona el botón "Aceptar".	14. El sistema muestra un mensaje de confirmación, *Si el actor se decide por la opción "Sí" ver paso 15 del flujo normal de eventos. En caso que el actor seleccione la opción "No" ver flujo Alterno 14.1.*
15. El actor selecciona la opción "Si".	16. El sistema muestra mensaje de información y genera el modelo semántico carga la interfaz con el listado de los orígenes de datos y actualiza el listado de modelos semánticos con el nuevo modelo actualizado.

Prototipo de Interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	6.1. Ver flujo normal de eventos 4.
	6.2 El sistema deshabilita el botón siguiente.
	8.1. Retorno al paso 4 del *Flujo Normal de Eventos.*
	10.1. El sistema cierra la interfaz "Generar Modelo".
	14.1 El sistema cierra el mensaje de confirmación y mantiene la interfaz "Generar

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

Tabla 3. Descripción del CU Diseñador de Modelo

A continuación se presenta la descripción del Caso de Uso Adicionar origen de datos.

Caso de Uso:	Adicionar origen de datos
Actores:	Diseñador de Modelos
Resumen:	El caso de uso comienza cuando en el CU "Diseñar modelo", se selecciona la opción "Adicionar", en el Diseñador de Modelo. Permite adicionar un nuevo origen de datos desde donde se extrae la información requerida. Finaliza al guardar el origen de datos.
Precondiciones	
Referencias	RF1,RF2,RF3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción *"Diseñador de modelos".*	2. El sistema muestra una interfaz "Orígenes de datos", (*ver Anexo 4)* brindándole un listado de los orígenes de datos existentes, brinda la opción de crear un nuevo origen de datos mediante el botón "Adicionar".
3. El actor en el área de origen de datos selecciona la opción de Adicionar.	4. Se muestra la interfaz "Datos a Registrar". (ver Anexo 7)
5. El actor selecciona el tipo de gestor del cual se quiere crear el origen de datos; en caso de que seleccione Access ver sección "*Access*", en caso	

de seleccionar *Oracle11g* ver sección "*Oracle11g*". En caso que el actor seleccione la opción ¨Cancelar¨ ver flujo alterno 1.1.	
Flujos Alternos	
Actor	**Sistema**
	1.1 Cierra la interfaz donde se muestran los datos a llenar para adicionar el origen de datos.
Sección 1: Access	
Acción del Actor	**Respuesta del Sistema**
1. El actor introduce el campo obligatorio nombre y los campos opcionales usuario, clave y selecciona el campo Base de datos.	2. El sistema muestra las bases de datos disponibles. (ver Anexo 8) En caso de que el actor se decida por la opción "Cancelar", ver Flujo Alterno 2.1.
3. El actor selecciona la base de datos.	4. El sistema habilita las opciones "Probar conexión" y "Adicionar". *Si el actor selecciona la opción "Probar conexión", ver paso 5 del Flujo Normal de Eventos. Si el actor se decide por la opción "Adicionar", ver paso 7 del Flujo Normal de Eventos. En caso de que el actor se decida por la opción "Cancelar", ver Flujo Alterno 2.1.*
5. El actor selecciona la opción "Probar conexión".	6. El sistema muestra un mensaje de tiempo de espera y seguido muestra un mensaje con el resultado de la prueba de la conexión. (ver Anexo 10) *Si el actor selecciona la opción "Adicionar", ver paso 7 del Flujo Normal de Eventos. En caso que el actor se decida por la opción "Cancelar", ver Flujo*

	Alterno 2.1.
7. El actor selecciona la opción ‘‘Adicionar’’.	8. El sistema muestra un mensaje de confirmación. Si el actor se decide por la opción “Sí”, ver paso 9 del Flujo Normal de Eventos; en caso de que el actor seleccione la opción “No” el sistema cierra el mensaje de confirmación y mantiene la interfaz “Datos a registrar”.
9. El actor selecciona la opción “Sí”.	10. El sistema adiciona el nuevo origen de datos a la lista de orígenes de datos y actualiza el mismo. (ver Anexo 4)
Flujos Alternos	
Actor	**Sistema**
	2.1 El sistema ignora la petición de adicionar un nuevo origen de datos y cierra la ventana.

Prototipo de Interfaz

Sección 2: Oracle11g	
Actor	**Sistema**
1. El actor introduce los campos obligatorios como son nombre, servidor, puerto, usuario, clave y Base de datos.	2. El sistema habilita las opciones “Probar conexión” y “Adicionar”. *Si el actor selecciona la opción “Probar conexión”, ver paso 3 del Flujo Normal de Eventos. Si el actor se decide por la opción “Adicionar”, ver paso 5 del Flujo Normal de Eventos. En caso de que el actor se decida por la opción “Cancelar”, ver Flujo Alterno 2.1.*
3. El actor selecciona la opción “Probar conexión”.	4. El sistema muestra un mensaje de tiempo de espera y seguido muestra otro mensaje con el resultado de la prueba de la conexión, ver Anexo 10.
5. El actor selecciona la opción ‘‘Adicionar’’.	6. El sistema muestra un mensaje de confirmación. *Si el actor se decide por la opción “Sí”, ver paso 7 del Flujo Normal de Eventos*; *en caso de que el actor seleccione la opción “No”* el sistema cierra el mensaje de confirmación y mantiene la interfaz “Datos a registrar”.
7. El actor selecciona la opción “Sí”.	8. El sistema adiciona el nuevo origen de datos a la lista de orígenes de datos y actualiza el mismo. (ver Anexo 4)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra un mensaje informando que los datos son incorrectos 2.1 El sistema ignora la petición de a

	adicionar un nuevo origen de datos y cierra la ventana.

Prototipo de Interfaz

Poscondiciones	Se registra en el sistema el origen de datos

Tabla 4. **Descripción del CU Adicionar origen de datos**

2.5 Modelo de diseño

El modelo del diseño se utiliza para documentar el diseño de un sistema, es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de la implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación del sistema y es de ese modo utilizada como una entrada fundamental de las actividades de implementación.

2.5.1 Diagrama de clases del diseño

El diagrama de clases del diseño (DCD) muestra las propiedades y funcionalidades de cada clase en el lenguaje de desarrollo y tecnologías seleccionados. Expresa la colaboración y responsabilidades de cada clase entorno al sistema que conforman y muestra cómo quedaría implementada toda la aplicación en términos lógicos.

El siguiente diagrama de clase del diseño (ver figura 3) contiene las principales clases con sus respectivos métodos y atributos para el caso de uso Adicionar Origen de datos.

