Universidad de las Ciencias Informáticas

Facultad 6

Título: “Diseño e Implementación del módulo Diseñador de Consultas del Generador Dinámico de Reportes”

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

Autor (es): Orelvi Gazquez Martínez

Yadrian Serrano García.

.

Tutor (es): Ing. Jose Rolando Lafaurie Olivares.

Ing. Yoander Iñiguez Bermúdez.

Junio, 2011

Declaración de Autoría.

Declaramos que somos autores de la presente tesis y reconocemos, a la Universidad de las Ciencias Informáticas, sus derechos patrimoniales sobre la misma con carácter exclusivo.

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

_____	_____
Orelvi Gazquez Martínez.	Yadrian Serrano García.
Autor	**Autor**
_____	_____
Ing. Jose Rolando Lafaurie Olivares.	Ing. Yoander Iñiguez Bermúdez.
Tutor	**Tutor**

Datos de Contacto

Tutores:

Jose Rolando Lafaurie Olivares: Ingeniero en Ciencias Informáticas, Profesor Instructor con 2 años de experiencia.
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba
Email: jrlafaurie@uci.cu

Yoander Iñiguez Bermúdez: Ingeniero en Ciencias Informáticas, Profesor Instructor con 1 año de experiencia.
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba
Email: yiniguez@uci.cu

Agradecimientos

Agradezco primeramente a Dios por haberme dado el privilegio de conocerle formando en mí una nueva persona; por cuidarme y guiarme en todo momento, por darme sabiduría y entendimiento; por estar siempre conmigo.

A mi madre Sonia Marlene García Pupo y a mi padre Lorenzo Fermín Serrano Rondón, por su amor y cariño, por su apoyo en todas mis decisiones, por el esfuerzo y trabajo con el cual me han criado haciendo de mi lo que ahora soy.

A mi esposa por su agradable compañía.

A mi hermana melliza Yariannis, a mis otros hermanos: Yudania, Yanicet, Danilo y Aniuska. A mis abuelos y al resto de mi familia por su incondicional apoyo y preocupación.

A mis suegros que ya son parte de mi familia.

A mis amigos, los de antes, los de la vocacional; los nuevos, que he cosechado aquí; de manera especial a Orelvi que más que un amigo es para mí un hermano, igualmente a Amed, Roberto y Luis.
A mis hermanos en la fe, los contemporáneos, a los que recién comienzan y a los que ya no están.

A mis compañeros de grupo y de apartamento que hemos navegado en

el mismo barco y aunque muchos quedaron en el camino la mayoría llegamos a tierra firme.

A mis dos tutores Yoander y Rolando por guiarnos y apoyarnos. Al profe Miguel del centro DATEC, a Aldi, en fin a todos aquellos que de una manera u otra influyeron positivamente en el desarrollo de la tesis así como en la estancia en esta universidad.

Yadrian Serrano García.

En primer lugar quiero agradecer a mi Dios, ya que él ha sido mi principal fuente de sustento en estos cinco años de universidad. Caminando a razón de él he aprendido que cuando David dijo: "aunque mi padre y mi madre me dejaran, con todo, Jehová me recogerá", no lo decía en sentido de metáfora.

A mis padres por todo el esfuerzo que han hecho a lo largo de estos 12 años en que he estado becado, por la educación y el amor que me han dado desde los primeros años de mi vida. De ellos he aprendido que un hijo vale más que todas las cosas que existen en la tierra.

A mi novia, por estar siempre presta a escucharme, por ser mi paño de lágrimas en los momentos más difíciles y por ser la solución en muchos de estos momentos. De ella he aprendido la segunda mayor lección de amor de mi vida.

A mi abuelo, por su apoyo incondicional, aun cuando no tengo la razón, por soportarme todas mis pesadeces y por su inmenso amor. De él he aprendido que aunque hay cosas importantes, hay otras que lo son aún más.

A mi hermano Yadrian, por su ayuda a lo largo de estos cinco años, pues creo que él ha sido el principal instrumento que Dios ha usado para trasmitirme conocimiento, conocimiento sin el cual no estaría hoy redactando en este documento. De él he aprendido que aunque hay cosas buenas, hay otras que son mucho mejores, y esas solo se consiguen a través de sacrificio.

A todos mis hermanos en Cristo, a Armis por soportar también mis pesadeces, a mis amigos, a todos los que me han ayudado en mi trabajo de tesis, a mis profesores del proyecto, a mis tutores y en especial al profesor Armando Robert Lobo. A todos Dios los ha utilizado, influyendo esto de manera especial en mi vida.

Orelvi Gazquez Martinez.

Dedicatoria

Dedicamos este trabajo a nuestros padres por su esfuerzo y sacrificio en nuestros 24 años de edad y 18 años de estudio. A nuestros hermanos y a toda nuestra familia por su apoyo y preocupación.

Resumen

El Centro de Tecnologías y Gestión de Datos (DATEC), se dedica al desarrollo de sistemas informáticos encaminados a satisfacer las necesidades de gestión de la información. Entre estos sistemas se encuentra el proyecto Generador Dinámico de Reportes (GDR), que a su vez cuenta con el módulo Diseñador de Consulta. Este funciona como un editor gráfico de consultas SQL. Pero a medida de que se le incorporan nuevas funcionalidades han ido siendo evidentes los problemas arquitectónicos que posee su diseño de clases, la incorrecta separación de responsabilidades que este posee, así como la ausencia de algún tipo de documentación. En función de dar solución a esta problemática, se realiza un estudio de los principales sistemas generadores de reportes que existen en la actualidad, haciendo énfasis en la relevancia que tiene la inclusión del módulo diseñador de consultas en el GDR. Se elige la metodología de desarrollo a utilizar y se presentan las principales técnicas y tecnologías usadas para la solución del problema. Se eligen los patrones GRASP como patrones de diseño y el patrón Modelo – Vista – Controlador como patrón de arquitectura. Partiendo del resultado del diseño, se modela el sistema en términos de componentes, se dan a conocer las técnicas usadas en la solución del problema y se aplican pruebas funcionales de tipo caja negra. Finalmente se obtiene como resultado del trabajo de diploma, el Diseñador de Consultas del GDR, haciendo uso de buenos patrones de diseño y aplicando la arquitectura de referencia de la línea.

Palabras Clave

base de datos, entidad, generador de reportes, intérprete

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Introducción

La información está destinada a resolver determinados problemas, sirviendo para el desarrollo individual y corporativo, estando presente en todos los niveles de actividad y ramas de la economía, la política y la sociedad. La gestión de información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de una organización. Para tomar una decisión, sin importar su naturaleza, es necesario conocer, comprender, analizar un problema, para así poder darle solución. En algunos casos por ser tan simples y cotidianos, este proceso se realiza de forma implícita y se soluciona muy rápidamente. Pero existen otros casos en los cuales las consecuencias de una mala o buena elección pueden tener repercusiones en la vida y si es en un contexto laboral, en el éxito o fracaso de la organización. Para estos casos es necesario realizar un proceso más estructurado, que permita dar seguridad e información para resolver el problema.

La línea de Soluciones Integrales del Centro de Tecnologías y Gestión de Datos (DATEC), perteneciente a la infraestructura productiva de la Universidad de las Ciencias Informáticas (UCI), se dedica al desarrollo de sistemas informáticos encaminados a satisfacer las necesidades de gestión de la información existentes en una empresa, una institución o un país en general. Basándose en esto y con el fin de elevar la eficacia, eficiencia y calidad de la gestión pública nacional unido al desarrollo socioeconómico de las naciones, diferentes organizaciones entre las que se destaca la Oficina Nacional de Estadística de Cuba (ONE), han destinado presupuestos, recursos y capital humano en planes y proyectos capaces de medir el desarrollo económico y social. Algunos de estos proyectos consisten en sistemas informáticos con una serie de herramientas, que basándose en métodos estadísticos, brindarán una mayor eficacia en el proceso de toma de decisiones, por lo que se ha elegido la línea de Soluciones Integrales de DATEC para que desarrolle dichos sistemas informáticos.

Entre estos sistemas informáticos se encuentra el activo Generador Dinámico de Reportes (GDR), este proyecto brinda la posibilidad de chequear el funcionamiento periódico de una o varias entidades, mediante la formulación de reportes en diferentes formatos y modelos personalizables, obtenidos a través de datos estadísticos existentes en las bases de datos, que en este se pueden importar.

El Generador Dinámico de Reportes a su vez cuenta con el módulo Diseñador de Consultas, que funciona como un editor gráfico de consultas SQL, permitiendo crear, editar y ejecutar consultas del tipo antes mencionado, sin necesidad de conocer el gestor de base de datos ni el lenguaje de consulta estructurado (SQL). Este Diseñador de Consultas ha sido pensado con el objetivo de satisfacer a un usuario que desee obtener información estadística sobre los resultados obtenidos cuando dos

entidades se relacionan entre si y su funcionamiento es en conjunto, brindando información sobre el comportamiento de las empresas u organizaciones que funcionan una dependiendo de la otra.
El Centro de Tecnologías y Gestión de Datos ya cuenta con un Diseñador de Consultas en la versión actual del Generador Dinámico de Reportes, pero a medida que se le incorporan nuevas funcionalidades han ido siendo más evidentes los problemas de integración que posee su diseño de clases, basado en una arquitectura que no está alineada a la arquitectura de referencia de la línea y que tampoco cumple con patrones de diseño. Su código fuente presenta grandes problemas entre los que se destacan: el poco aprovechamiento de las características de la programación orientada a objeto (como la herencia y el polimorfismo), existiendo innecesariamente códigos repetidos; los nombres de las clases y las variables usadas en estas son poco o nada sugerentes, no hay una correcta separación de responsabilidades; además de toda la desorganización de código, no existe ningún tipo de documentación. Todos estos problemas influyen en el rendimiento de la aplicación, así como en el mantenimiento de la misma, imposibilitando de manera considerable la integración de nuevas funcionalidades por la dificultad de interpretación y manipulación de código.

A partir de la problemática planteada se genera el **Problema Científico:** ¿Cómo facilitar la construcción, edición y ejecución de consultas SQL de forma gráfica?

El problema de la investigación se enmarca en el **Objeto de Estudio:** Proceso de gestión de información en los sistemas generadores de reportes.

El objeto de estudio está delimitado por el **Campo de Acción**: Módulo Diseñador de Consultas en del Generador Dinámico de Reportes.

En función de dar solución al problema planteado se traza como **Objetivo General:** Realizar el Diseño e Implementación del módulo Diseñador de Consultas del Generador Dinámico de Reportes para facilitar la construcción, edición y ejecución de consultas SQL de forma gráfica.

Objetivos Específicos

- Diseñar el módulo Diseñador de Consultas del GDR.
- Implementar el módulo Diseñador de Consultas del GDR.
- Integrar un intérprete de SQL al módulo Diseñador de Consultas que permita la edición de consultas manualmente.
- Realizar pruebas exploratorias al módulo.

Idea a Defender: Si se implementa el módulo Diseñador de Consultas del GDR utilizando buenos patrones de diseño, siguiendo la arquitectura de referencia de la línea de desarrollo y una apropiada

separación de responsabilidades, aumentará el rendimiento de la aplicación, resultando más sencillo el mantenimiento de la misma así como la incorporación de nuevas funcionalidades.

Tareas de la Investigación

- Diseño del modelo de clases que atenderán las peticiones del módulo Diseñador de Consultas del GDR en el lado del servidor.
- Diseño del modelo de clases que manejarán las interacciones de los usuarios con la aplicación.
- Implementación del modelo de clases que atenderán las peticiones del módulo Diseñador de Consultas del GDR en el lado del servidor.
- Implementación del modelo de clases que manejarán las interacciones de los usuarios con la aplicación.
- Integración de un intérprete de SQL que permita la edición de consultas manualmente.
- Realización de pruebas funcionales al módulo de tipo caja negra.

Métodos Científicos de la Investigación

Teóricos:

- **Analítico – Sintético:** Se utilizará para el estudio de las técnicas de interpretación y compilación, conjuntamente con las tecnologías java script.
- **Histórico – Lógico:** Con el fin de estudiar los sistemas más actuales que tienen incorporado un Diseñador de Consultas.

Empíricos:

- **Método experimental:** Se utiliza para hacer las pruebas al producto con el fin de verificar su estado**.**

El trabajo está estructurado en tres capítulos:

Capítulo 1. Marco Conceptual: Este capítulo presenta un estudio de las tecnologías a utilizar, así como de las tendencias, técnicas, metodologías y software usados para dar solución al problema.

Capítulo 2. Diseño de la Solución: Se presenta una vista abstracta del diseño del sistema, utilizando diagramas de diseño.

Capítulo 3. Implementación y Prueba: Se presenta una vista abstracta del funcionamiento del sistema y se especifican los casos de pruebas realizadas al módulo.

Capítulo 1: Marco Conceptual

Introducción

En el presente capítulo se hace un estudio de los principales sistemas generadores de reportes que existen en la actualidad, haciendo énfasis en la relevancia que tiene la inclusión del módulo Diseñador de Consulta en el GDR. Se selecciona la metodología de desarrollo a utilizar, se proponen los patrones de software para el diseño de la solución, y se describen las principales tendencias, técnicas y tecnologías usadas en la actualidad, en las que se apoya para la solución del problema que se enfrenta.

Generadores de Reportes

Los generadores de reportes son herramientas complementarias de los sistemas de información, que utilizan una especie de lenguaje transparente para el usuario, por medio del cual se realizan consultas a la base de datos, obteniendo información de ella en forma de reporte. Estos ofrecen a los trabajadores calificados un mejor nivel de detalle y flexibilidad de los datos, además de la capacidad de interactuar con los resultados obtenidos, basándose en estos para tomar sus propias decisiones. [5]

Sistemas Generadores de Reportes en el Mundo

Stimulsoft Reports.Web

Stimulsoft Reports.Web es una herramienta de presentación de informes diseñada para crear y hacer reportes web. Puede crear, mostrar, exportar e imprimir reportes extrayendo datos de MS-SQL, Oracle, PostgreSQL, SQLCE, SQLite entre otros. Stimulsoft Reports.Web proporciona el ciclo completo de elaboración de reportes, comenzando con plantillas de reportes y finalizando con la visualización de estos en un explorador web. No hay necesidad de instalar el framework .NET, componentes ActiveX u otros complementos adicionales en la computadora del cliente, todo lo que necesita es un explorador web con Flash Player 9 o superior. [1]

Crystal Reports

Crystal Reports es un producto de alta tecnología para la creación e integración de reportes con datos provenientes de múltiples fuentes de datos entre las que se encuentran: PostgreSQL, MySQL, Oracle, DB2, MS-SQL, Informix, InterBase, Sybase y Frontbase. Seleccionado por más de 300 socios de negocios por su alta tecnología de elaboración de reportes, es un líder comprobado en el diseño de reportes que cumplan los desafíos que día a día enfrentan los analistas de negocios y los desarrolladores, ya que permite transformar rápidamente cualquier fuente de datos en contenido

interactivo e integrar estrechamente capacidades de diseño, modificación y visualización en aplicaciones .NET o Java. Permitiendo a los usuarios finales acceder e interactuar con los reportes a través de portales Web, dispositivos móviles y documentos de Microsoft Office. [2]

Smart Report Maker

Smart Report Maker es un Generador de Reportes PHP/MySQL de muy fácil uso, que permite crear y administrar un número ilimitado de Reportes MySQL basados en tablas o consultas. Cuenta con una interface de asistente amigable que permite seleccionar únicamente los campos que se desean en el reporte, configura múltiples niveles de agrupamiento, ordena los datos de forma ascendente o descendente. Tiene la capacidad de enviar los reportes generados con mensajes personales por correo electrónico dentro de un script. Los reportes generados se despliegan en el navegador y también pueden ser almacenados en un directorio, en su servidor, esto permite crear enlaces o marcar los reportes generados para su uso futuro (los reportes generados serán actualizados automáticamente, siempre y cuando se agreguen nuevos registros a su base de datos). La apariencia de los reportes generados es totalmente personalizable, de tal forma que se elige el diseño y el estilo de hoja. [3]

Agata Report

Agata Report es un conjunto de herramientas para sacar lo más sustantivo de las bases de datos, permitiendo extraer datos de PostgreSQL, MySQL, Oracle, DB2, MS-SQL, Informix, InterBase, Sybase y Frontbase; entre muchas funcionalidades interesantes, proporciona un módulo para crear gráficas en curvas o en bastones, permite cruzar informes con otras bases de datos y soporta parámetros temporales de ejecución. Agata Report es una herramienta estable y adaptada a las necesidades de aquellos que manipulan datos con frecuencia, con una interfaz intuitiva y disponible en Inglés, Portugués, Español, Italiano, Francés, Alemán y Sueco; permite su uso a cualquier persona con unas pocas nociones de SQL. Bajo licencia GNU, Agata Report funciona tanto para Linux como para Microsoft Windows. [4]

Active Report

Active Report es una herramienta de plataforma propietaria que se caracteriza por soportar gráficos de tipo 2D y 3D, imágenes y sub-reportes. Es capaz de emplear una amplia variedad de orígenes de datos, además cuenta con un lenguaje de programación propio. Permite la adición de nuevos reportes sin necesidad de recompilar la aplicación, dando soporte tanto a aplicaciones web como de escritorio y exporta los reportes en diferentes formatos como: PDF, XML y HTML. [5]

Sistemas Generadores de Reportes en la Universidad

Generador Dinámico de Reportes (GDR)

El Generador Dinámico de Reportes de DATEC es una herramienta multiplataforma desarrollada con tecnologías web que permite la creación y edición de informes/reportes extrayendo datos de una amplia gama de gestores de bases de datos, entre los que se encuentran PostgreSQL, MySQL, Oracle, SQLite y MS-SQL. Los reportes son creados siguiendo el siguiente estándar: encabezado del documento, encabezado de página, cuerpo, pie del documento y pie de página. El GDR permite cargar en una vista estándar los modelos de bases de datos con los que se desea trabajar a través de una interfaz amigable, creada con el objetivo de brindar la mayor cantidad de opciones posibles para el diseño de los reportes, estos pueden ser personalizados dependiendo de las necesidades del informe deseado, brindando la posibilidad de agregar gráficas e imágenes que muestren una representación clara de los datos a valorar. Lo que hace especial al GDR es la inclusión del módulo Diseñador de Consultas ya que este permite crear, editar y ejecutar consultas SQL de forma gráfica, mostrando el comportamiento de las entidades en el momento en que se relacionan unas con otras, esto crea una gran demanda por parte de las empresas y organizaciones interesadas en el producto, ya que muchas dependen del desarrollo de pequeñas corporaciones o de diferentes organizaciones.

Akademos

Akademos es una aplicación desarrollada en plataforma .NET, en el 2006, en la Universidad de las Ciencias Informáticas, la cual brinda actualmente importantes servicios académicos tanto a los estudiantes como a los departamentos docentes. Cuenta con una herramienta de generación de reportes, la cual los genera con eficiencia, pero desde el punto de vista del dinamismo de la generación, se encuentra adaptada solamente al negocio de la gestión académica en la UCI, restringiendo que sean elaborados con la información contenida en la base de datos específicamente utilizada por el software, lo cual dificulta que esta herramienta sea de propósito general. Además, está diseñada sólo para la Web. [5]

Los sistemas generadores de reportes presentados con anterioridad, poseen diferentes particularidades que pueden ser usadas o desechadas en dependencia del entorno de trabajo en que el sistema desempeñe su rol, pero el Generador Dinámico de Reportes, incluye el módulo Diseñador de Consultas, brindando con esto la posibilidad de obtener datos estadísticos de forma rápida, agilizando así el proceso de la toma de decisiones.

Diseñadores de Consultas

Los Diseñadores de Consultas son herramientas que facilitan la visualización de los datos existentes en una base de datos mediante la construcción, edición y ejecución de consultas SQL de forma gráfica. Las entidades almacenadas se muestran en un área de diseño brindando la posibilidad de seleccionar y relacionar los campos de los cuales se desea obtener información, posteriormente los datos son mostrados en forma de tabla. Generalmente estos sistemas se desarrollan para satisfacer a un usuario con poco dominio del lenguaje de consulta estructurado.

Diseñador de Consultas del GDR

El Diseñador de Consultas del GDR es una herramienta para proporcionar versatilidad en el proceso de visualización de resultados estadísticos, encaminados a satisfacer las necesidades de gestión de información que posea una empresa, una organización o un país; permitiendo conjugar los resultados obtenidos entre varias entidades que se relacionan. La rapidez que este proporciona en cuanto a obtención de datos agilizará el proceso de toma de decisiones, brindando la posibilidad de guardar consultas previamente diseñadas con el objetivo de reutilizarlas en el futuro y generar reportes basados en estas sin necesidad de ser diseñadas nuevamente.

Intérprete SQL

En ciencias de la computación intérprete o interpretador es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel. En el presente trabajo se utiliza dicha herramienta con la intención de satisfacer a un usuario con conocimientos en el lenguaje de consulta estructurado, capaz de modificar el código de una consulta, el intérprete facilitará los recursos necesarios para que dicha consulta se grafique correctamente. Implementar un intérprete utilizando un generador como JavaCC , ANTLR o LPG, es una opción bastante costosa, debido a la complejidad de dichos generadores, teniendo en cuenta que el proyecto no se basa solamente en esta tarea, se decidió usar un intérprete ya existente y de Licencia Pública General Reducida (LGPL).

JSqlParser

JSqlParser es un proyecto de software libre bajo licencia LGPL, basado en un intérprete generado con JavaCC. Posee una documentación aceptable, además cuenta con la ayuda permanente de un foro que permitirá dar respuestas a las dudas que puedan surgir. Es útil tanto para analizar como para generar consultas SQL.

Metodología de Desarrollo

Una metodología de desarrollo de software indica paso a paso todas las actividades a realizar en

función de lograr un producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben tener; detalla la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. [5]
La metodología escogida para el desarrollo del trabajo es: Proceso Unificado de Desarrollo de Software (RUP), según la arquitectura definida por el proyecto.

RUP

RUP es un proceso de desarrollo de software (conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema de software), un marco genérico que puede especializarse para una variedad de tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y diferentes tamaños de proyectos. RUP está basado en componentes, dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental.

Roles Desempeñados

De acorde a la metodología seleccionada para el diseño e implementación de la solución del sistema, se definieron los roles a ser desempeñados:

Arquitecto de Software

Este rol dirige el desarrollo de la arquitectura de software del sistema, que incluye la promoción y la creación de soporte para las decisiones técnicas claves que restringen el diseño global y la implementación del proyecto. El arquitecto de software tiene la responsabilidad global de dirigir las principales decisiones técnicas, expresadas como la arquitectura de software; esto habitualmente incluye la identificación y la documentación de los aspectos arquitectónicamente significativos del sistema, que incluye las vistas de requisitos, diseño, implementación y despliegue del sistema. El arquitecto también es responsable de proporcionar el fundamento de estas decisiones, equilibrando las preocupaciones de los diferentes interesados, reduciendo los riesgos técnicos, y garantizando que las decisiones se comuniquen, se validen con eficacia y que se acaten. [6]

Diseñador

Este rol dirige el diseño de una parte del sistema, dentro de las restricciones de los requisitos, arquitectura y proceso de desarrollo para el proyecto. El diseñador identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño; se asegura de que el diseño sea coherente con la arquitectura de software y que esté detallado hasta un punto en que pueda proceder la implementación. [6]

Implementador

Este rol desarrolla los componentes de software y efectúa las pruebas de desarrollador para la integración en subsistemas más grandes de acuerdo con los estándares adoptados por el proyecto. Cuando los componentes de prueba, como controladores o fragmentos para simulación deben crearse para dar soporte a las pruebas, el implementador también es responsable de su desarrollo. [6]

Artefactos generados

En dependencia de la metodología y los roles definidos para el desarrollo de la solución del sistema se generan los siguientes artefactos:

Artefactos a ser generados por el Arquitecto de Software

Arquitectura del sistema

Este artefacto proporciona una visión general de la arquitectura del sistema, mediante una serie de vistas arquitectónicas que representan diferentes aspectos del sistema. En el presente trabajo se proporcionarán las siguientes vistas: Vista Lógica, Vista de Despliegue y Vista de Implementación. [6]

Modelo de Despliegue

Este artefacto muestra la configuración de los nodos de proceso en el tiempo de ejecución, los enlaces de comunicación entre ellos, las instancias de componente y los objetos que residen en ellas. El modelo de despliegue consta de uno o más nodos (elementos de proceso que poseen como mínimo un procesador, memoria y posiblemente otros dispositivos), dispositivos (nodos estereotipados sin capacidad de proceso en el nivel modelado de abstracción), conectores entre nodos y conectores entre nodos y dispositivos. [6]

Artefactos a ser generados por el Diseñador

Clases del Diseño

Este artefacto es una descripción de un conjunto de objetos que comparten las mismas responsabilidades, relaciones, operaciones, atributos y semánticas. [6]

Paquetes del Diseño

Este producto de trabajo es una recopilación de clases, relaciones, ejecuciones de guión de uso, diagramas y otros paquetes. Se utiliza para estructurar el modelo de diseño dividiéndolo en componentes más pequeños, para agrupar los elementos del modelo de diseño relacionados con los objetivos organizativos y a menudo para la gestión de la configuración. [6]

Realización de Subsistemas del Diseño

El artefacto subsistema de diseño encapsula comportamiento, proporcionando interfaces explícitas y formales, y no expone (por convenio) el contenido interno. Esto proporciona la capacidad de encapsular completamente las interacciones de una serie de clases y/o subsistemas. [6]

Artefactos a ser generados por el Implementador

Modelo de Implementación

El modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación (directorios y archivos, incluyendo código fuente, datos y archivos ejecutables), identifica los componentes físicos de la implementación para que puedan comprenderse y gestionarse mejor, define las principales unidades de integración alrededor de las cuales se organizan los equipos, así como las unidades que se pueden versionar, desplegar y reemplazar separadamente. [6]

Patrones de Software

Un patrón se define como una solución probada con éxito que aparece una y otra vez ante determinado tipo de problema en un contexto dado. Este engloba conocimientos específicos, y aplicarlo constituye un eslabón importante en el aprovechamiento de la experiencia acumulada en el campo en cuestión. Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación. En dependencia del problema que solucionan, estos pueden ser clasificados en: patrones de arquitectura, patrones de análisis, patrones de diseño, entre otros. [7]

Patrones de Arquitectura

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos, representando el nivel más alto en el sistema de patrones. Ayudan a especificar la estructura fundamental de una aplicación. Cada actividad de desarrollo es gobernada por esta estructura; por ejemplo: el diseño detallado de los subsistemas, la comunicación y colaboración entre diferentes partes del sistema. Cada patrón de arquitectura ayuda a conseguir una propiedad específica en el sistema global. [5]

Es mayoritariamente común encontrar en una misma solución varios estilos combinados, cada capa o componente puede ser internamente de un estilo diferente al de la totalidad, muchos estilos se encuentran ligados a dominios específicos, o a líneas de producto particulares.

En la arquitectura del trabajo están presentes los siguientes patrones arquitectónicos:

Modelo – Vista – Controlador

El patrón arquitectónico Modelo – Vista – Controlador separa conceptualmente la representación visual de la aplicación, las acciones que intercambian datos y el modelo de negocio y su dominio. En el sistema se concreta con la identificación de 3 elementos diferentes: la vista implementada Java Script que reside del lado del cliente en tiempo de ejecución, y el controlador y el modelo que residen del lado del servidor; la interacción entre la vista y el controlador se realiza a través de una solicitud AJAX y la respuesta dada por el controlador puede encontrarse en JSON o XML según corresponda a la solicitud.

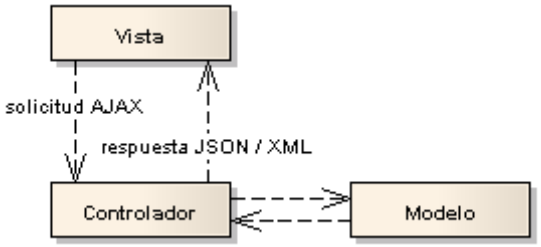

Figura 1: Modelo - Vista - Controlador.

Cliente Servidor

El patrón de arquitectura Cliente Servidor se caracteriza por la existencia de un nodo (o más) servidor donde reside el servicio que se expone y varios clientes que consumen dicho servicio. En el sistema este estilo responde al hecho de que se trate de una aplicación web y se concreta con un servidor de bases de datos, un servidor web y varios clientes que acceden al sistema a través de un navegador.

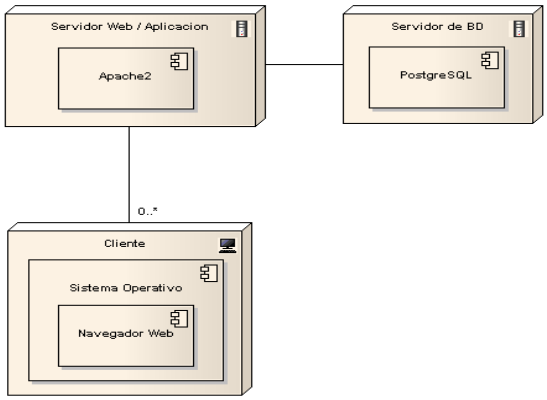

Figura 2: Cliente Servidor.

Basado en Componentes

La Arquitectura basada en componentes identifica como elemento fundamental a los componentes de software y se centra en la integración de varios de ellos para conformar un sistema, se apoya fundamentalmente en la ingeniería de software basada en componentes. Los componentes que

constituyen una arquitectura pueden ejercer influencia sobre la calidad del sistema final.

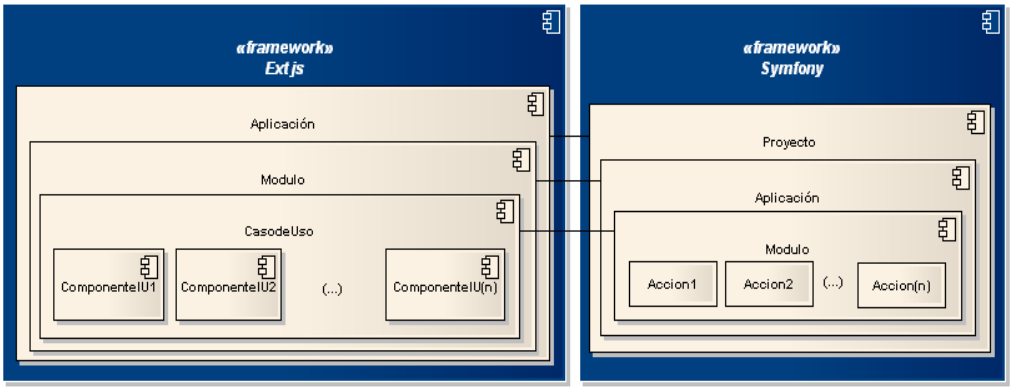

Figura 3: Basado en Componentes.

Multicapas

El patrón Multicapas se caracteriza por organizar los componentes del sistema en capas con responsabilidades bien definidas y donde las capas del nivel más alto invocan los servicios de las del nivel inferior, en el sistema (Figura 4) es posible identificar las siguientes capas: presentación, negocio, acceso a datos, datos e infraestructura.

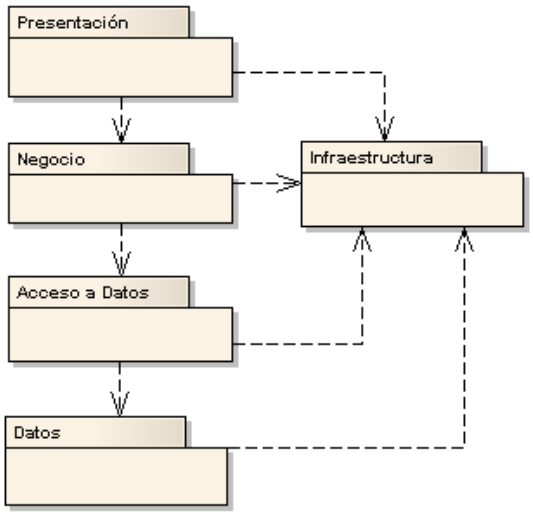

Figura 4: Multicapas.

Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características; una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores; otra es que debe ser reusable, lo

que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. [8]

Patrones GRASP

Los Patrones GRASP son patrones generales de software para asignación de responsabilidades, es el acrónimo de " Patrones Generales de Software para Asignación de Responsabilidades", una serie de buenas prácticas de aplicación recomendable en el diseño de software, que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. [8]

Experto

Experto en información es el principio básico de asignación de responsabilidades. Indica que la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Mantiene el encapsulamiento, los objetos usan su propia información para llevar a cabo sus tareas, soportando un bajo acoplamiento que favorece el hecho de tener sistemas más robustos y de fácil mantenimiento. Distribuye el comportamiento entre las clases que contienen la información requerida, brindando una alta cohesión. [8]

Creador

Identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Se brinda un soporte al bajo acoplamiento. [8]

Alta Cohesión

Plantea que la información que almacena una clase debe ser coherente y estar en la mayor medida relacionada con la clase. Estas clases mejoran la claridad y la facilidad con que se entiende el diseño, además simplifican el mantenimiento y las mejoras en funcionalidad. A menudo genera un bajo acoplamiento, el mejorar en cuanto a funcionalidad permite soportar una mayor capacidad de reutilización, ya que una clase muy cohesiva puede destinarse a un propósito en específico. [8]

Bajo Acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. Estas clases no se afectan por los cambios en otros componentes, fáciles de entender por separado y de fácil reutilización. [8]

Controlador

Asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas

actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Le ofrece mayor potencial de los componentes reutilizables, garantizando que los procesos de dominio sean manejados por la capa de los objetos de dominio y no por la de la interfaz. [8]

Patrones Presentes en Framework Symfony

Front Controller

(Controlador frontal): un controlador concentra todas las solicitudes de la aplicación delegándolas en el manejador específico que generalmente es un comando. [7]

Singleton

(Instancia única): es un patrón creacional a nivel de objetos, el problema que lo motiva es la necesidad de tener una instancia única de un objeto, su propósito es garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma. [7]

Command

(Orden): tiene por propósito encapsular en un objeto la acción que satisface una petición. En el framework Symfony las acciones son un ejemplo de este patrón. [7]

Intercepting Filter

(Filtros Interceptores): tiene por objetivo realizar procesamientos antes y después de que se maneje la acción. Es útil para tratar problemas de seguridad, validaciones, conversiones etc., también se implementa en Symfony. [7]

Herramienta de Modelado

Visual Paradigm

Es una herramienta UML profesional que 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, generar código inverso, código desde diagramas y documentación. Está diseñado para desarrollar software bajo el estilo de Programación Orientada a Objeto (POO). Dentro de sus características fundamentales se encuentran:

- Multiplataforma: Es soportada en plataformas Java para Sistemas Operativos Windows, Linux y Mac OS X.
- Interoperabilidad: Intercambia diagramas UML y modelos con otras herramientas. Soporta exportar e importar a XMI19, XML y archivos Excel. Importa archivos de proyectos de Rational Rose. Posee integración con Microsoft Office Visio.