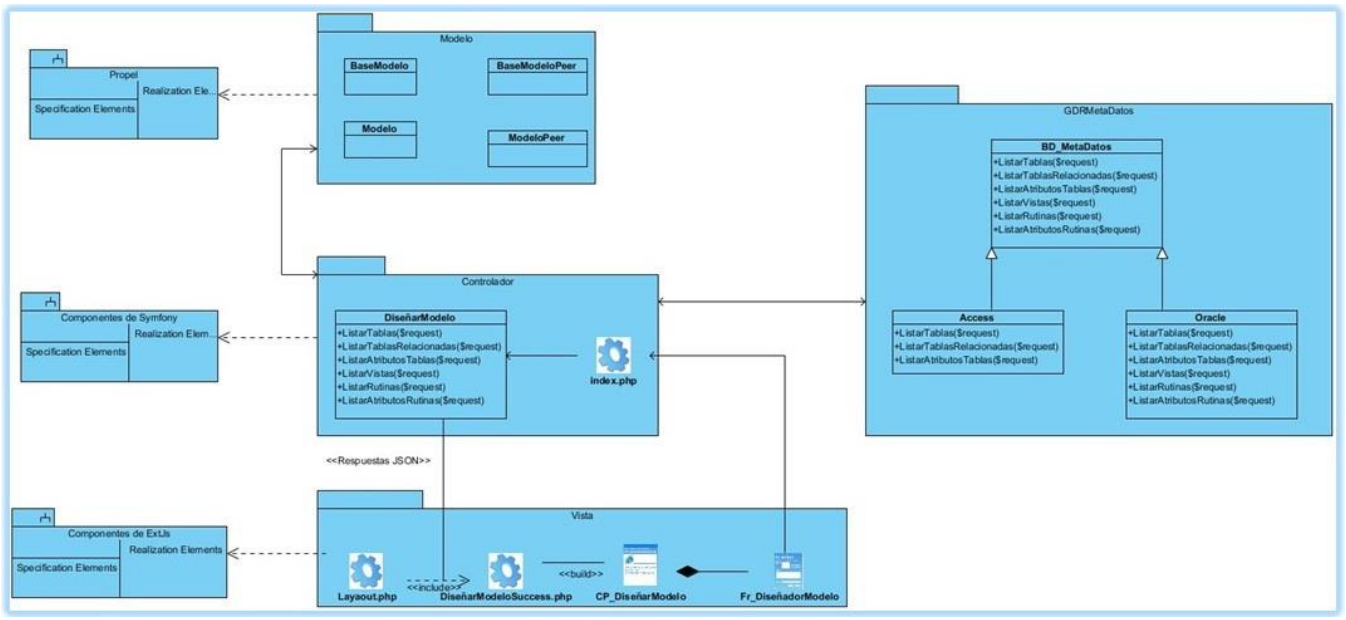


Figura 3. Diagrama de clases del diseño para el CUS Diseñar Modelo

En este diagrama, los elementos del modelo corresponden a las clases generadas por el ORM Propel, modelado por el subsistema de igual nombre. El mismo crea cuatro clases por cada tabla de la base de datos. Las acciones, elementos que construyen las respuestas del servidor, están implementadas en el lenguaje PHP, con la dependencia del subsistema “Componentes de Symfony y comprobadas por el controlador frontal “index.php”. Las solicitudes se realizan por medio de la tecnología AJAX y las respuestas son devueltas específicamente en formato JSON. Los elementos del lado del cliente corresponden a componentes específicos del negocio, en lenguaje JavaScript utilizando los componentes del subsistema “Componentes ExtJS”.

2.5.2 Diagrama de interacción (Secuencia o colaboración)

Los diagramas de interacción (secuencias o colaboración) muestran la forma en que un grupo de objetos se comunican (interactúan) entre sí a lo largo del tiempo, estos constan de objetos, mensajes entre estos objetos y una línea de vida del objeto representada por una línea vertical. El siguiente diagrama de secuencia (ver figura 4) pertenece al escenario Adicionar origen de datos.

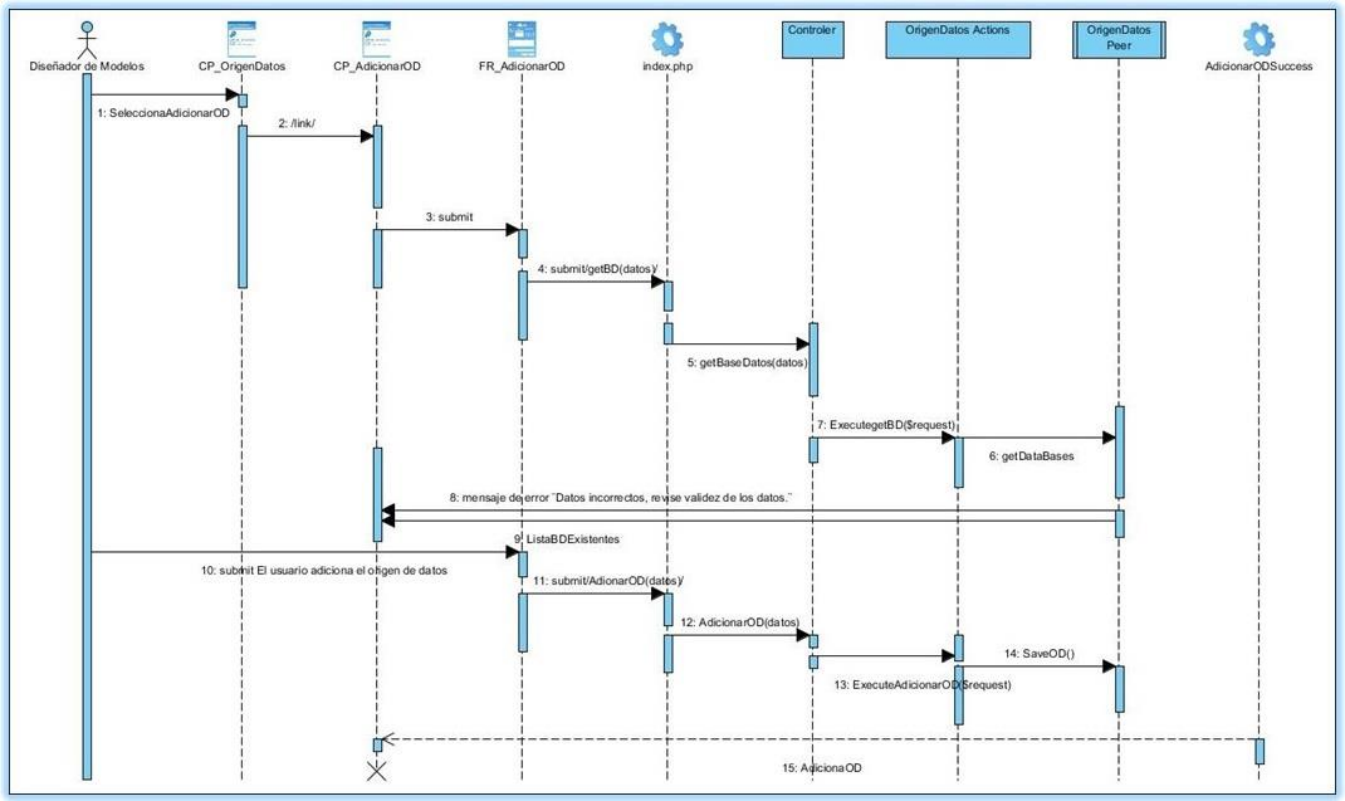


Figura 4. Diagrama de secuencia correspondiente al escenario Adicionar Origen de Datos