- Modelamiento de los Requisitos: Permite captura de requisitos con diagrama de requisitos, modelamiento de casos de uso y análisis textual.
- Colaboración de Equipo: Realiza el modelado simultáneamente con el Visual Paradigm TeamWork Server y Subversion.
- Generación de Documentación: Comparte y genera los diagramas y diseños en formatos como PDF, HTML y Microsoft Word.
- Editor de Detalles de Casos de Uso: Posee entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Ingeniería de Código: Permite generación de código e ingeniería inversa en lenguajes como Java, C, PHP, XML, Python, C#, VB .NET, ActionScript, Delphi y Perl.
- Modelado de Procesos de Negocio: Visualiza, comprende y mejora los procesos de negocio con la herramienta para estos procesos.
- Integración con Entornos de Desarrollo: Apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación, en IDE como Eclipse, Microsoft Visual Studio, NetBeans, Sun ONE, Oracle JDeveloper, JBuilder y otros.
- Modelamiento de Bases de Datos: Generación de bases de datos y conversión de diagramas entidad -relación a tablas de base de datos, además de mapeos de objetos y relaciones.

Marcos de Trabajo (framework)

Symfony 1.1.7

Symfony es un framework bajo licencia MIT que sigue el patrón arquitectónico MVC para desarrollar aplicaciones web comerciales, gratuitas y/o de software libre escritas en PHP5. Tiene como objetivo acelerar la creación y mantenimiento de aplicaciones web y sustituir repetitivas tareas de codificación; proporcionando para esto varias herramientas que siguen la mayoría de las mejores prácticas y patrones de diseño para la web. Este framework es fácil de instalar y extender, lo que permite su integración con librerías desarrolladas por terceros entre ellas Zend Framework. Presenta también herramientas de configuración que lo hacen lo suficientemente flexible como para adaptarse a las necesidades de la aplicación que se desea desarrollar. Otras de las características de Symfony es que brinda soporte a la internacionalización y es independiente del sistema gestor de base de datos, algo que se garantiza a través de un ORM (Propel o Doctrine). Se puede ejecutar tanto en plataformas Unix y Linux, como en plataformas Windows. Su código fuente incluye más de 8000 pruebas unitarias y funcionales, y ha sido utilizado en numerosos proyectos reales como Yahoo Bookmarks y Yahoo

Answers. Las aplicaciones que emplean Symfony, pueden controlar hasta el último acceso a la información, debido a que el framework brinda herramientas para ello, e incluye por defecto protección contra ataques XSS. [7]

Ext JS 3.0

Ext JS es una librería de Java Script para el desarrollo de aplicaciones web enriquecidas, haciendo un uso intensivo de las tecnologías AJAX, XHTML/ DHTML y DOM. Originalmente fue creado como una extensión de Yahoo User Interface (YUI) otro framework similar, Ext incluye interoperabilidad con jQuery, Prototype y Scrpt.aculo.US.

Sus principales características son:

- Alto rendimiento en ejecución debido a la optimización de código Java Script.
- Controles de usuario personalizables.
- Modelo orientado a componentes, bien diseñado y extensible.
- Posee una API intuitiva y fácil de utilizar.
- Es distribuido bajo licencias Open Source y comerciales.

La versión 3.0 fue liberada el 3 de junio de 2009 entre sus mejoras incluye:

- Soporte para peticiones directas, CRUD y REST.
- Nuevos ejemplos y componentes (incluye un componente para gráficas).
- Más de 1000 mejoras y correcciones.
- Administración de memoria mejorada para Internet Explorer 6.
- API documentada.
- Compatibilidad con versiones anteriores.

Lenguajes de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; permitiendo al desarrollador comunicarse con los dispositivos de hardware y software existentes. [5]

PHP5

Hypertext Pre Processor es un lenguaje script del lado del servidor que es embebido dentro del código HTML, utilizado para la generación de páginas Web dinámicas. Es un lenguaje sencillo de sintaxis cómoda, es rápido y dispone de una gran cantidad de librerías que facilitan el desarrollo de aplicaciones.

Entre sus principales ventajas se encuentran:

- PHP corre en diversas plataformas utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en 25 plataformas, incluyendo diferentes versiones de Unix, Windows (95, 98, NT, ME, 2000, XP, Vista, Seven) y Mac.
- La sintaxis de PHP es similar a la de C, por esto cualquier programador con experiencia en lenguajes del estilo C podrá entender rápidamente PHP.
- PHP es completamente expandible. Está compuesto de un sistema principal, un conjunto de módulos y una variedad de extensiones de código.
- Posee muchas interfaces distintas para cada tipo de servidor. Este lenguaje actualmente se puede ejecutar bajo Apache, IIS, AOLServer, Roxen y THTTPD.
- Puede interactuar con numerosos motores de bases de datos tales como MySQL, SQL, Oracle, Informix, PostgreSQL, entre otros.
- PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.
- Está completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.
- PHP es de código abierto, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además no está forzado a pagar actualizaciones anuales para tener una versión del producto. [9]

El 13 de julio de 2004, fue lanzado PHP5, utilizando el motor Zend Engine 2.0. La versión más reciente de PHP es la 5.3.6 que fue lanzada el 7 de marzo de 2011.

XHTML

El XHTML acrónimo en inglés de Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir al HTML como estándar para las páginas Web.

Está encaminado al uso de un etiquetado correcto, por lo que exige una serie de requisitos básicos a cumplir en cuanto al código. Algunos de estos requisitos son:

- Elementos correctamente anidados.
- Etiquetas en minúsculas.
- Elementos cerrados correctamente.
- Atributos de valores entrecomillados.

Java Script

Es un lenguaje de tipo script compacto, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.

Además es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Los programas Java Script van incrustados en los documentos XHMTL o en un fichero js, y se encargan de realizar acciones en el cliente, como puede ser: pedir datos y/o confirmaciones, mostrar mensajes, crear animaciones y comprobar campos.

Ventajas:

- Los programas escritos en este lenguaje no requieren de mucha memoria ni tiempo adicional de transmisión, por ser pequeños y compactos.
- No requiere un tiempo de compilación, ya que los scripts se pueden desarrollar en un período de tiempo relativamente corto.
- Es independiente de la plataforma hardware o sistema operativo, y funciona correctamente siempre y cuando exista un navegador que lo soporte.
- Asegura la permanencia de una operación realizada, y aunque falle el sistema esta no podrá deshacerse.

CSS

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Entre las características de este lenguaje se encuentra la separación de los contenidos de su presentación, siendo esto imprescindible para crear páginas Web complejas.

JSON

Acrónimo de Java Script Object Notation, es un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de Java Script que no requiere el uso del lenguaje de marcas extensible (XML). Es muy sencillo de usar, especialmente como alternativa a XML. Una de sus ventajas sobre XML como formato de intercambio de datos es que prevalece un formato mucho más sencillo a la hora de escribir un analizador semántico del mismo.

XML

Es el estándar de Extensible Markup Language (Lenguaje de Etiquetado Extensible), conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades y contiene tres características muy importantes que son: extensibilidad, estructura y validación.

Ventajas de XML:

- Las aplicaciones se pueden generar rápidamente y su mantenimiento es sencillo.
- Separa los datos de la presentación y del proceso, lo que permite mostrar y procesar los datos al gusto deseado con sólo aplicar distintas hojas de estilo y aplicaciones.
- La información es más accesible y reutilizable, por la flexibilidad de las etiquetas de XML que permiten su utilización sin tener que amoldarse a reglas específicas de un fabricante.
- Ofrece un formato para la descripción de datos estructurados, facilitando declaraciones de contenido más precisas y resultados de búsquedas más significativos en varias plataformas.

Java

Es un lenguaje de programación desarrollado por Sun Microsystems a principios de los 90. Está diseñado para ser suficientemente simple, permitiendo que los programadores puedan lograr fluidez con el lenguaje. Trabaja con sus datos como objetos y con interfaces a estos, además proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

Sus principales características son:

- Orientado a objetos: soporta las características esenciales del paradigma de la programación orientado a objetos: encapsulación, herencia y polimorfismo.
- Robusto: elimina el uso de apuntadores para referenciar áreas de memoria, además libera al desarrollador de la necesidad de desalojar la memoria que la aplicación ya no usa.
- Multiplataforma: el mismo código Java que funciona en un sistema operativo, funciona en cualquier otro sistema operativo que tenga instalada la máquina virtual de Java.
- Multitareas: a pesar de que las capacidades multitarea que pueden ser implementadas en Java dependen en gran parte del sistema operativo en el cual se ejecuten, digamos Windows o Unix, dichas capacidades superan en gran manera a los entornos de flujo único que ofrecen otros lenguajes de programación. Permite la ejecución concurrente de varios procesos ligeros o hilos de ejecución. [5]

Tecnología AJAX

AJAX, acrónimo de Asynchronous Java Script And XML (Java Script asíncrono y XML) es una tecnología que facilita la creación de aplicaciones interactivas en la Web que se ejecutan en el navegador de los usuarios y mantienen comunicación asíncrona con el servidor; posibilitando que se puedan efectuar cambios sobre una página sin necesidad de recargarla, aumentando de esta manera

la interactividad, velocidad y usabilidad de la misma.

AJAX está conformado por:

- XHTML y CSS: para crear una presentación basada en estándares.
- DOM: para la interacción y manipulación dinámica de la presentación.
- XML, JSON: para el intercambio y la manipulación de información.
- XMLHttpRequest: para el intercambio asíncrono de información.
- Java Script: para unir todas las demás tecnologías.

Herramientas de Desarrollo

NetBeans IDE

NetBeans IDE es un entorno de desarrollo integrado distribuido por SUN Microsystems bajo licencia dual (GPL y CDDL), lo que significa que es libre, gratuita y sin restricciones de uso. Permite a los programadores escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos a los que se puede extender. [10]

NetBeans funciona en sistemas operativos compatibles con la máquina virtual de Java (Windows XP, Windows Vista, Windows 7, Ubuntu, Solaris, Mac OS X 10.5 o superior). En su versión 6.5 se ha incorporado el lenguaje de trabajo PHP y en su versión actual se ha incluido a Symfony como framework por defecto. El 15 de Junio de 2010 fue liberada la versión 6.9.

Servidor Web Apache 2

El servidor Apache es desarrollado por la Apache Software Foundation, presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, entre sus principales ventajas se encuentran: posee gran cantidad de extensiones para diversas tecnologías, además de una amplia documentación, es libre, modular y multiplataforma.

Se señala como desventaja que no posee interfaz gráfica que facilite su configuración. [7]

Servidor Web Apache Tomcat

Tomcat es un servidor web con soporte de Servlets y JSPs. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que

disponga de la máquina virtual Java. Tomcat es mantenido y desarrollado por miembros de la Fundación de Software Apache y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software Licence. La primera distribución de Tomcat fue la versión 3.0. La versión más reciente es la 7.0.

Conclusiones

En el presente capítulo se realizó un estudio de los principales sistemas generadores de reportes existentes en el mundo y en la Universidad de las Ciencias Informáticas, haciendo énfasis en las ventajas que brinda la inclusión del módulo Diseñador de Consultas en el GDR. Se seleccionó como intérprete SQL JSqlParser. Se presentó como metodología de desarrollo RUP, de la cual se desempeñarán los roles de: Arquitecto, Diseñador e Implementador. Se decidió por una arquitectura basada en los patrones arquitectónicos: Modelo – Vista – Controlador, Cliente Servidor, Arquitectura Basada en Componentes y Arquitectura Multicapa. Se escogieron los patrones GRASP como patrones de diseño, se decidió utilizar como herramienta de modelado Visual Paradigm, como lenguajes de programación PHP5, XHTML, Java Script, CSS, JSON y XML; como herramientas de desarrollo NetBeans IDE, como servidor web para correr código PHP Apache2 y como servidor web para correr código Java Apache Tomcat.

Capítulo 2: Diseño de la Solución

Introducción

En el presente capítulo se describe el diseño escogido para dar solución al problema que se enfrenta, a través del cual se modela la aplicación que será capaz de cumplir los requisitos del trabajo. Para ello se desarrollan los artefactos correspondientes al flujo de trabajo diseño.

Objetivos del Diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción, lo cual contribuye al desarrollo de una arquitectura estable y sólida. Este flujo de trabajo es un refinamiento del análisis, que tiene en cuenta los requisitos no funcionales.

Sus principales objetivos son:

- Transformar los requisitos en un diseño del sistema en creación.
- Evolucionar una arquitectura sólida para el sistema.
- Adaptar el diseño para que se ajuste al entorno de implementación, con un diseño pensado para el rendimiento. [6]

Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. El módulo Diseñador de Consultas del Generador Dinámico de Reportes debe cumplir con los requisitos funcionales que se muestran a continuación.

RF1 Diseñar una consultas SQL a partir de un modelo semántico.
RF2 Modificar el diseño de una consulta SQL.
RF3 Especificar operaciones a una consulta SQL.
RF4 Especificar condiciones a una consulta SQL.
RF5 Editar las condiciones de una consulta SQL.
RF6 Relacionar tablas de una consulta SQL.
RF7 Editar la relación existente entre dos tablas de una consulta SQL.
RF8 Mostrar el código de la consultas SQL en diseño.
RF9 Editar el código de una consulta SQL en diseño.
RF10 Actualizar el diseño de una consulta SQL.
RF11 Guardar una consulta SQL.
RF12 Cargar una consulta SQL previamente guardada.

RF13 Ejecutar una consulta SQL.

Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. El Diseñador de Consultas es un módulo del producto Generador Dinámico de Reportes, perteneciente al proyecto: Sistema de Información de Gobierno, por lo que debe poseer los requerimientos no funcionales definidos en la arquitectura del mismo.

Usabilidad

La herramienta se desarrolla para la WEB, pero con características muy similares a las aplicaciones de escritorio, en cuanto al diseño de las interfaces visuales y los tiempos de respuesta, por lo que el usuario debe poder diseñar una consulta SQL de manera ágil y sencilla, sin ser un experto en el proceso de diseño de un reporte. [11]

Eficiencia

La eficiencia del sistema depende en gran medida de la velocidad de conexión a las base de datos, así como del volumen de información contenido en las mismas. Sin embargo el sistema debe mantener tiempos de respuestas en un marco razonable de diez segundos, permitiendo que existan al menos 100 usuarios conectados de forma simultánea. [11]

Interfaz

El usuario deberá acceder a la aplicación a través del protocolo HTTP usando el navegador Firefox en su versión 2.0 o superior. [11]

Restricciones de Diseño e Implementación

El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.2 o superior, como framework de desarrollo se usará Symfony en su versión 1.1.7 y la librería de Java Script Ext JS versión 3.0. Como herramienta de desarrollo de usará NetBeans 6.9 y como sistema gestor de base de datos se utilizará PostgreSQL 8.4. [11]

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 4.0 GNU/Linux o superior.
- Paquetes: apache2, php5, libapache2-modphp5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-xsl, php5-gd.

- Usuario con privilegios de administración. [11]

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 4 GNU/Linux o superior.
- PostgreSQL versión 8.3 o superior.
- PGAdmin III o algún administrador para PostgreSQL.
- Usuario con privilegios para instalar la base de datos.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP. [11]

Hardware

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

- Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- 512 MB de RAM.
- 40 GB de espacio en disco duro. [11]

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.
- 512 MB de RAM.
- 40 GB de espacio en disco duro. [11]

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

- Procesador Intel Pentium 4 1.7 GHz, o AMD similar.
- 256 MB RAM.
- 20 GB de espacio en disco duro. [12]

Diagrama de Casos de Uso del Sistema

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

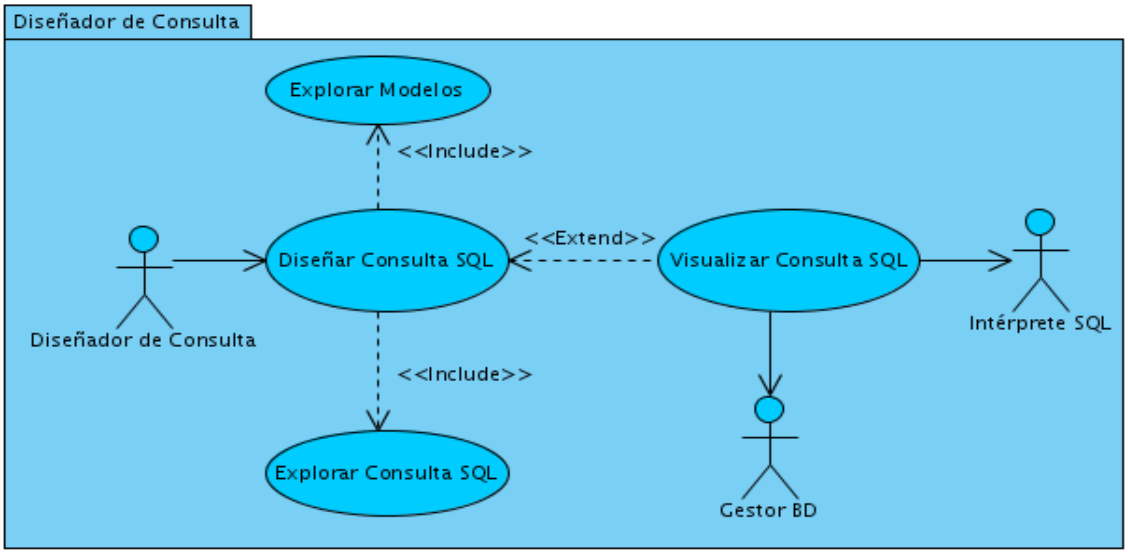

Figura 5: Diagrama de casos de uso del sistema.

Vista Lógica

La Vista lógica proporciona una base para comprender la estructura y la organización del diseño del sistema, en el flujo de trabajo de análisis y diseño. En el presente trabajo se muestra la vista lógica de las clases que abarcan el comportamiento arquitectónicamente significativo, aplicando la arquitectura definida por el ingeniero: Armando Robert Lobo, arquitecto actual del proyecto, en su trabajo de diploma “Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas”.

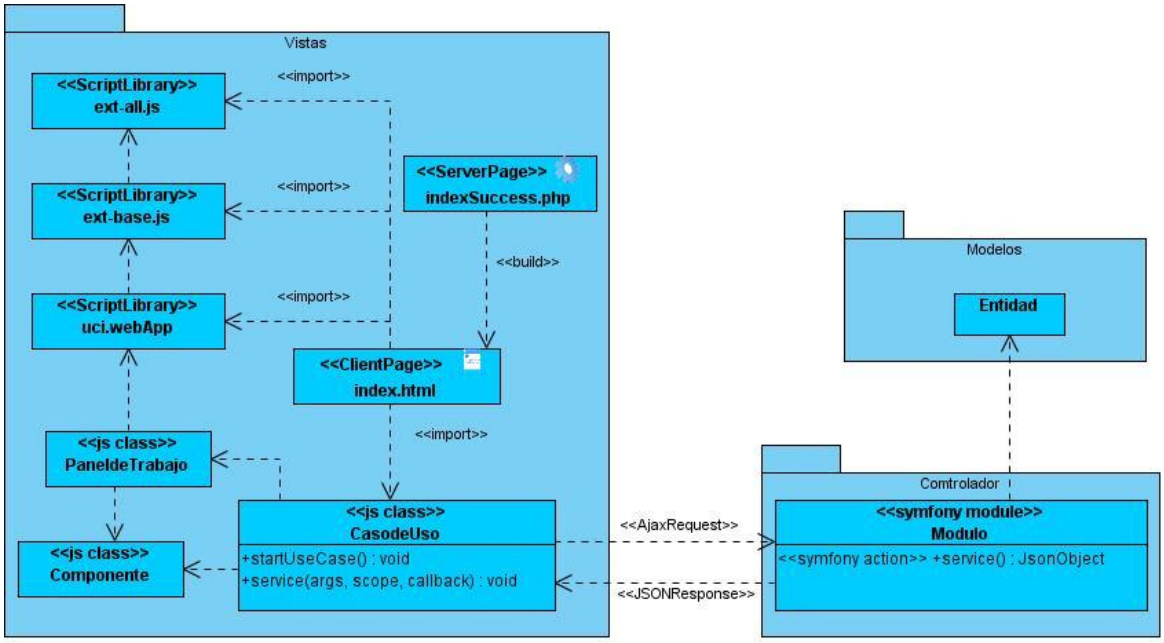

Figura 6: Vista lógica del sistema.

Como se trata de una aplicación enriquecida para la web, que utiliza AJAX como tecnología, es necesario considerar los elementos que se encontrarán del lado del cliente y los que se encontrarán del lado del servidor, diferenciándose que la implementación de los primeros es en Java Script y la de los segundos es fundamentalmente en PHP.

Del lado del cliente los principales elementos corresponden a las librerías “ext-base.js”, “ext-all.js”, “uci.webApp”, a los componentes específicos del negocio como paneles de trabajos, componentes propios del negocio y servicios asociados al caso de uso, y a los componentes genéricos fuertemente reutilizables. Del lado del servidor se encuentran las acciones (vistas con el estereotipo “symfony action”) agrupadas en el correspondiente módulo (visto con el estereotipo “symfony module”) al que pertenecen, se destaca la relación que existe de flujo de información entre el componente caso de uso del lado del cliente y el componente módulo del lado del servidor. Se puede observar que las solicitudes de AJAX se encuentran implementadas de manera concentrada en el Caso de Uso que representa el enlace con las acciones que del lado del servidor respaldan estas solicitudes, devolviendo las respuestas específicamente en formato JSON al caso de uso.

Puesto que es necesario tener en cuenta el hecho que cuando se invoca un servicio en el caso de uso y se realiza una solicitud AJAX esta invocación es asíncrona se hace imprescindible disponer de un observador que esté al tanto de la ocurrencia o del fallo de la operación, por esta razón se incluye además de los argumentos formales propios de la solicitud un argumento “callback” que se corresponde con la función que observa la terminación y respuesta de la solicitud AJAX y un argumento “scope” que representa al objeto de ámbito sobre el que se invoca la función “callback”.

Las relaciones fundamentales entre estos elementos son las de importación, dependencia, y las de flujo de datos. La importación se garantiza por la vía tradicional a través de las correspondientes etiquetas HTML. La dependencia entre el panel de trabajo y el caso de uso es necesaria dado que es el panel de trabajo quien estructura y enlaza a los diferentes componentes con los servicios, de esta forma se garantiza el desacoplamiento entre los componentes y la lógica del negocio en particular. La relación de flujo de datos entre el caso de uso y el módulo representa la correspondencia de las invocaciones AJAX con las acciones que del lado del servidor darán respuesta a las mismas, se transfiere tanto en la solicitud como en la respuesta el objeto con todos los atributos que se envían. [7]

Diagramas de Clases del Diseño

El diagrama de clases del diseño muestra los atributos y métodos de cada clase, representando de forma sencilla la colaboración y las responsabilidades de cada una de ellas, entorno al sistema que conforman.

Diagrama de Clases del Diseño del Caso de Uso Explorar Modelos

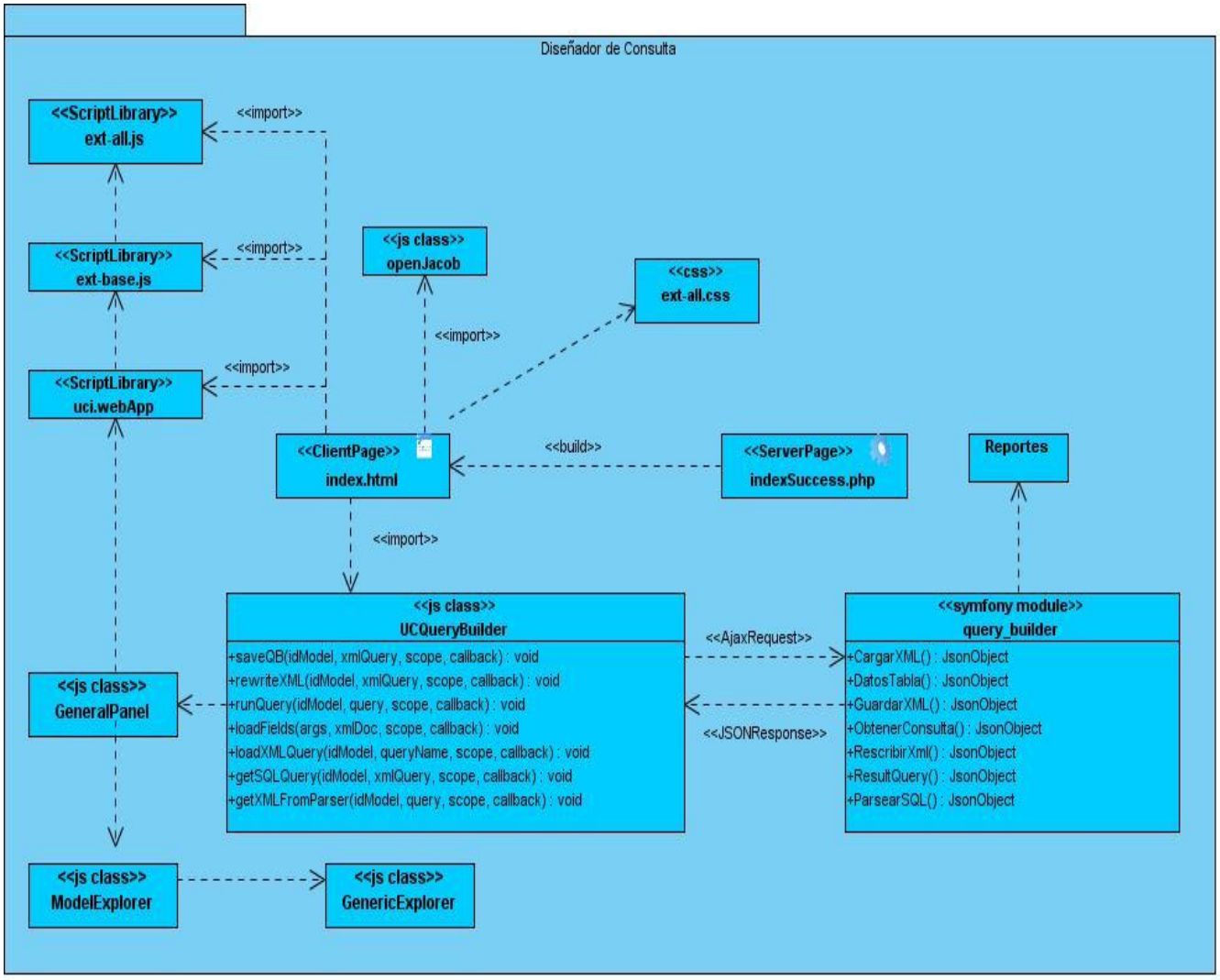

Figura 7: Diagrama de clases del diseño CU Explorar Modelos.

Diagrama de Clases del Diseño del Caso de Uso Explorar Consulta SQL

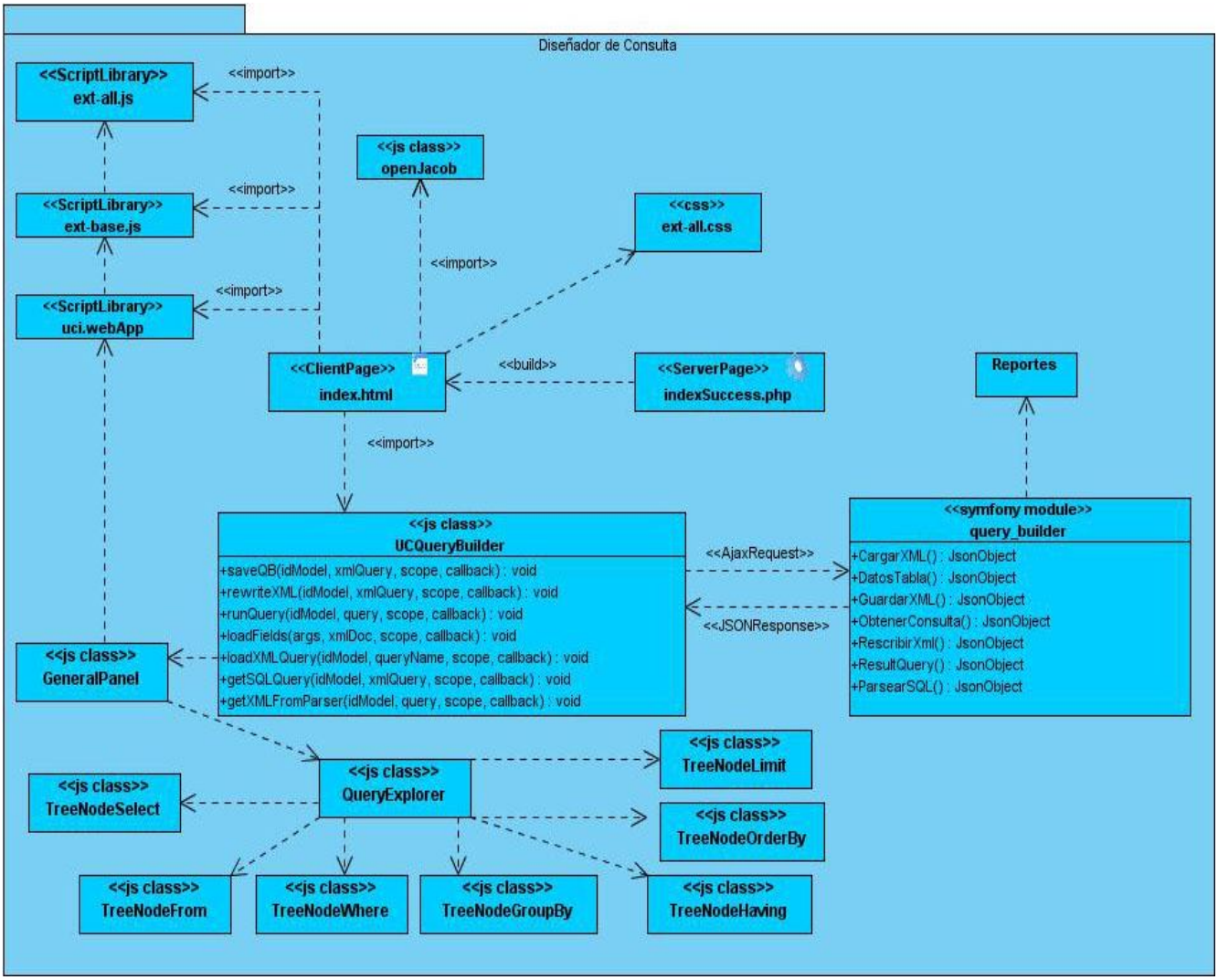

Figura 8: Diagrama de clases del diseño CU Explorar Consulta SQL.

Diagrama de Clases del Diseño del Caso de Uso Diseñar Consulta SQL

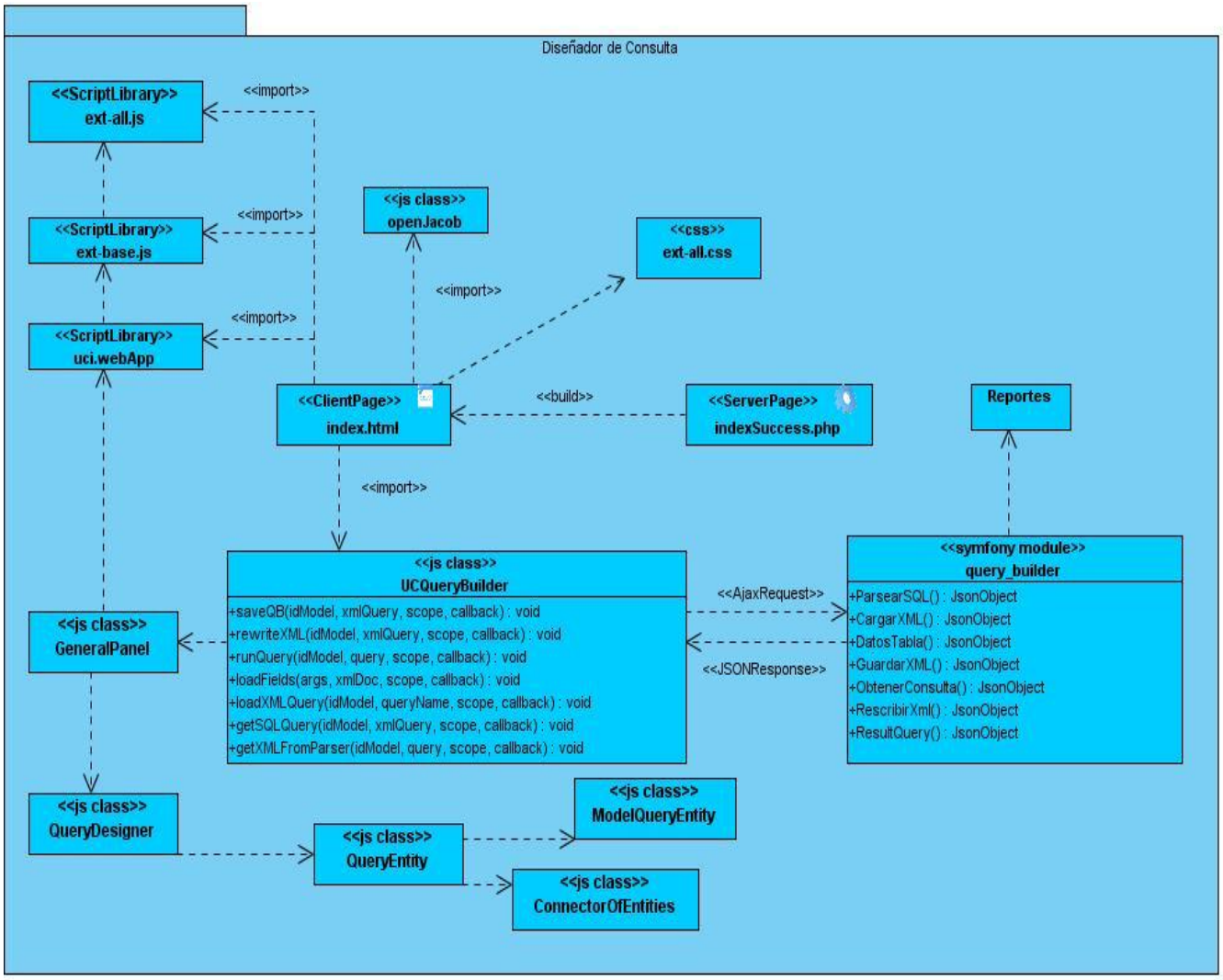

Figura 9: Diagrama de clases del diseño CU Diseñar Consulta SQL.