En este diagrama de secuencia el actor Diseñador de Modelos selecciona la opción Adicionar para registrar un origen de datos en el sistema. La página cliente del origen de datos se encarga, a través de la operación “link”, de abrir la página cliente Adicionar Origen de datos la cual contiene un formulario donde el usuario introduce los datos obligatorios para crear un origen de datos y envía estos datos mediante la operación “submit” al controlador frontal index.php. Este a su vez le envía al origen de datos actions la tarea de ejecutar la acción “ExecutegetBD($request)”, posteriormente se le envía a la página cliente Adicionar origen de datos la lista de las base de datos disponibles, seguidamente el usuario selecciona la

base de datos desde donde quiere extraer la información y envía mediante la operación submit todos los datos necesarios y obligatorios para crear el origen de datos.

2.6 Patrones de software

Los patrones de software estandarizan principios y buenas prácticas en la solución de sistemas informáticos. Estos han permitido agrupar soluciones a problemas existentes en la construcción de aplicaciones y generar una respuesta común que resuelva de forma genérica las principales deficiencias en la creación de software.

2.6.2 Patrones de arquitectura

Los patrones arquitectónicos ofrecen soluciones a problemas de arquitectura en la ingeniería de software. Estos expresan un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. Asimismo, describen los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre su empleo. Las arquitecturas más utilizadas para el desarrollo de software son: Monolítica, Cliente-Servidor y Tres Capas. Esta última constituye una especialización de la arquitectura Cliente-Servidor, donde la carga se divide en tres capas, con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (modelado el negocio) y una tercera para el almacenamiento (persistencia), una capa solamente tiene relación con la siguiente.

MVC

MVC, acrónimo de (Model View Controller) es un patrón de arquitectura que pertenece a la familia de los estilos arquitectónicos de Llamada y Retorno. El mismo separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos: Modelo, Vista y Controlador.

Por ende, el modelo encapsula los datos y la funcionalidad de la aplicación. La vista despliega la información contenida en el modelo. El controlador está asociado a cada vista, recibe entradas que traduce en invocaciones de métodos del modelo o de vista. De ahí que su empleo garantiza una mayor claridad en el diseño y facilita una mayor escalabilidad. Asimismo, fomenta la reutilización de los componentes. Simplifica el mantenimiento de los sistemas. Igualmente, separa la interfaz, lógica de negocio y de presentación

2.6.1 Patrones de diseño

Los patrones de diseño constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Además son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. (20)

En la solución del sistema se aplicaron principalmente los siguientes patrones:

- **Controlador Frontal:** Se aplica en la creación de un único punto de acceso para todas las peticiones realizadas al módulo, en este caso el "index.php" escucha las peticiones que vienen desde una URL, luego se encarga de llamar al controlador específico, el cual maneja la acción requerida para satisfacer la petición realizada.

- **Alta Cohesión**: Symfony organiza el trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases que tienen una alta cohesión. Por ejemplo la clase "getDataBases" tiene la responsabilidad de definir las acciones para obtener las bases de datos disponibles según el origen de datos.

- **Experto:** Symfony utiliza Propel como ORM, el cual crea cuatro clases: dos clases de acceso a datos que trabajan directamente con la base de datos, dos de abstracción de datos que son las que tienen los atributos necesarios para realizar dicha función. Por tanto se debe implementar la responsabilidad de realizar las acciones directamente con la base de datos y ahí es donde se aplica el patrón. Por ejemplo, la clase "baseOrigenDatos" no conoce los atributos para comenzar a interactuar con la base de datos, tan sólo le avisa a la "baseOrigenDatosPeer" lo que tiene que hacer, pues esta si tiene todos los datos necesarios para ejecutar la acción.

- **Command (Orden):** En el framework Symfony se utiliza este patrón, el cual está representado por las acciones, donde cada acción, encapsula una petición en un objeto.

- **DAO:** Se ve evidenciado este patrón en la clase "gdrPDO.php" en la cual se tiene implementado el método "getGdrPDO()". En este método es donde se configura la conexión a los distintos gestores de base de datos que soporta GDR, se hace una instancia de este en las clases de acceso a datos para reconfigurar la conexión según el gestor que sea.

- **Factory:** Se evidencia ya que normalmente el patrón DAO se completa con algún tipo de Factoría, en este caso una clase a la que, al pedirle la "gdrPDO" decide cuál de las implementaciones instanciar y la devuelve.

- ***Singleton*:** Se evidencia el uso de este patrón en la clase "gdrPDO.php", la cual contiene el método "getGdrPDO()" para la configuración del acceso a los diferentes gestores de base de datos que el GDR soporta y en la clase "DBMetaData.php" se realiza una instancia de este método para configurar la conexión según el gestor que sea.

2.7 Diagrama de Despliegue

El diagrama de despliegue permite mostrar la arquitectura, en tiempo de ejecución, del sistema respecto al hardware y software. Este se utiliza en el diseño y la implementación, es más limitado que el diagrama de componentes, en el sentido de que representa la estructura del sistema solo en tiempo de ejecución, pero no en tiempo de desarrollo o compilación. (21)