Diagrama de Clases del Diseño del Caso de Uso Visualizar Consulta SQL

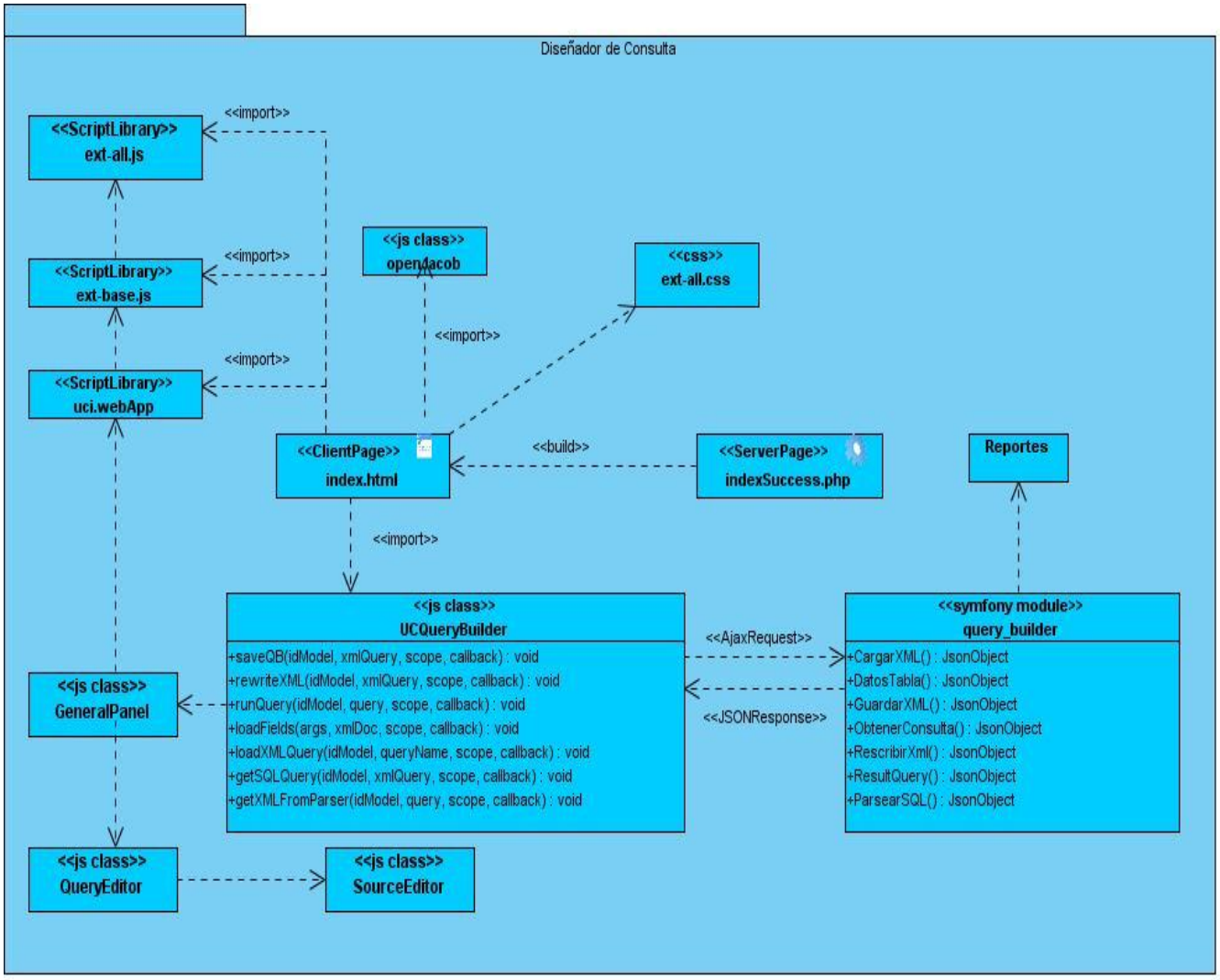

Figura 10: Diagrama de clases del diseño CU Visualizar Consulta SQL.

Descripción de las Principales Clases

UCQueryBuilder: Es la clase controladora principal, contiene las principales operaciones que se realizan del lado del cliente.

Tabla 1: Clase controladora UCQueryBuilder

Nombre: UCQueryBuilder	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	**Descripción:**
saveQueryBuilderTemplate()	Guarda una consulta previamente diseñada.
runQueryBuilderTemplate()	Ejecuta una consulta previamente diseñada.
rewriteXMLQuery()	Reescribe una consulta.
loadFieldsFromTable()	Carga los campos de una tabla.
loadXMLQuery()	Obtiene el XML de una consulta.
getSQLQuery()	Obtiene el código SQL de una consulta.

query_builder: Es el módulo de Symfony encargado de responder las peticiones Ajax del lado del servidor.

Tabla 2: Módulo de Symfony query_builder.

Nombre: query_builder	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	**Descripción:**
CargarXML()	Devuelve el XML de una Consulta.
DatosTabla()	Devuelve los datos de una tabla.
GuardarXml()	Guarda el XML de una consulta.
ObtenerConsulta ()	Devuelve el código SQL de una consulta.
ReescribirXml()	Reescribe el XML de una consulta.
ResultQuery()	Devuelve los datos de la ejecución de una consulta.

Patrones de Diseño Usados en la Solución del Problema

En función de resolver la problemática existente con respecto a la mala separación de responsabilidades, en versiones anteriores del módulo, el presente trabajo refuerza el uso de los patrones GRASP en la solución del problema propuesto.

Experto

Teniendo en cuenta que el patrón GRASP Experto en Información es el principio básico de asignación de responsabilidades, en el presente trabajo se hace uso del mismo en la clase controladora UCQueryBuilder, indicando que la responsabilidad de la creación del objeto GeneralPanel debe recaer sobre la misma, ya que es quien posee toda la información necesaria para crearlo.

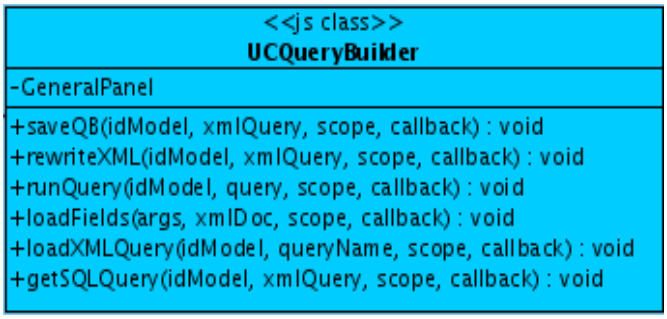

Figura 11: Ejemplo del patrón Experto.

Creador

El patrón GRASP Creador se pone de manifiesto en la clase GeneralPanel, ya que a esta están asignadas las responsabilidades relacionadas con la creación e instanciación de los objetos: QueryExplorer, QueryDesigner, QueryEditor y ModelExplorer.

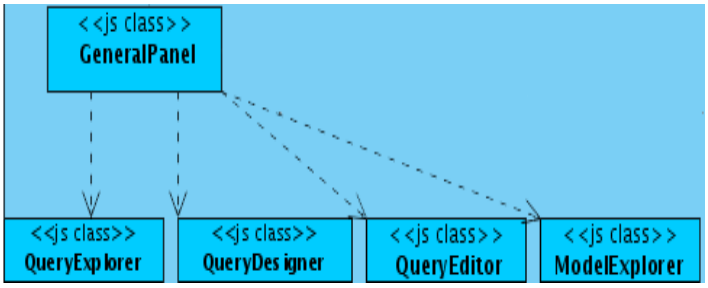

Figura 12: Ejemplo del patrón Creador.

Controlador

El patrón GRASP Controlador, no solo se utiliza para separar la lógica de negocio de la capa de presentación, sino que también es usado del lado del cliente en función de crear un intermediario entre la clase que representa el caso de uso (UCQueryBuilder), las interfaces y el algoritmo que las implementa.

Diagramas de Interacción del Diseño

Los diagramas UML llamados diagramas de interacción (secuencia y colaboración) se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el presente trabajo se usan diagramas de secuencia.

Un diagrama de secuencia muestra una interacción que está organizada como una secuencia temporal. En particular, muestra los objetos que participan en la interacción mediante sus líneas de vida y los mensajes que intercambian, organizados en forma de una secuencia temporal. [10]

A continuación solo se representan los escenarios más significativos, los demás diagramas se pueden ver en el Anexo 1

Diagrama de Secuencia del Caso de Uso Diseñar Consulta SQL

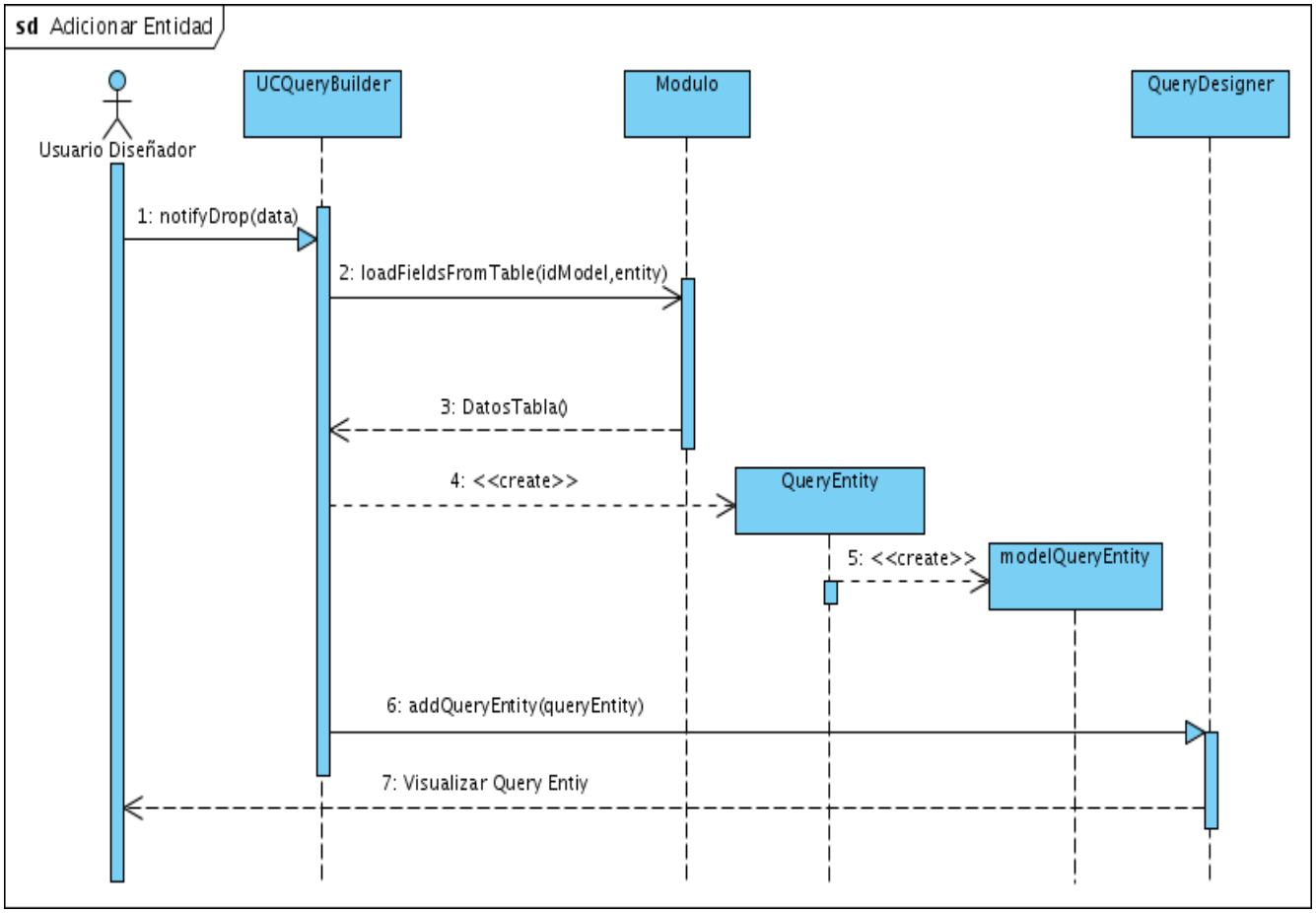

Figura 13: Diagrama de secuencia escenario Adicionar Tabla del CU Diseñar Consulta SQL.

Diagrama de Secuencia del Caso de Uso Explorar Consulta SQL

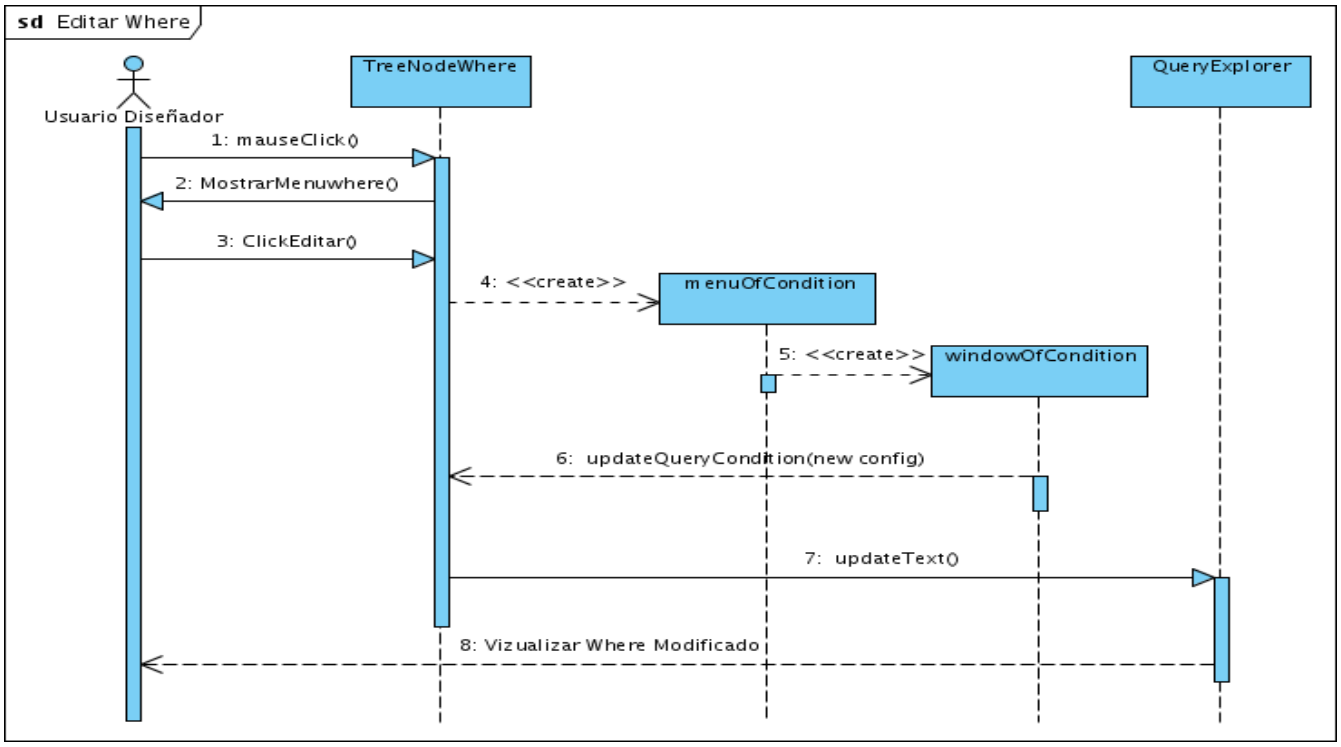

Figura 14: Diagrama de secuencia escenario Editar Where del CU Explorar Consulta SQL.

Diagrama de Secuencia del Caso de Uso Visualizar Consulta SQL

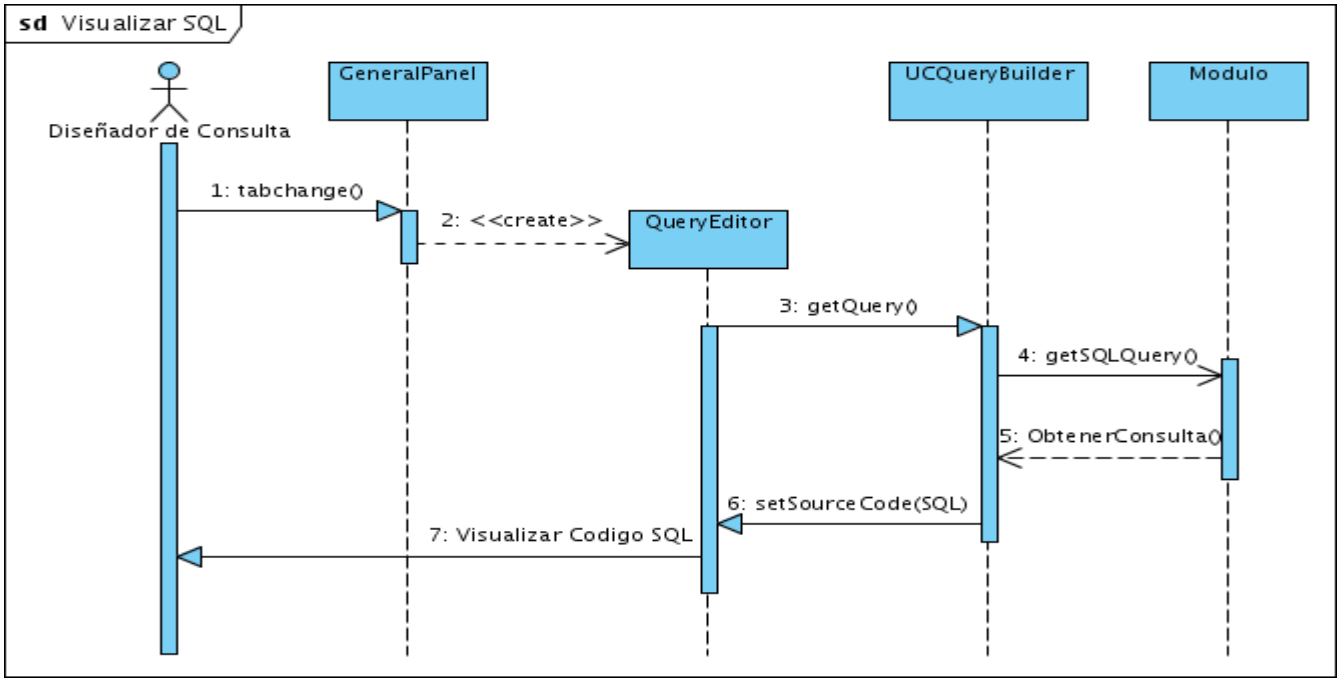

Figura 15: Diagrama de secuencia escenario Visualizar SQL del CU Visualizar Consulta SQL.

Vista de Despliegue

La vista de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos, formando un grafo de nodos unidos por conexiones de comunicación. Un nodo será una unidad computacional de algún tipo, desde un sensor hasta un mainframe.

El Diseñador de Consultas es un módulo del Generador Dinámico de Reportes, por lo que asume las características generales de este, pudiendo ser usado directamente o mediante un sistema externo que establezca una conexión con el mismo. El módulo en cuestión se encarga de facilitar la construcción, edición y ejecución de consultas SQL de forma gráfica, por lo que requiere un servidor de base de datos para la ejecución de las consultas.