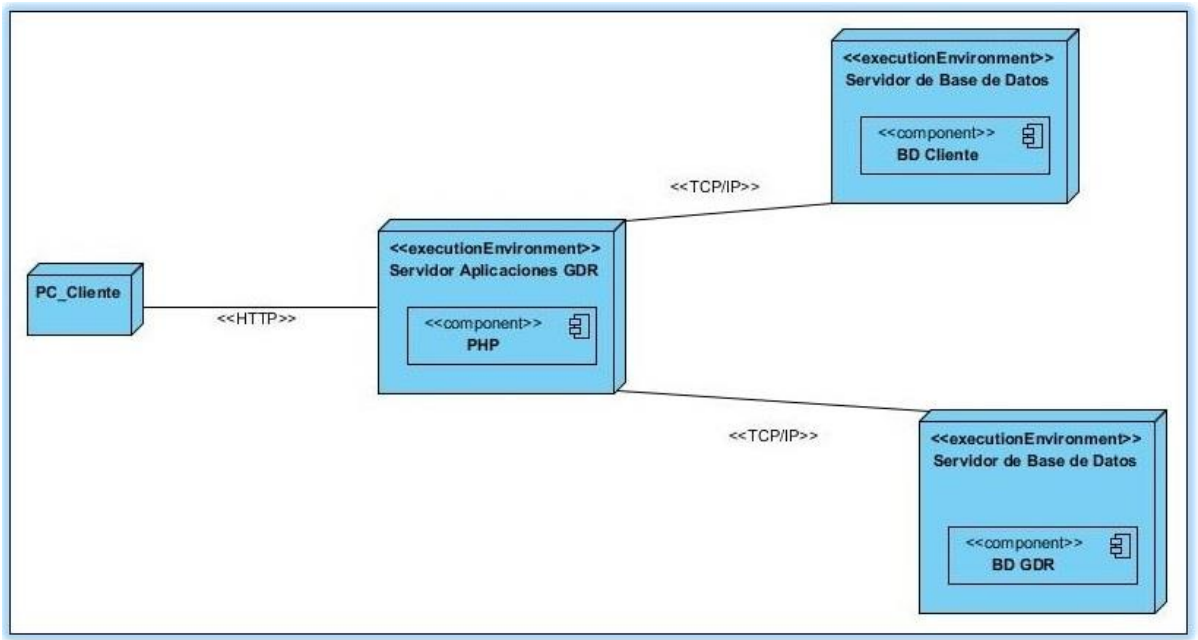


Figura 5. Diagrama de Despliegue

- **PC_Cliente**: computadora desde donde se ejecuta la url para acceder a la aplicación.
- **Servidor Aplicaciones GDR**: computadora donde está publicada la aplicación.

- **BD GDR**: contiene toda la información del GDR.
- **BD Cliente**: contiene toda la información del cliente.
- **HTTP**: se utiliza para conectar la computadora del cliente y el servidor donde está la aplicación.
- **TCP/IP**: protocolo para conectar el servidor de aplicaciones con las bases de datos.

2.8 Conclusiones parciales

En el presente capítulo se definieron tres requisitos no funcionales y diez funcionales, representando las funcionalidades que el sistema debe tener y cumplir respectivamente. Los requisitos funcionales se agruparon en dos casos de uso, lo que permitió el diseño del diagrama de casos de uso de sistema, representando así la interacción de estos con los actores definidos. Se identificaron los diferentes patrones del diseño utilizados. Y finalmente los diagramas de clases del diseño y de secuencia permitieron representar la estructura estática y dinámica del sistema respectivamente, así como el hardware utilizado a través del diagrama de despliegue.

CAPÍTULO 3:
Implementación y Pruebas

3.1 Introducción

Una vez concluida la fase de diseño, en este capítulo, se realizan las actividades que se llevan a cabo durante la fase de implementación y pruebas. Se modela el sistema en términos de componentes, se especifican los tipos, niveles, métodos y casos de pruebas que se realizarán a los componentes de conexión. Y además, se muestran ejemplos de las implementaciones más relevantes.

3.2 Modelo de implementación

El modelo de implementación, describe cómo los elementos del modelo del diseño y las clases se implementan en términos de componentes, como ficheros de código fuente y ejecutables. Describe también, como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles, en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y como dependen los componentes unos de otros. (22)

3.2.1 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes, mostrando las dependencias que existen entre ellos. Los componentes físicos incluyen: archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables y paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema; estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. 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 situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Uno de los usos principales es que pueden servir para mostrar qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. (22)

En la figura 6 se muestra, el diagrama de componentes perteneciente al CU Diseñar Modelo:

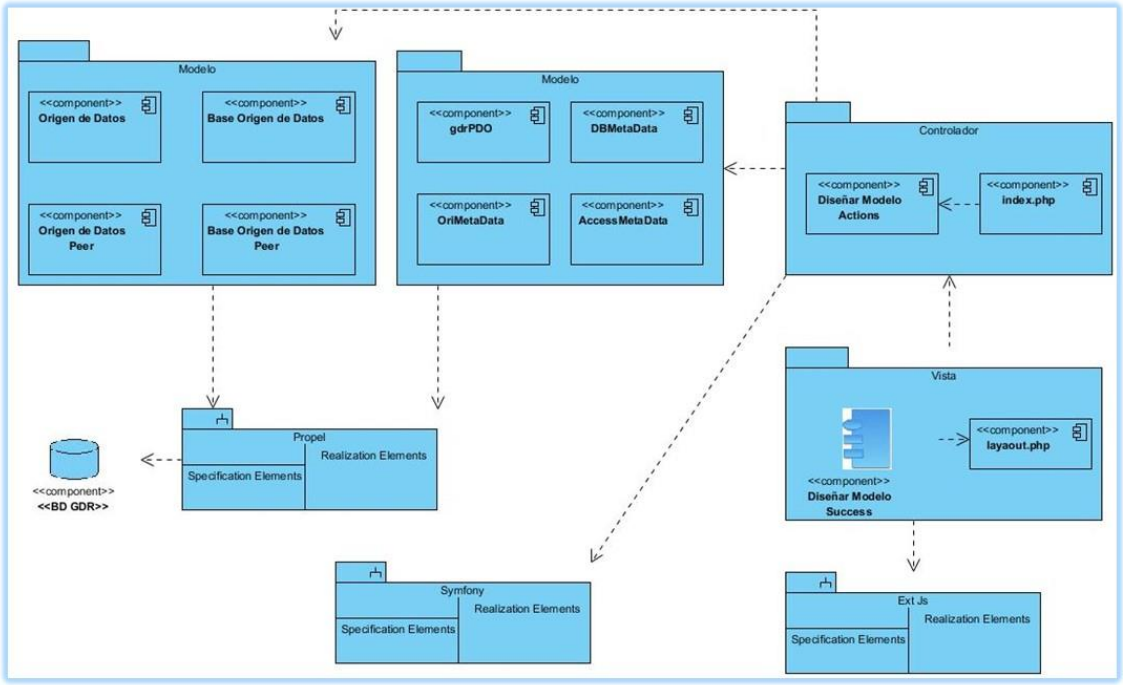


Figura 6. Diagrama de Componentes del CUS Diseñar Modelo.

En este diagrama, en el componente “Diseñar Modelo_Actions” se agrupan todas las acciones que intervienen en el caso de uso Diseñar Modelo, cada una de estas en su respectivo fichero PHP. Los componentes del modelo corresponden a las cuatro clases generadas por el subsistema Propel por cada una de las tablas de la base de datos del sistema. Mientras que en el componente “Diseñar Modelo Success” se engloban las diferentes vistas que intervienen en este caso de uso.

3.3 Código fuente

El código fuente de un programa informático es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento.

3.4.1 Estándares de codificación.

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.

Un estándar de codificación comprende todos los aspectos de la generación de código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento.

Se utilizó el estándar de codificación que define DATEC, con las siguientes especificaciones:

- El estándar de codificación será el propuesto por el IDE Netbeans 7.3 para PHP con la excepción de que deberá cambiarse la configuración de las llaves y en todos los casos estas se colocarán en una nueva línea.
- Todos los métodos, nombres de clases y variables se escribirán en estructura “camelCase”, comenzando siempre en minúscula.