Diagrama de Despliegue del Módulo Diseñador de Consultas del GDR

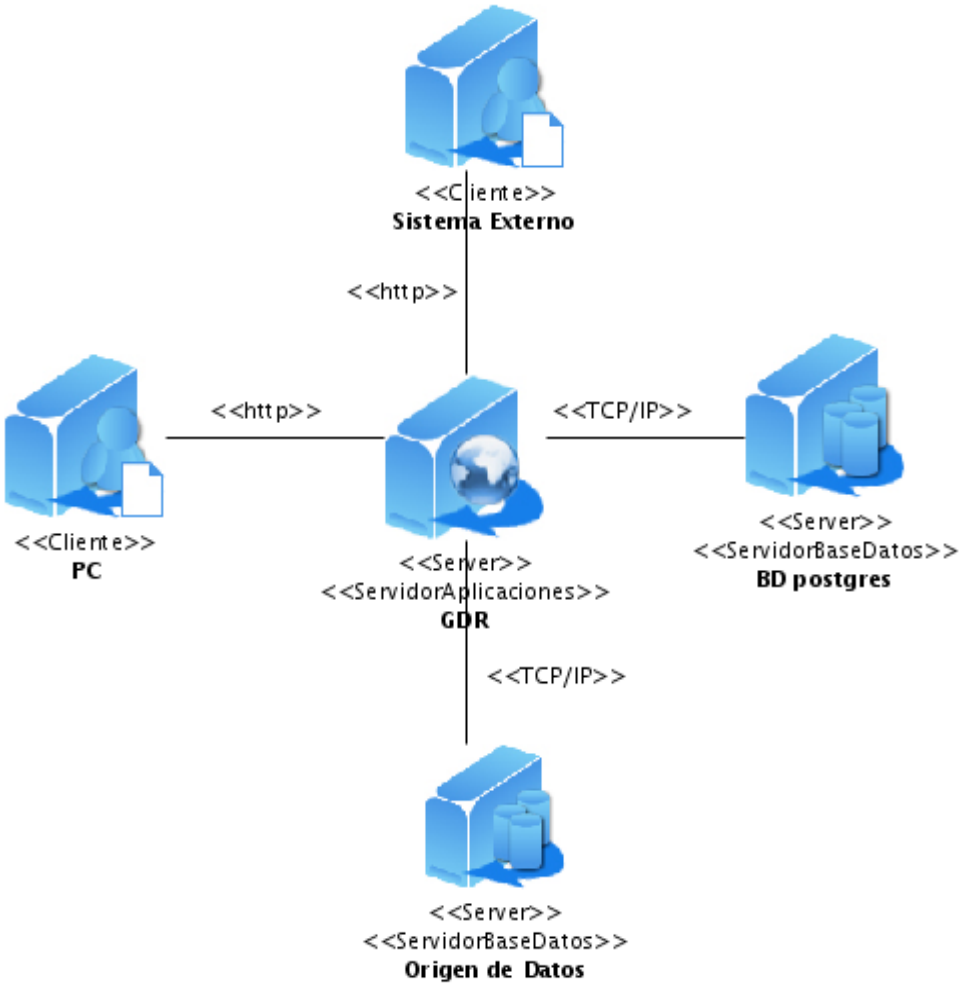

Figura 16: Diagrama de despliegue del sistema.

Conclusiones

En el presente capítulo se ha tomado como punto de partida el diagrama de casos de uso del sistema, así como los requisitos funcionales y no funcionales, con el fin de lograr confeccionar un modelo de diseño, que satisfaga dichos requisitos. En el marco concebido se han modelado los diagramas de clases del diseño, haciendo énfasis en la particularidad de dichos diagramas, ya que se trata de una aplicación enriquecida para la web que hace un uso intensivo de Ajax. Los diagramas de clases serán utilizados posteriormente en la implementación del producto, conjuntamente con los diagramas de iteración representados (en este caso diagramas de secuencia). Con el fin de asegurar el despliegue de la aplicación se ha modelado el diagrama de despliegue del módulo, enfatizando en las características que este posee por ser parte del Generador Dinámico de Reportes.

Capítulo 3: Implementación y Prueba

Introducción

Partiendo del resultado del diseño, en el presente capítulo se generan los artefactos concernientes a la fase de implementación, se modela el sistema en términos de componentes, se presenta un diagrama de despliegue actualizado con los componentes ejecutables distribuidos por los nodos, se especifican los casos de pruebas realizadas al módulo, se dan a conocer las técnicas usadas en la solución del problema y se muestran ejemplos de las implementaciones más relevantes.

Implementación

Esta disciplina explica cómo desarrollar, organizar, realizar pruebas de unidad e integrar los componentes implementados basándose en las especificaciones del diseño. [6]

La finalidad de la implementación es:

- Definir la organización del código, en términos de los subsistemas de implementación, organizados en capas.
- Implementar los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).
- Probar y desarrollar componentes como unidades.
- Integrar los resultados producidos por los implementadores individuales, o equipos en un sistema ejecutable. [6]

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. Muestran 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. A continuación solo se representan los diagramas de componentes pertenecientes a los casos de uso Diseñar Consulta SQL y Explorar Consulta SQL, los

demás diagramas se pueden ver en el Anexo 2

Diagrama de Componentes del Caso de Uso Diseñar Consulta SQL

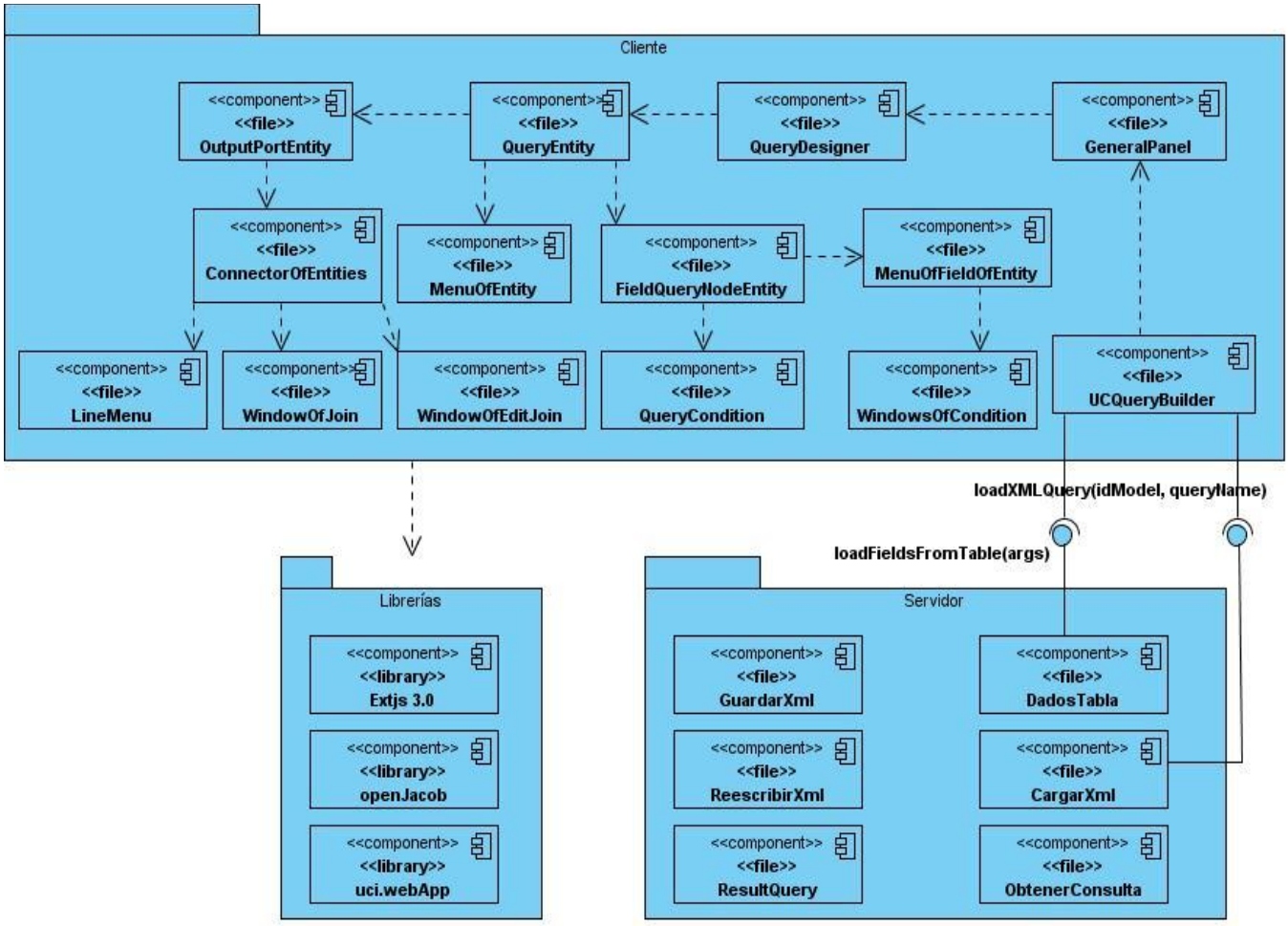

Figura 17: Diagrama de componentes del CU Diseñar Consulta SQL.

Este diagrama representa los componentes que intervienen en la solución propuesta para el caso de uso Diseñar Consulta SQL, el componente QueryDesigner es el encargado de todas las acciones que se realizan en el área de diseño.

Diagrama de Componentes del Caso de Uso Explorar Consulta SQL

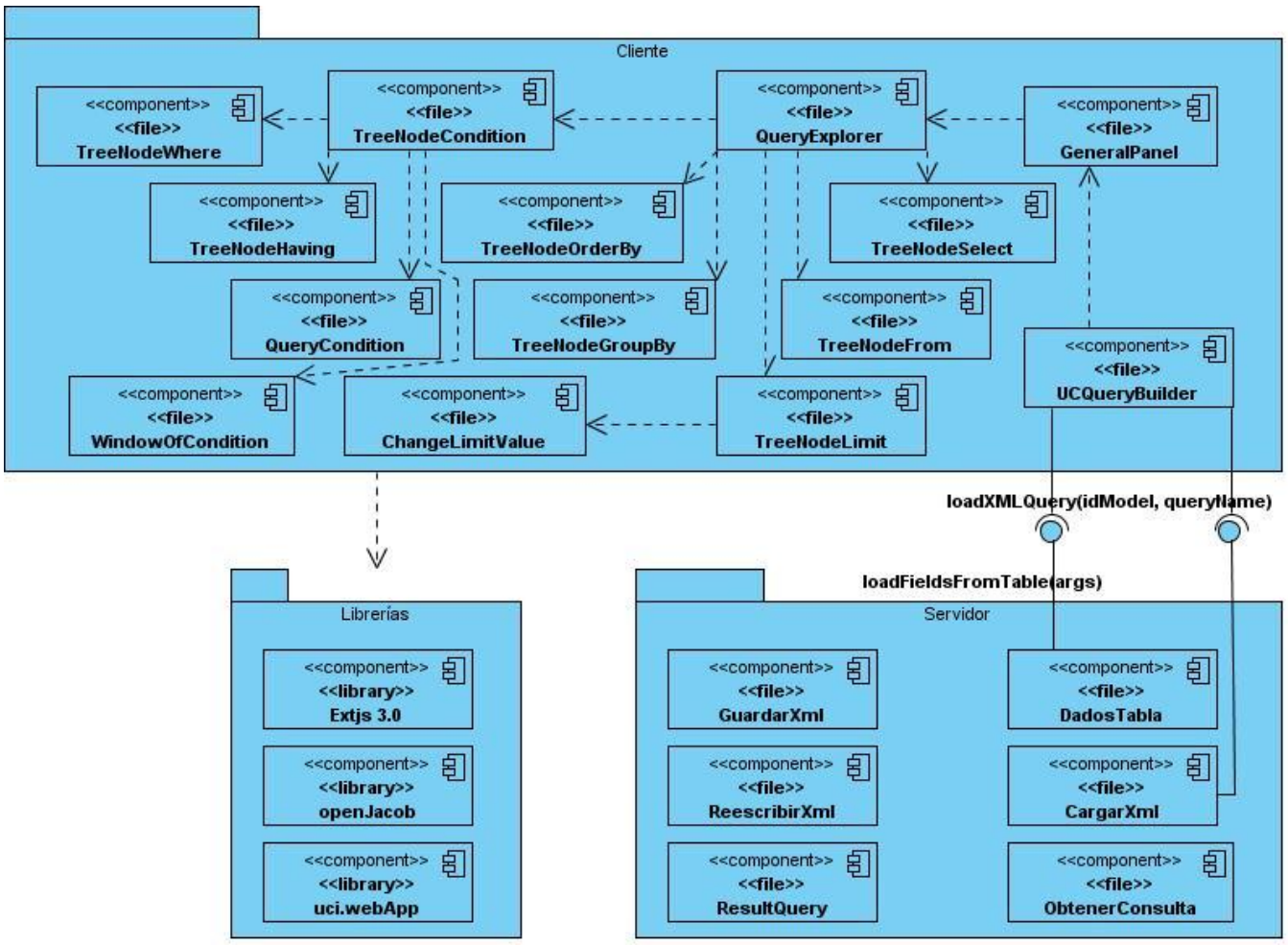

Figura 18: Diagrama de Componentes del CU Explorar Consulta SQL.

Este diagrama representa los componentes que intervienen en la solución propuesta para el caso de uso Explorar Consulta SQL, el componente QueryExplorer es el encargado de facilitar la navegación, a través de las diferentes sentencias generadas.

Diagrama de Despliegue Actualizado

Este diagrama presenta una vista actualizada del diagrama de despliegue, muestra los principales artefactos y componentes de la aplicación, distribuidos en sus correspondientes nodos, así como las dependencias que existen entre cada uno de ellos.

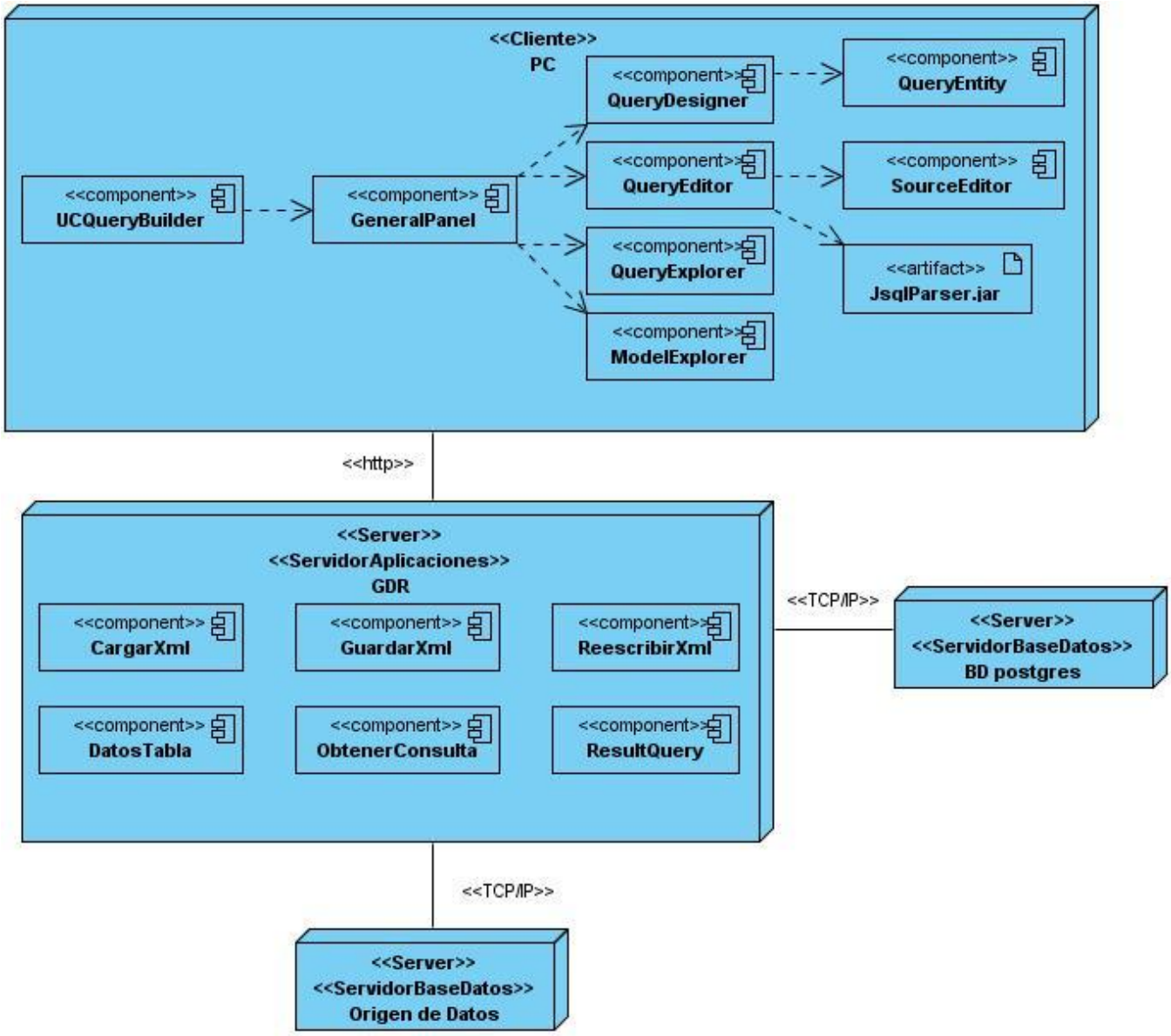

Figura 19: Diagrama de despliegue con los componentes ejecutables distribuidos en los nodos

Mecanismos de Implementación

- El sistema se desarrolla utilizando una arquitectura Modelo-Vista-Controlador.
- La parte cliente se implementa haciendo uso generalizado del framework de desarrollo de interfaces de usuario con JavaScript ExtJS, versión 3.0.
- La parte del servidor se implementa en el lenguaje de programación PHP5, utilizando Symfony 1.1.7 como framework de desarrollo para dicho lenguaje.
- Se respeta el estilo arquitectónico MVC propuesto por Symfony.

- La capa del modelo (acceso a datos) se genera utilizando Propel como ORM incluido en la versión de Symfony a utilizar.
- La vista de Symfony (templates) no se utiliza, y el intercambio de información entre el cliente y el servidor se hace únicamente en formato JSON.
- Cada acción de la capa del controlador se escribe en un fichero independiente según la estructura que define Symfony para este propósito.
- Todos los métodos, nombres de clases y variables se escriben en estructura camelCase, comenzando siempre en minúscula.
- Todas las consultas a ejecutar en bases de datos externas se almacenan y gestionan en formato XML, y se transformarán en el momento de su ejecución mediante transformaciones XSLT en dependencia del gestor de BD.
- Todas las clases tienen el mismo nombre que el fichero al cual pertenecen. Los nombres de las clases están antecedidos por el espacio de nombres al cual pertenece.
- Todas las solicitudes al servidor están implementadas como métodos en el caso de uso donde son utilizadas.

Implementaciones Relevantes

Uno de los requisitos funcionales de la aplicación es: mostrar el código fuente de la consulta en diseño; con el fin de dar cumplimiento al mismo se necesita de una funcionalidad que sea capaz de obtener un código SQL a partir de un código en lenguaje de marcas extensible (XML), para ello se utiliza la función getSQLQuery, esta se encarga de establecer una conexión asíncrona a través de Ajax con la acción: ObtenerConsulta, que se encuentra en el módulo symfony: QueriBuilder, del lado del servidor. La función getSQLQuery envía al servidor el identificador del modelo al cual pertenece la consulta, y el XML de la misma; una vez obtenida una respuesta, en caso de que esta sea satisfactoria, se procederá a modificar el contenido de la Vista SQL, en caso contrario se notificará que ha ocurrido un error al procesar la consulta. La acción symfony ObtenerConsulta se encarga de generar el código fuente correspondiente y enviarlo a la página cliente como una respuesta a la petición Ajax realizada anteriormente.

Funcionalidad getSQLQuery en el Caso de Uso

```
getSQLQuery: function(){
  var queryEditor = this.GeneralPanel.getQueryEditor();
  var xmlQuery = this.GeneralPanel.getXMLQuery();
  var idModel = this.GeneralPanel.getIdModel();
  if(idModel && xmlQuery){
  var mask = this.getLoadMask(queryEditor.getEl(), patdsi.report_generator.query_builder.UCQueryBuilder.LOAD);
        mask.show();
        Ext.Ajax.request({
            url: patdsi.report_generator.query_builder.UCQueryBuilder.GET_SQL_QUERY,
            method: 'POST',
            params: {
                idModel: idModel,
                xmlQuery: xmlQuery
            },
            success: function(response){
                var httpResponse = Ext.decode(response.responseText);

                mask.hide();
                queryEditor.setSourceCode(httpResponse.sql);
            },
            failure: function(){
                mask.hide();
                Ext.MessageBox.show({
                    msg: "Error al procesar la consulta...",
                    icon: Ext.MessageBox.ERROR,
                    title: "Error",
                    buttons: Ext.MessageBox.OK
                });
            }
        });
    }
},
```

Figura 20: Funcionalidad getSQLQuery de la clase UCQueryBuilder.

Funcionalidad ObtenerConsulta en el Módulo Symfony

```
class ObtenerConsultaAction extends sfAction {

    public function execute($request) {
        try {
            $response = array();
            if($request->hasParameter('idModel')) {
                $idModel = $request->getParameter('idModel');
            }

            if($request->hasParameter('xmlQuery')) {
                $xmlQuery = $request->getParameter('xmlQuery');
            }

            $sql=QueryBuilderFunction::getSQLQueryFromXML($idModel, $xmlQuery);

            $response ['success'] = true;
            $response ['sql'] = $sql;

            return $this->renderText(json_encode($response));
        }
        catch (Exception $exc) {
            $response ['success'] = false;
            $response ['msg'] = $exc->getTraceAsString();
            return $this->renderText(json_encode($response));
        }
    }
}
```

Figura 21: Funcionalidad ObtenerConsulta en el Módulo QueryBuilder.

Componentes Reutilizados

JSqlParser: es un proyecto de software libre bajo licencia LGPL, basado en un intérprete generado con JavaCC.

SourceEditor: es un editor de código SQL desarrollado por la línea de Soluciones Integrales del centro, que utiliza la librería codepress.

Pruebas Funcionales

Las pruebas del software son el proceso que permite verificar y mostrar la calidad de un producto, a través de resultados registrables que proporcionan una evaluación. Los casos de prueba son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, haciendo una evaluación de algún aspecto o componente del sistema.

Teniendo en cuenta que el Generador Dinámico de Reportes será revisado por el grupo de calidad de la universidad, en el presente trabajo solo se realizan pruebas funcionales, de tipo caja negra a nivel de desarrollador.

Diseño de los Casos de Pruebas

Tabla 3: Caso de prueba Adicionar Entidad.

Caso de prueba Diseñar Entidad.	
Caso de uso:	Diseñar Consulta SQL.
Entrada:	Se arrastra una tabla desde el explorador de modelos hasta el área de diseño.
	Se arrastra una tabla que ya está graficada.
Resultado:	El sistema grafica correctamente la tabla arrastrada, mostrando todos su campos seleccionados.
	El sistema pregunta si se desea cambiar el alias de la tabla, si se elige la opción de cambiar, el sistema grafica correctamente la tabla arrastrada, mostrando todos sus campos seleccionados.
Condiciones:	En el Explorador de Modelo debe existir al menos un modelo y este debe contener al menos una tabla.

Tabla 4: Caso de prueba Editar Condición Where.

Caso de prueba Editar Condición Where.	
Caso de uso:	Explorar Consulta SQL.
Entrada:	En el explorador de consulta se elige la condición Where que se desea editar y se da clic derecho sobre ella, posteriormente aparecerá un menú en el que se elige la opción editar. Se modifican los campos deseados y se presiona aceptar.
Resultado:	El sistema muestra una ventana con todos los campos de la condición Where seleccionada, permitiendo que estos puedan modificarse, una vez presionado el botón aceptar, el sistema guarda satisfactoriamente los cambios realizados, mostrando estos en el Explorador de Consulta.
Condiciones:	En el Área de Diseño debe existir una consulta graficada. La consulta graficada debe tener al menos una condición Where.

Tabla 5: Caso de prueba Ejecutar Consulta SQL.

Caso de prueba Ejecutar Consulta SQL.	
Caso de uso:	Visualizar Consulta SQL.
Entrada:	Se elige la opción Ejecutar en el menú Consulta, estando activa la pestaña del Área de Diseño.
	Se elige la opción Ejecutar en el menú Consulta estando activa la pestaña de la Vista SQL
Resultado:	El sistema activa la pestaña Vista SQL, mostrando así el código fuente de la consulta graficada, si esta es correcta muestra los datos de la consulta en una nueva ventana, en caso contrario muestra los errores que ha devuelto el servidor en el panel de errores de la Vista SQL.
	Si la consulta es correcta el sistema muestra los datos de esta en una nueva ventana, en caso contrario muestra los errores que ha devuelto el servidor en el panel de errores de la Vista SQL.
Condiciones:	En el Área de Diseño debe existir una consulta graficada.

Tabla 6: Caso de prueba Modificar Consulta SQL.

Caso de prueba Modificar Consulta SQL.	
Caso de uso:	Visualizar Consulta SQL.
Entrada:	Se modifica el código fuente de la consulta en la Vista SQL, y se cambia al Área de Diseño.
Resultado:	El sistema modifica satisfactoriamente el diseño gráfico de la consulta en el Área de Diseño, de forma tal que este se corresponda con el código fuente en la Vista SQL, en caso de que las modificaciones contengan algún error sintáctico o estructuras que no estén soportadas en la versión actual del Diseñador de Consulta el sistema notificará al usuario en el panel de errores de la Vista SQL.
Condiciones:	En el Área de Diseño debe existir una consulta graficada.

Tabla 7: Caso de prueba Adicionar Relación.

Caso de prueba Adicionar Relación.	
Caso de uso:	Diseñar Consulta SQL.
Entrada:	Se da clic sobre uno de los cuatro puertos que posee una tabla y se arrastra hasta uno de los puertos de otra tabla, se establecen los parámetros de la relación y se presiona aceptar.
	Se da clic derecho encima de la línea de la relación y se selecciona la opción Editar Join, en la ventana que aparece se presiona el botón Adicionar y se establecen los parámetros de la nueva relación.
Resultado:	El sistema muestra una ventana donde se establecen los parámetros de la relación, una vez presionado el botón aceptar esta se muestra en el TreeNode FROM.
	El sistema muestra una ventana donde se pueden adicionar nuevas relaciones y configurar los parámetros de las mismas, luego de presionar el botón aceptar esta se muestran en el TreeNode FROM.
Condiciones:	En el Área de Diseño deben existir al menos dos tablas. Debe existir al menos un puerto disponible en cada una de las tablas.

Tabla 8: Caso de prueba Guardar Consulta SQL.

Caso de prueba Guardar Consulta SQL.	
Caso de uso:	Explorar Modelos.
Entrada:	Se elige la opción Salvar en el menú Archivo.
Resultado:	El sistema guarda satisfactoriamente la consulta que se encuentra graficada en área de diseño. En caso de existir una consulta con el mismo nombre se le notificará al usuario, si este elige la opción aceptar, la consulta se reescribirá.
Condiciones:	En el área de diseño debe existir una consulta graficada.

Los demás casos de prueba se pueden ver en el Anexo 3

Conclusiones

En el presente capítulo se ha realizado la descripción de la implementación del módulo Diseñador de Consultas del GDR, representando las dependencias que existen entre los principales componentes, a través del diagrama de componentes correspondiente a cada caso de uso. Se ha representado el diagrama de despliegue de la aplicación, con los componentes ejecutables distribuidos en los nodos que conforman al mismo. Además, se muestran algunas pantallas de funcionalidades de la aplicación con una breve descripción de las mismas, así como los mecanismos de implementación utilizados. Finalmente se han realizado pruebas exploratorias al módulo, arrojando resultados satisfactorios.

Conclusiones

En el presente trabajo se describió el diseño escogido, a través del cual se modela la aplicación que será capaz de cumplir los objetivos propuestos. Se eligieron los patrones GRASP como patrones de diseño y el patrón Modelo – Vista – Controlador como patrón de arquitectura, se presentó la vista lógica de la aplicación, así como el diagrama de despliegue de la misma. Partiendo del resultado del diseño, se realizó la implementación de la aplicación, para ello se modeló el sistema en términos de componentes, se integró un intérprete SQL al módulo, se dieron a conocer las técnicas usadas en la solución del problema y se mostraron ejemplos de las implementaciones más relevantes. Finalmente se aplicaron pruebas funcionales de tipo caja negra, descritas en los casos de pruebas realizados al módulo, validando con estas el correcto funcionamiento del mismo.

Recomendaciones

Luego de haber cumplido los objetivos propuestos mediante la realización del trabajo, se recomienda:

- ✓ Implementar el soporte de consultas anidadas.
- ✓ Implementar el soporte para procedimientos almacenados.

Bibliografía

1. **Stimulsoft.** Stimulsoft. *Stimulsoft.* [Online] 10 6, 2008. [Cited: 11 12, 2010.] http://www.stimulsoft.com.

2. **Crystal Solutions.** Crystal Solutions. *Crystal Reports.* [Online] [Cited: 11 2010, 12.] http://www.crystalsolutions.com.ar/productos/crystalreports.html.

3. **Free Download Manager.** Free Download Manager. *Free Download Manager.* [Online] 11 2006, 1. [Cited: 11 12, 2010.] http://www.freedownloadmanager.org/es/downloads/el_fabricante_de_informe_elegante_%28MySQL_relata_el_generador%29_46911_p/.

4. **Aplicaciones Empresariales.** Aplicaciones Empresariales. *Aplicaciones Empresariales.* [Online] 10 19, 2007. [Cited: 11 12, 2010.] http://www.aplicacionesempresariales.com/agata-report-tu-base-de-datos-al-descubierto.html.

5. **Olivares, José Rolando Lafaurie.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador.* Habana : s.n., 2008.

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

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

8. **Hernández, Pedro Veloso.** *Uso de patrones de arquitectura.*

9. **netbeans.org.** netbeans.org. [Online] www.netbeans.org.

10. **Quintana, Abelardo López.** *Diseño e implementación del módulo de Gestión de indicadores del proyecto del Ministerio del Poder Popular de la Comunicación y la Información.* Ciudad de la Habana : s.n., 2010.

11. **Universidad de las Ciencias Informáticas.** *Especificación de Requisitos de Software V2.0 del Generador Dinámico de Reportes.* s.l. : Sistema de Información de Gobierno.

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

13. **Quintana, Abelardo López.** *Diseño e implementación del módulo de Gestión de indicadores del proyecto del Ministerio del Poder Popular de la Comunicación y la Información.* Ciudad de la Habana : s.n., 2010.

14. **Pérez, Roberto Sarmiento.** *Diseño de la Base de Datos para un Sistema Integral de Gestión de Portafolios de Proyectos.* La Habana : s.n., 2010.

15. **Universidad de las Ciencias Informáticas.** Tesis.uci. [En línea] 2010. http://tesis.uci.cu.

Referencias Bibliográficas

1. **Stimulsoft.** Stimulsoft. *Stimulsoft.* [Online] 10 6, 2008. [Cited: 11 12, 2010.] http://www.stimulsoft.com.

2. **Crystal Solutions.** Crystal Solutions. *Crystal Reports.* [Online] [Cited: 11 2010, 12.] http://www.crystalsolutions.com.ar/productos/crystalreports.html.

3. **Free Download Manager.** Free Download Manager. *Free Download Manager.* [Online] 11 2006, 1. [Cited: 11 12, 2010.] http://www.freedownloadmanager.org/es/downloads/el_fabricante_de_informe_elegante_%28MySQL_relata_el_generador%29_46911_p/.

4. **Aplicaciones Empresariales.** Aplicaciones Empresariales. *Aplicaciones Empresariales.* [Online] 10 19, 2007. [Cited: 11 12, 2010.] http://www.aplicacionesempresariales.com/agata-report-tu-base-de-datos-al-descubierto.html.

5. **Olivares, José Rolando Lafaurie.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador.* Habana : s.n., 2008.

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

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

8. **Hernández, Pedro Veloso.** *Uso de patrones de arquitectura.*

9. **netbeans.org.** netbeans.org. [Online] www.netbeans.org.

10. **Quintana, Abelardo López.** *Diseño e implementación del módulo de Gestión de indicadores del proyecto del Ministerio del Poder Popular de la Comunicación y la Información.* Ciudad de la Habana : s.n., 2010.

11. **Universidad de las Ciencias Informáticas.** *Especificación de Requisitos de Software V2.0 del Generador Dinámico de Reportes.* s.l. : Sistema de Información de Gobierno.

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

Anexos

Anexo 1

Diagrama de Secuencia del Caso de Uso Diseñar Consulta SQL

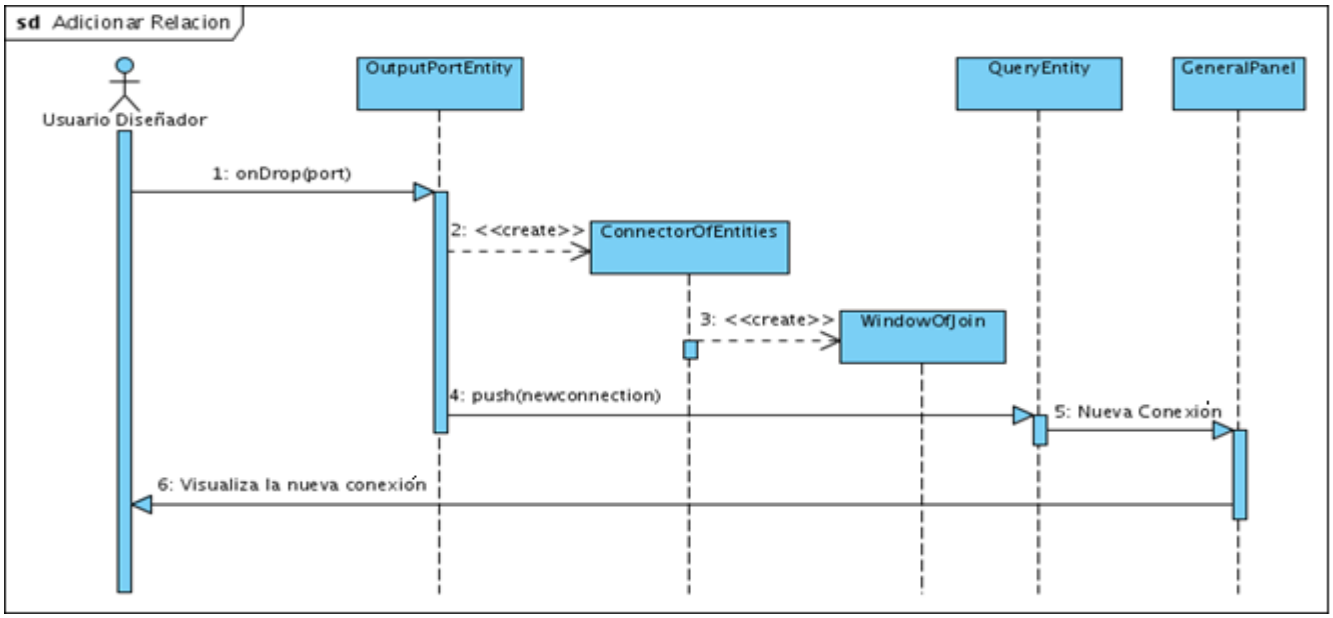

Figura 22: Diagrama de secuencia escenario Adicionar Relación del CU Diseñar Consulta SQL.