3.4.2 Implementación de los componentes

A continuación se presenta algunos ejemplos con los cambios necesarios, implementados en el GDR, para la realización de la presente solución. En el módulo Diseñador de Modelos, se adicionaron las clases “AccessMetaData.php” y “Oracle11gMetaData.php” estas contienen los métodos necesarios para establecer la conexión y obtener los metadatos de los SGBD Access y Oracle11g. También se agregó la clase *“getAccessDsnAction.php”* para obtener las bases de datos existentes en el gestor Access y posteriormente listarlas, se modificó la clase de interfaz del Diseñador de Modelo *“temp.js”*, agregando los formularios para adicionar los orígenes de datos de cada uno de los gestores.

La clase *“gdrPDO.php”* también sufrió cambios agregando la configuración para la conexión con cada uno de los gestores. Con todos estos cambios se garantiza la adición de un origen de datos y posteriormente poder crear un modelo, dándole cumplimiento a los dos primeros casos de uso del sistema. Para garantizar la exportación de los reportes se modificó la clase *“Report.php”* que es la clase utilizada por el phpreport para generar reportes, en esta sufrió cambios el método *“getTabularReport”* encargado de obtener los reportes, donde se le agregó la interface para Oracle11g y Access.

A continuación se muestra un fragmento del código de los métodos utilizados para la conexión con el gestor de base de datos *Access*. (Ver Figura 7).

```
protected function AuxConfigureConnection($target, $user, $pass, $host, $port, $db) {
    if (!$this->conexion) {
        $dsn = "odbc://$user:$pass@localhost/$db";
        $this->conexion = Creole::getConnection($dsn);
    }
    return $this->conexion;
}
```

Figura 7. Ejemplo de Código Fuente para la conexión con Access.

A continuación se muestra un fragmento del código de los métodos utilizados para la obtención de los metadatos provenientes del gestor de base de datos Access. (Ver Figura 8).

```
public function queryToTablesAndViews($result) {
    $tables = array();
    while ($result->next()) {
        $table = new cbdTable();
        $schema = '';
        $tname = $result->getString('Name');
        $table->setName($tname);
        $table->setSchema($schema);
        $table->setAlias($table->getName());
        $table->setType(EntityType::$Table);
        $tables[] = $table;}
    return $tables;}
public function getFieldsByTable($tablename) {
    $user = $this->dataSource->getUsername();
    $db = $this->dataSource->getDb();
      $pass = $this->dataSource->getPassword();
    $conn = @odbc_connect($db, $user, $pw, SQL_CUR_USE_DRIVER);

    $result = @odbc_columns($conn, $db, '', $tablename);
          while (odbc_fetch_row($result)) {
        $field = new Field();
        $name = odbc_result($result, 'COLUMN_NAME');
        $type = odbc_result($result, 'DATA_TYPE');
        $is_nullable = odbc_result($result, 'NULLABLE');
        $field->setType($this->getType($type));
        $field->setName($name);
        $field->setAlias($name);
        $field->setNull($is_nullable);
        $fields[] = $field;
    }
    @odbc_free_result($result);
```

Figura 8. Ejemplo de Código Fuente Obtener de metadatos con Access.

A continuación se muestra un fragmento del código de los métodos utilizados para la obtención de las base de datos provenientes del gestor de base de datos *Access*. (Ver Figura 9).

```
class getAccessDsnAction extends sfAction {

    public function execute($request) {
        $dsn = array(
            'data' => array()
        );
        try {
            if (!file_exists('/etc/odbc.ini')) {
                return $this->renderText(json_encode($dsn));
            }
            $dsnArray = parse_ini_file('/etc/odbc.ini', true);
            foreach ($dsnArray as $index => $val) {
                $dsn['data'][] = array('dsn' => $index);
            }
            return $this->renderText(json_encode($dsn));
        } catch (Exception $exception) {
            $exceptionResult = $this->getArrayErrorDescription($exception);
            return $this->renderText(json_encode($exceptionResult));
        }
    }

}

?>
```

Figura 9. Ejemplo de Código Fuente Obtener las bases de datos en Access.

3.4 Pruebas de software

Las pruebas de software son un elemento esencial para la garantía de calidad del software y representan una revisión final de las especificaciones del diseño y de la codificación. Estas consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo. (23)

3.5.1 Nivel de Prueba

Las pruebas son aplicadas con diferentes tipos de objetivos y en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas: prueba de desarrollador, unidad, integración, sistema y aceptación. Para el desarrollo del módulo se seleccionaron las pruebas de desarrollador e integración.

3.5.2 Tipo de Pruebas

Para validar el correcto funcionamiento de un sistema se pueden realizar varios tipos de prueba, en este caso se decidió aplicar las pruebas funcionales y de integración, explicando a continuación en qué consisten cada una de ellas.

Pruebas Funcionales:

Las pruebas funcionales están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Se elaboran mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.

Pruebas de Integración:

Las pruebas de integración constituyen con conjunto de pruebas unitarias, funcionales, de regresión y aceptación que se realizan para probar el software. Enmarcan su atención en la integración de los componentes de una aplicación y su objetivo es verificar que el sistema funcione correctamente una vez integrado.

3.5.3 Método de Prueba

Existen dos métodos de pruebas para aplicar las pruebas funcionales: el método de Caja Negra y de Caja Blanca. El método de Caja negra describe las pruebas que se aplican sobre la interfaz del software utilizando los casos de prueba. Con estos, se pretende demostrar que las funciones del software son operativas, se definen las entradas al sistema y los resultados esperados de estas. Son diseñados para validar los requisitos funcionales sin detenerse en el funcionamiento interno del programa.

3.5.4 Diseño de Casos de Prueba

A continuación se muestra los datos referentes a la sección 1 de pruebas, para el CU Adicionar origen de datos.

Nombre de la Sección	Escenarios de la Sección	Descripción de la funcionalidad
SC1 "*construirOrigenDatosAction*"	SC 1.1: Adicionar origen de Datos	El diseñador de modelos decide introducir un nuevo origen de datos, el sistema envía un mensaje indicando que la petición se realizó.

Tabla 5. Sección de prueba para el caso de uso Adicionar Origen de Datos.

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

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	campo de texto	No	Es el nombre del origen de datos a registrar, para el cual solo se permiten letras, números y los siguientes caracteres:("-","_",".") entre las letras o números.
2	Tipo de gestor	campo de selección	No	Se selecciona el tipo de gestor de base de datos al cual se conectará el sistema para realizar los reportes.
3	Servidor	campo de texto	No	Es el nombre o el IP del servidor de bases de datos que representa el origen de datos, acepta cualquier tipo de carácter.
4	Puerto	campo de texto	No	Es el puerto que usa el servidor de bases de datos; se permiten números de hasta 5 dígitos en este campo y se auto completa cuando se escoge el tipo de gestor.
5	Usuario	campo de texto	No	Es el usuario para conectarse al servidor de bases de datos. Admite

				cualquier tipo de carácter.
6	Clave	campo de texto	No	Es la clave del usuario para conectarse al servidor de bases de datos. Admite cualquier tipo de carácter.
7	Base de datos	campo de selección	No	Se selecciona la base de datos asociada al servidor, admite cualquier carácter, este se usa en el caso de que el tipo de gestor sea Access.
8	Base de datos	campo de texto	No	Es el nombre de la base de datos, admite cualquier carácter, este se usa en el caso de que el tipo de gestor sea Oracle11g.

Tabla 6. Descripción de las variables

Luego de haber descrito las variables se realizó la 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. 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.

Escenario	**Variables (Enumeradas según la descripción de las variables)**							**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
	1	**2**	**3**	**4**	**5**	**6**	**7**			

Adicionar Origen de Datos	V	V	V	V	V	V	V	El sistema guarda los nuevos cambios del origen de datos en la base de datos, Enviando un mensaje al usuario.	Satisfactorio.	El sistema envía los datos codificados en formato *JSON*, retornando al cliente la respuesta de confirmación de la ejecución.
	I	V	V	V	V	V	V	El sistema devuelve un mensaje tipo *Exception* debido a que ha ocurrido un error con los datos del nuevo origen de datos.	Satisfactorio.	
Probar Conexión	V	V	V	V	V	V	V	El sistema comprueba la conexión a la base de datos.	Satisfactorio	El sistema envía los datos codificados en formato *JSON*, retornando al cliente la respuesta de confirmación de la ejecución.

Tabla 7. Matriz de Datos

3.5.5 Resultado de las Pruebas

Para validar el correcto funcionamiento del software se realizaron las pruebas de caja negra a través de los casos de pruebas basados en los casos de uso. Se ejecutó un caso de prueba y tres iteraciones, identificándose en total 13 no conformidades. La siguiente tabla muestra un resumen de las deficiencias encontradas luego de realizar el caso de prueba Adicionar Origen de datos, en el cual se verificó que el sistema permite adicionar un origen de datos, validando que sólo se acepten los caracteres válidos para cada campo.

Fecha	Iteración	Caso de Prueba	Tipo de error	Descripción
5/04/2013	1	CU Adicionar Origen de Datos	Ortografía	El campo de nombre no aceptaba las tildes.
8/04/2013	2	CU Adicionar Origen de Datos	Validación	El campo usuario y contraseña en blanco.
10/04/2013	2	CU Adicionar Origen de Datos	Programación	Error en el dsn.
11/04/2013	3	CU Adicionar Origen de Datos	--	

Tabla 8. No conformidades asociadas al CU Adicionar Origen de datos

A continuación se muestran las no conformidades que se identificaron en cada iteración, asociadas a cada caso de uso. En la primera iteración se identificaron nueve, en la segunda iteración cuatro y en la tercera iteración no se encontraron no conformidades.

Casos de Uso	Iteración 1	Iteración 2	Iteración 3
Adicionar Origen de Datos	5	3	--

Diseñar Modelo	3	1	--
Exportar Reportes	1	--	--

Tabla 9. Resumen de las no conformidades asociadas al CU Adicionar Origen de datos.

En la siguiente gráfica (ver figura 10) se observan los resultados de las pruebas basándose en la tabla anterior.

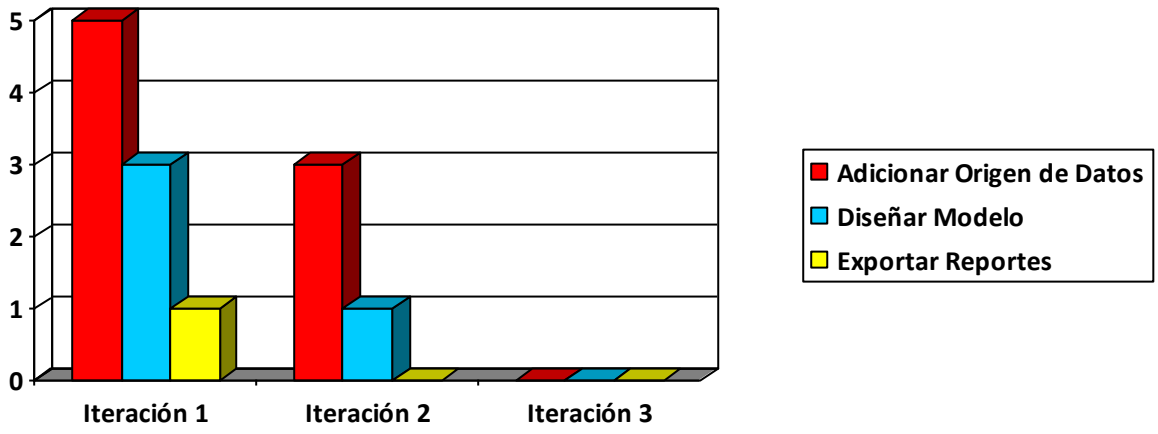


Figura 10. No conformidades asociadas al caso de uso Adicionar Origen de datos.

3.5 Conclusiones Parciales

En el presente capítulo se describió e implementó el sistema, representándolo en términos de componentes con el objetivo de lograr una mejor compresión del mismo. Se enunció el estándar de codificación definido para el desarrollo del sistema, mostrando el uso del mismo a través de imágenes con el código de las principales funcionalidades de la aplicación. La realización de las pruebas funcionales (método de Caja Negra) y de integración permitieron confirmar la correcta definición e implementación de los requisitos.

CONCLUSIONES

- A partir de las funcionalidades descritas se realizó el análisis y el modelo de diseño de los componentes, proporcionando el punto de partida para las actividades de implementación.

- La correcta especificación de los requerimientos y la realización del diseño del sistema permitieron implementar los componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g.

- Se probó el adecuado funcionamiento del módulo a través de las pruebas de caja negra realizadas mediante los casos de prueba.

- Como resultado del trabajo realizado se obtuvieron los componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g.

RECOMENDACIONES

Luego del cumplimiento de los objetivos trazados, se recomienda:

- Permitir la conexión del GDR con bases de datos no relacionales o NoSQL.

REFERENCIAS BIBLIOGRÁFICAS

1. **Cobo, Angel.** *Diseño de programacion de base de datos. Didactica escolar.* s.l. : Vision Libros, 2007. 8498214599.

2. *Informatica Para Las Oposiciones a la Comunidad Autonoma de Las Islas Baleares. Temario Comun Y Test Ebook.* s.l. : MAD-Eduforma, 2007. 8466506756, 9788466506755.

3. **Oracle Enterprise Inc.** Products and Services | Oracle. [En línea] Oracle Inc. [Citado el: 11 de 12 de 2012.] http://www.oracle.com/us/products/index.html.

4. **Hernandez, Yasmany.** *Propuesta de Arquitectura de para Sistema Generador de Reportes. .* 2009.

5. **Lafosse, Jérôme.** *Expert IT Struts 2 - El framework de desarrollo de aplicaciones Java EE.* s.l. : Ediciones ENI, 2010.