Diagrama de Secuencia del Caso de Uso Explorar Consulta SQL

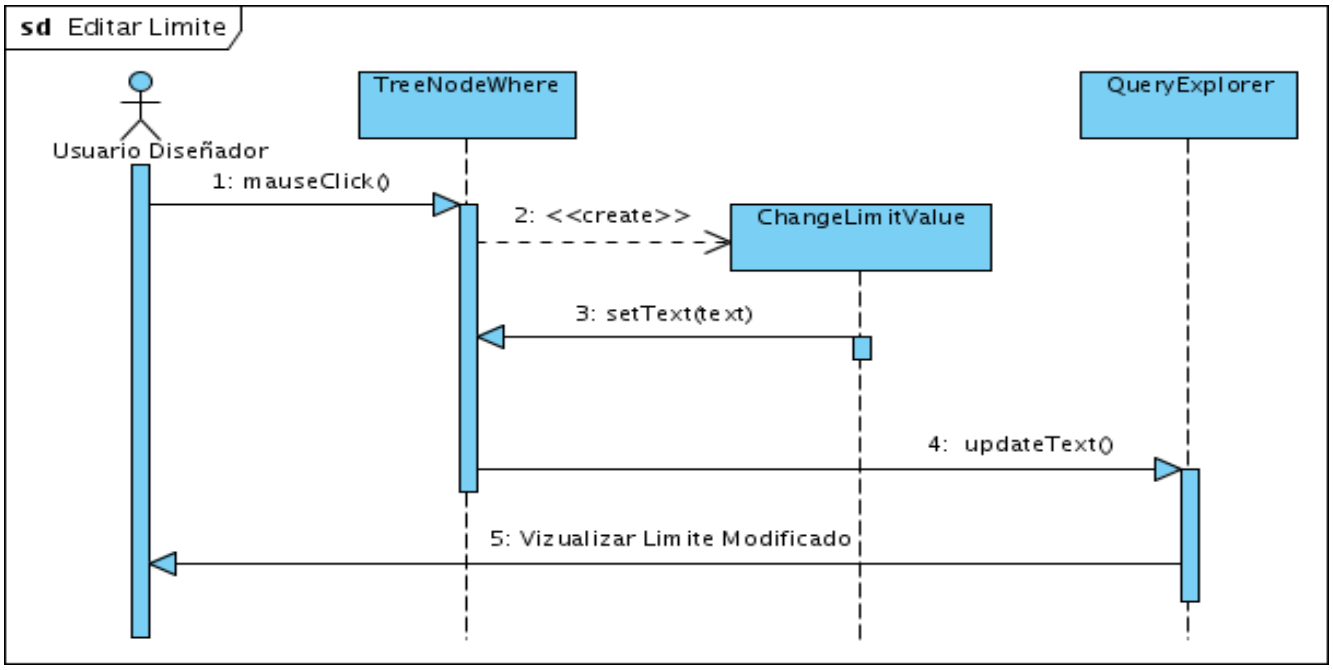

Figura 23: Diagrama de secuencia escenario Editar Límite del CU Explorar Consulta SQL.

Diagrama de Secuencia del Caso de Uso Visualizar Consulta SQL

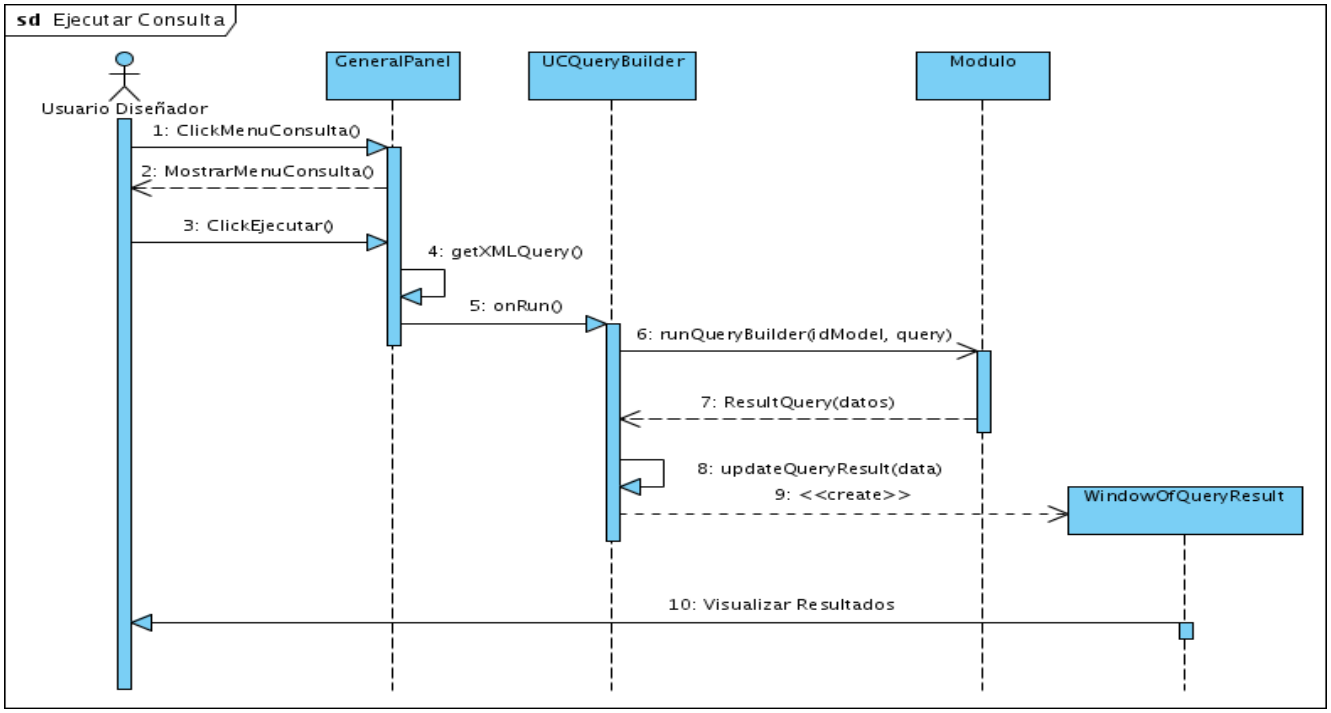

Figura 24: Diagrama de secuencia escenario Ejecutar Consulta del CU Visualizar Consulta SQL.

Diagrama de Secuencia del Caso de Uso Explorar Modelos SQL

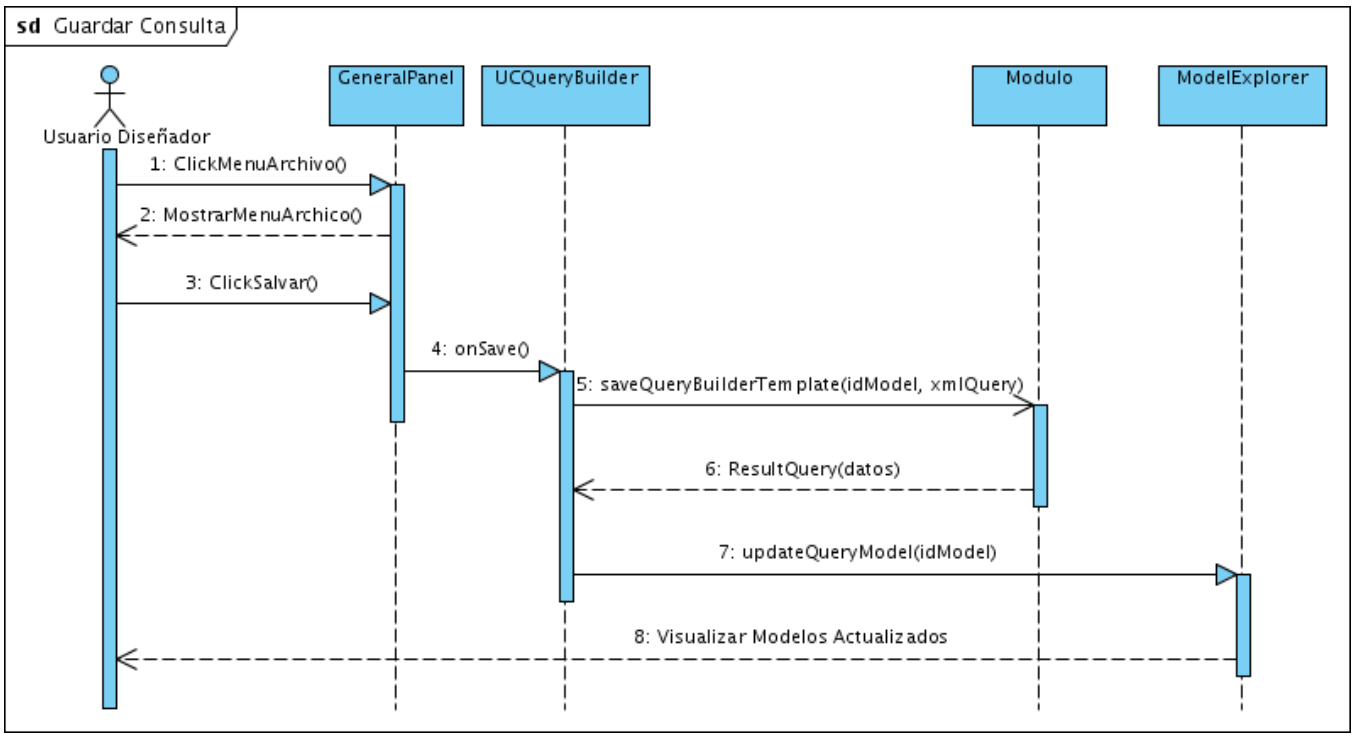

Figura 25: Diagrama de secuencia escenario Guardar Consulta del CU Visualizar Modelos.

Anexo 2

Diagrama de Componentes del Caso de Uso Visualizar Consulta SQL

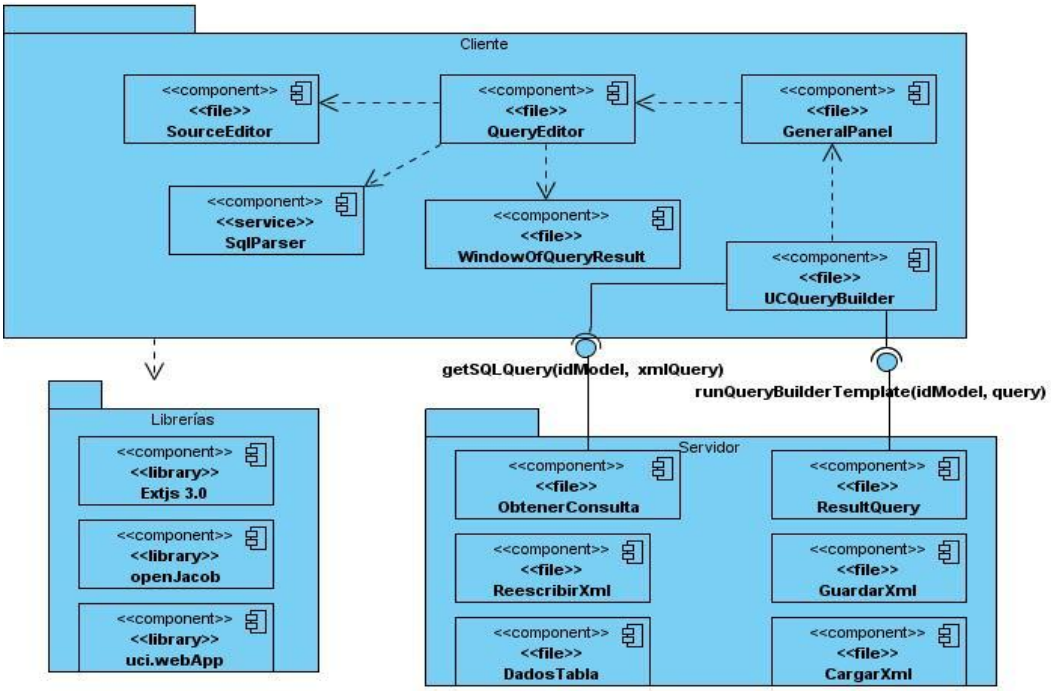

Figura 26: Diagrama de Componentes del CU Visualizar Consulta SQL.

Diagrama de Componentes del Caso de Uso Explorar Modelos

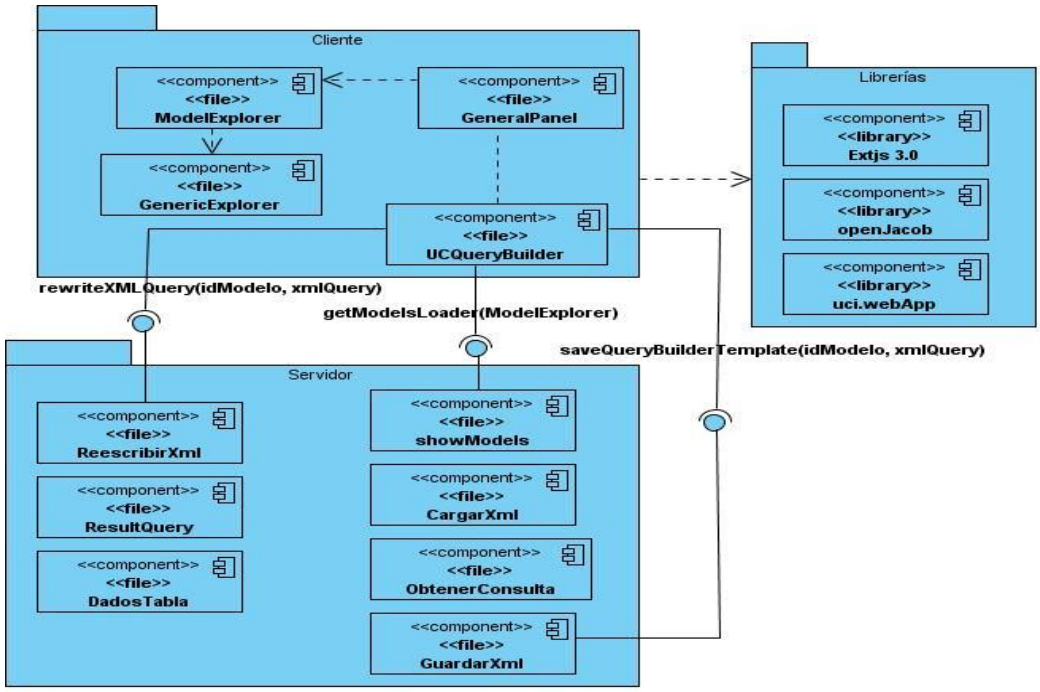

Figura 27: Diagrama de Componentes del CU Explorar Modelos.

Anexo 3

Tabla 9: Caso de prueba Editar Límite.

Caso de prueba Editar Límite.	
Caso de uso:	Explorar Consulta SQL.
Entrada:	En el explorador de consulta se despliega el TreeNode LIMIT, posteriormente se da doble clic en el límite inferior o el superior, según el que se quiera modificar, se entra el nuevo valor y se presiona aceptar.
Resultado:	El sistema muestra una ventana para entrar el nuevo valor, una vez presionado el botón aceptar, el sistema guarda satisfactoriamente los cambios realizados, mostrando estos en el explorador de consulta.
Condiciones:	En el área de diseño debe existir una consulta graficada.

Tabla 10: Caso de prueba Eliminar Entidad.

Caso de prueba Eliminar Entidad.	
Caso de uso:	Explorar Modelos.
Entrada:	Se presiona el clic derecho en el borde superior de la entidad, y se elige la opción Eliminar Tabla.
	Se selecciona la entidad que se desea eliminar y se presiona Delete.
Resultado:	El sistema elimina la entidad seleccionada, actualizando el código fuente de la consulta, así como el Explorador de Consulta.
Condiciones:	En el Área de Diseño debe existir al menos una entidad graficada.

Tabla 11: Caso de prueba Cargar Consulta.

Caso de prueba Cargar Consulta.	
Caso de uso:	Diseñar Consulta SQL.
Entrada:	Se arrastra una consulta desde el Explorador de Modelos hasta el Área de Diseño.
	Se elige la opción Abrir en el menú Archivo y se selecciona la consulta

	deseada.
Resultado:	El sistema carga la consulta arrastrada en Área de Diseño.
	El sistema muestra una vista del Explorador de Modelos donde permite seleccionar la consulta deseada y presionar un botón de aceptar, posteriormente se grafica la consulta seleccionada en el Área de Diseño.
Condiciones:	Debe existir al menos una consulta guardada.

Glosario

Entidad: Es la representación de un objeto o concepto del mundo real que se describe en una base de datos.

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

LGPL: Licencia Pública General Reducida de GNU.

PostScript: Formato de documentos, creado por la empresa Adobe, para describir documentos listos para imprimir.

RUP: Proceso Unificado de Rational.

Arquitectura: Conjunto de decisiones significativas acerca de la organización de un sistema de software.

MIT: Es una licencia de software libre originada en el Instituto de Tecnología de Massachusetts (MIT) que permite modificar el código fuente del software.

Eval: Es una función para analizar si JSON permite una buena transferencia de datos.

Frameworks: Marco de trabajo.

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

Servlets: Objetos que corren dentro del contexto de un contenedor de servlets.

UML: Unified Modeling Language (Lenguaje Unificado de Modelado).

CU: Caso de Uso.

Mainframe: Es una computadora grande, potente y costosa usada principalmente por una compañía para el procesamiento de una gran cantidad de datos.

camelCase: Estilo de escritura que se aplica a frases o palabras compuestas.

JavaCC: Java Compiler Compiler es un generador de analizadores sintácticos de código abierto para el lenguaje de programación Java.