6. **Eguiluz, Javier.** *Desarrollo Web Agil con Symfony.*

7. **Frederick, Shea.** *Learning Ext Js.* s.l. : Packt Publishing, 2008. 978-1-847195-14-2.

8. **Bell, Douglas.** *Java para estudiantes.* s.l. : Pearson Educación, 2007. 9702601444.

9. **EcuRed.** EcuRed. [En línea] [Citado el: 9 de 1 de 2013.] http://www.ecured.cu/index.php/NetBeans.

10. **Roque, E.Garcia Cuevas.** *Principios Básicos.* s.l. : Libreria-Editorial Dykinson, 2007. 8498490987.

11.**Solis, Santiago Marquez.** *La Web Semantica.* s.l. : Lulu.com, 2007. 184753192X, 9781847531926.

12. **Pérez, Javier Eguíluz.** *Introducion a JavaScript.* 2011.

13. **Navarra, Pablo Lara.** *Organizacion del conocimiento en intern et.* s.l. : UOC. 8497885422, 9788497885423.

14. **Born, Gunter.** *Compendium HTML: con XHTML, DHTML, CSS, XML, XSL y WML.* s.l. : Morcombo, 2001. 8426713084, 9788426713087.

15. **Areba, Jesus Barranco de.** *Metodología Del Análisis Estructurado de Sistemas.* s.l. : Univ Pontificia Comillas, 2002. 8484680436, 9788484680437.

16. **M, Harvey.** *Como programar en Java.* s.l. : Pearson Educación, 2004. 97026005180,9789702605188.

17. **Rondon, Yoendry Quintana.** *Diseño de Base de Datos para Sistemas de Digitalizacion y Gestion de Medias.* habana cuba : s.n., 2011. 1667-8338.

18. . **Hernandez, Amabely Silva** *Generador Automatico de Reportes Dinamicos.***.** 2003.

19. http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.

20. **Harvey M. Deitel, Paul J. Deitel**- *Cómo programar en Java.* Pearson Educación, 2004. 9702605180, 9789702605188

21. **Benet Campderrich Falgueras.** *Ingeniería de software Biblioteca multimedia: Informática,* Editorial UOC, 2003, 8484297934, 9788484297932

22. **EDWARD FRASER, DUSTIN**. Scribd. [En línea] [Citado el: 5 de Abril de 2012.] http://es.scribd.com/doc/74894700/44/CAPITULO-4-IMPLEMENTACION-Y-PRUEBA-DEL-SISTEMA.

23. **Fernando Alonso Amo, Loïc A. Martínez Normand, Francisco Javier Segovia Pérez.** *Introducción a la ingeniería del software,* Delta Publicaciones, 2005

24. M.C. Laura Cecilia Méndez G. scribd. TIPOS DE PRUBAS DEL SOFTWARE. [En línea] [Citado el: 10 de 5 de 2013.] http://es.scribd.com/doc/51838546/TIPOS-DE-PRUEBAS-DE-SOFTW

BIBLIOGRAFÍA

1. **JACOBSON, I.; G. BOOCH**, et al. *El proceso Unificado de Desarrollo de Software*. p.xviii Addison Wesley Object Technology.
2. **LOBO, Armando Robert.** *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas*. Ciudad Habana: Trabajo de Diploma, 2009.
3. **PRESMAN, Roger S**.; 2007. *Ingeniería de software. Un enfoque práctico*. 6ta Edición. Parte I. Prólogo y Capítulos 1, 2 y 21. Páginas 1-18, 22-45 y 641-657.
4. **POTENCIER, Fabien y François Zaninotto.** *Symfony La Guía Definitiva.* 30 de diciembre de 2008.
5. **Sanchez, Linet Lores y Roque, Diana Monné.** *Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica.* Ciudad de la Habana, Junio 2009. Trabajo de Diploma para optar por el título de Ingeniero Informático.
6. **TELLO, Jesús Cáceres.** *Diagramas de Casos de Uso*. Universidad de Alcalá Departamento de Ciencias de Computación: s.n.
7. **Kernighan, Brian W, Ritchie, Dennis M. y Gómez Muñoz, Néstor.** Lenguajes de programación . [En línea] 1991. books.google.com.
8. **Tecnología y Synergix. [En línea] 2009. [Citado el: 2013 de Febrero de 16.]**
9. **GB Hall, JP Alperin, SK LEÓN.** GeoFocus. [En línea] 2008 . geofocus.rediris.es.
10. PgAdmin_III. [En línea] guia-ubuntu.org.
11. **I Arzuza, M Hernández, F Ortiz, C Rodríguez.** Symfony. [En línea] unisimonbolivar.edu.co.
12. **S Pérez Lovelle, F Orejas Valdés.** Revista Ingeniería. [En línea] 2010 . rii.cujae.edu.cu.
13. **SL Mora, SL Mora, MZ Fornieles, SL Mora.** El futuro de Internet:'XM. *Los secretos xml.* [En línea] dlsi.ua.es.
14. **Hernández Hernández, Yasmany**. Arquitectura de Software para el Sistema de Gestión de
15. Reportes Dinámicos. La Habana : s.n., 2009.
16. **Grupo de desarrollo del generador dinámico de reportes**. Pautas para el desarrollo basado en componentes para la capa de presentación.
17. **Grupo de desarrollo del generador dinámico de reportes**. Pautas para la Internalización del
18. Generador Dinámico de Reportes.

19. **Hernández Hernández, Yasmany**. Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos. Tesis La Habana. Universidad de las Ciencias Informáticas, 2009.
20. netbeans.org. netbeans.org. [En línea] [Citado el: 2 de diciembre de 2012]. Disponible en: <www.netbeans.org>
21. pgadmin.org. pgadmin.org. [En línea] [Citado el: 3 de diciembre de 2012]. Disponible en: <http://www.pgadmin.org/index.php

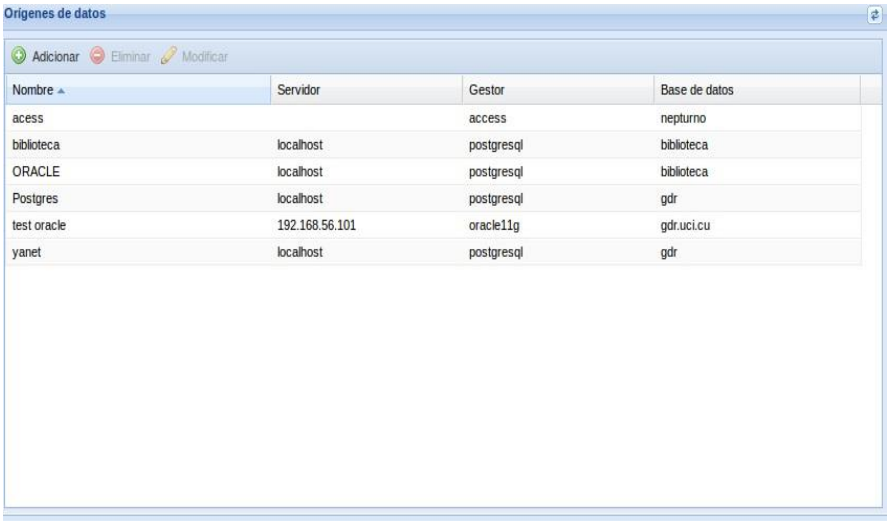
ANEXOS

Anexo 1: Interfaz de la aplicación. Adicionar Origen de datos de Oracle11g.

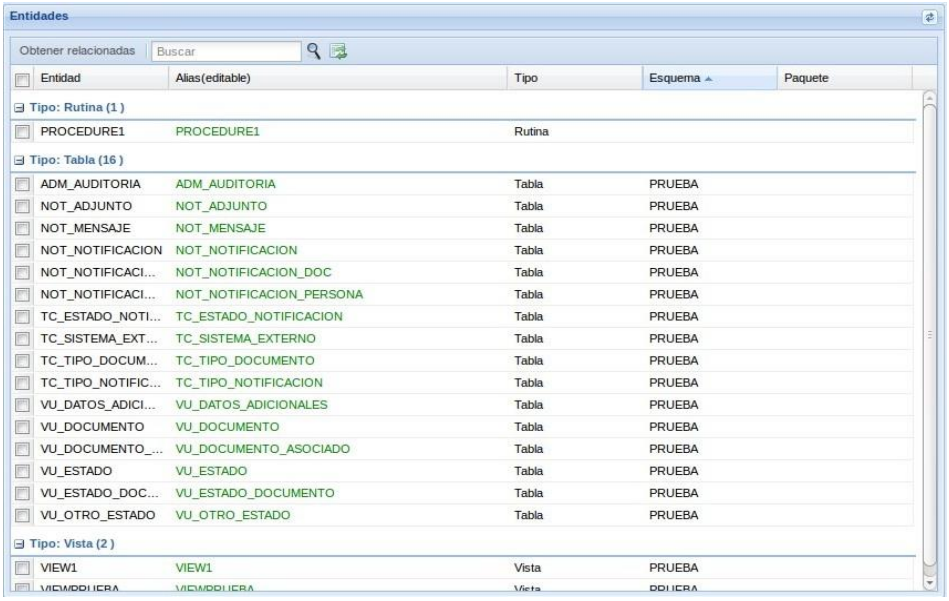
Anexo 2: Interfaz de la aplicación. Conexión no válida.

Anexo 3: Interfaz de la aplicación. Conexión funcionando correctamente.

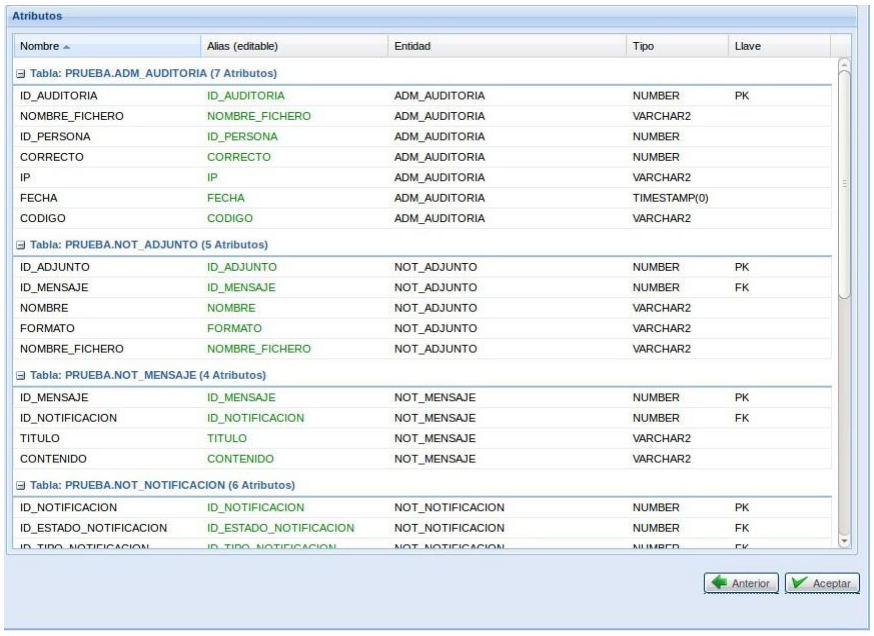
Anexo 4: Interfaz de la aplicación. Lista de orígenes de datos existentes.



Anexo 5: Interfaz de la aplicación. Listado de las entidades del origen de datos

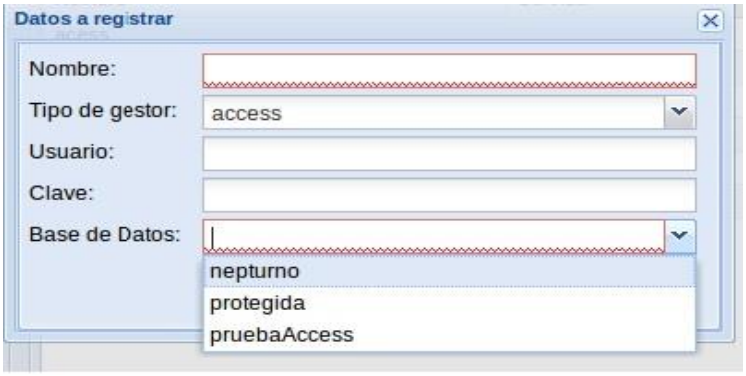


Anexo 6: Interfaz de la aplicación. Listado de los atributos de los orígenes de datos.



Anexo 7: Interfaz de la aplicación. Datos a registrar.

Anexo 8: Interfaz de la aplicación. Listar Base de Datos Access.



Anexo 9: Interfaz de la aplicación. Formulario Access.

Anexo 10: Interfaz de la aplicación. Formulario Access.

GLOSARIO DE TÉRMINOS

AJAX: Aplicación interactiva para páginas web.

CASE: Ingeniería de Software Asistida por Computación.

DATEC: Centro de Tecnologías de Gestión de Datos.

DHTML: Idioma de hipertexto dinámico, compuesto de opciones agregados en los documentos HTML que permite la gráfica y la interacción inmediata con el usuario

DOM: Define qué atributos son asociados con cada objeto y cómo los objetos y los atributos pueden ser manipulados.

DTD: Descripción de la estructura y sintaxis de un documento XML.

GDR: Generador Dinámico de Reportes.

GRASP: Patrones de Software para la Asignación General de Responsabilidad.

IDE: Entorno Desarrollo Integrado.

JSON: Notación de Objetos de JavaScript.

MVC: Modelo – Vista – Controlador.

OpenUp: Metodología para el proceso del desarrollo del software.

ORM: Mapeo de Objetos a Bases de Datos.

ORM: Object Relational Mappe, es una técnica de programación que permite generar clases a través de las tablas de la base de datos.

RUP: Proceso de ingeniería de software orientado a objetos.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje Unificado de Modelado.