

2011



Universidad de las Ciencias
Informáticas

Facultad 3

Implementación del módulo Diseñador de Consultas
y Generador de Reportes Dinámicos del Tutor
Virtual para la Evaluación del Aprendizaje Autónomo
de Inglés.

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autor: Yubisel Vega Alvarez

Tutor: Msc. Yoan Martínez Márquez.

Cotutores: Ing. Reinier Castillo González.

Lic. Moises A. Mayet Solano.

Ciudad de La Habana

Junio de 2011

Declaración de autoría

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

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

Yubisel Vega Alvarez

Autor

Msc. Yoan Martínez Márquez

Tutor



“El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres.”

Ernesto Guevara de la Serna

Datos de contacto

Tutor: MSc. Yoan Martínez Márquez

Especialidad de graduación:

- Licenciado en Educación. Especialidad: Lengua Inglesa. 2004.
- Máster en Ciencias de la Educación. Especialidad: Tecnologías en los Procesos Educativos. 2007

Correo electrónico: yoanm@uci.cu

Cotutor: Ing. Reinier Castillo González

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Correo electrónico: rcgonzalez@uci.cu

Cotutor: Lic. Moises Alain Mayet Solano

Especialidad de graduación: Licenciado en Ciencia de la Computación

Correo electrónico: mmayets@uci.cu

Agradecimientos

A todos los que de una forma u otra tuvieron que ver en mi educación.

A mis profesores, por haberme enseñado y soportado, a mis profes del politécnico que creyeron en mí, especialmente mi profesor guía en esos tiempos Enrique Figuera.

A mis tutores por estar siempre cuando los necesitaba y por su gran apoyo, por señalarme el norte como dice el buen cubano, sin ellos este trabajo no se hubiera realizado.

A mi familia, por ayudarme tanto y por apoyarme en todo momento, en especial mis padres, mis hermanos y mi sobrino, los quiero.

A mi novia Sisley por soportarme y ayudarme cuando me hizo falta.

A mis compañeros del proyecto, por haberme ayudado con la tesis.

A mis compañeros que vienen conmigo desde el politécnico, son mis otros hermanos.

A mis amigos y amigas, los que vienen desde 1ro y a los de más adelante, en especial a mis compañeros de cuarto en 1ro Alexander, Albert, Pedro, Yero y Daniel.

A todo en que algún momento me ha hecho alguna que otra pregunta, gracias a ellos conozco cosas que de no ser por esas preguntas no las supiera.

A todos en general.

Dedicatoria

A toda mi familia, en especial a mis padres, mis hermanos y mi sobrino.

A todo: el que creyó en mí, el que en algún momento me brindó su ayuda.

A mis amigos y amigas.

A todos.

Resumen

La edición gráfica de consultas y reportes constituye un factor determinante en la aceptación de cualquier aplicación que maneje información almacenada, pues los usuarios finales de estas aplicaciones no necesitan poseer grandes conocimientos del Lenguaje Estructurado de Consultas (SQL). La edición gráfica de consultas permite la selección visual de lo que se quiere mostrar o consultar en la Base de Datos.

La presente investigación persigue como objetivo implementar el módulo “Diseñador de Consultas y Generador de Reportes Dinámicos” del Tutor Virtual para la Evaluación del Aprendizaje Autónomo de Idiomas.

Para la implementación de la aplicación se utilizó como framework de desarrollo Symfony 1.2.8, como lenguaje de programación en el lado del servidor PHP 5.2.5, como entorno de desarrollo NetBeans 6.9, como sistema gestor de base de datos se utilizó PostgreSQL 8.3.4. Además se realizó una valoración crítica del diseño propuesto por los analistas, teniendo en cuenta los requisitos de la aplicación, se hizo un estudio acerca de los componentes a reutilizar y los algoritmos no triviales a implementar, para finalmente describir las nuevas clases y operaciones necesarias para darle solución al objetivo trazado en la presente investigación.

Para la validación, se aplicaron distintas pruebas de software y métricas que permitieron comprobar la validez de la solución propuesta. Se detallaron los casos de prueba, que permitieron detectar la ocurrencia de errores en la aplicación y la posterior solución de los mismos.

Palabras claves

Implementación, Entorno de Desarrollo, Consulta, Lenguaje Estructurado de Consultas, Base de Datos, Algoritmos.

Índice

Introducción	1
Capítulo 1: Fundamentación teórica	7
1.1 Introducción.....	7
1.2 Sistemas de generación de reportes.....	7
1.3 Análisis de herramientas de generación de reportes	8
1.4 Arquitectura Cliente-Servidor	9
1.5 Programación Orientada a Objetos	11
1.6 Lenguajes de Programación.....	12
1.7 Navegador recomendado	14
1.8 Servidor web	15
1.9 Sistemas gestores de bases de datos	15
1.9.1 PostgreSQL.....	17
1.10 Herramientas	17
1.10.1 NetBeans	17
1.10.2 WAMP (Wamp Server).....	18
1.10.3 EMS SQL Manager para PostgreSQL.....	19
1.11 Tecnologías y Framework	20
1.11.1 Symfony.....	20
1.11.2 ExtJS	21
1.11.3 AJAX.....	22
1.12 Metodología de desarrollo	22
1.13 Características e importancia del software a crear	24
1.14 Conclusiones de capítulo	24
Capítulo 2: Descripción y análisis de la solución propuesta.....	25
2.1 Introducción.....	25
2.2 Análisis de los diagramas	25
2.2.1 Diagramas de Clases del Diseño.....	25
2.2.2 Diagramas de componentes	26
2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes que puedan ser rehusados	27
2.3.1 La implementación del MVC que realiza Symfony	28
2.4 Funciones y clases reutilizados.....	29
2.5 Descripción de los algoritmos no triviales a implementar. Análisis de complejidad de los mismos..	31
2.6 Descripción de las nuevas clases u operaciones necesarias	32
2.7 Patrones de Diseño utilizados por el framework Symfony	38
2.7.1 Patrones GRASP	38
2.7.2 Patrones GoF	39
2.8 Estándares de Codificación.....	39
2.8.1 Identificadores	40
2.8.2 Indentación	41
2.8.3 Llaves	41

2.8.4 Líneas y espacios en blanco.....	42
2.8.5 Comentarios	44
2.9 Conclusiones del capítulo.....	45
Capítulo 3: Validación de la solución propuesta	46
3.1 Introducción.....	46
3.2 Diseño de las pruebas unitarias y funcionales	46
3.3 Tipos de pruebas de software	47
3.3.1 Pruebas de Caja Blanca	48
3.3.2 Pruebas de Caja Negra.....	49
3.4 Herramientas para la realización de pruebas	49
3.5 Diseño de Casos de Prueba de Caja Blanca aplicados	50
3.6 Diseño de Casos de Prueba de Caja Negra aplicados	55
3.7 Aplicación de Métricas	59
3.7.1 Tamaño Operacional de Clase	60
3.8 Conclusiones del Capítulo	63
Conclusiones	64
Recomendaciones	65
Bibliografía	66
Anexos	68
Glosario de términos.....	78

Introducción

En la actualidad se ha diversificado la posibilidad de acceso y asimilación de los conocimientos. Es una realidad que cada vez es mayor el número de personas, que por múltiples razones, deciden emprender un proceso de aprendizaje de forma autónoma. En este contexto el espacio virtual emerge como ambiente propicio y con él las Tecnologías de la Información y las Comunicaciones (TIC), sin negar la utilización de los restantes medios y espacios que bajo el principio de complementariedad favorecen la obtención de una solución viable a las demandas en el campo de la educación. Las políticas que fomentan el uso de las TIC benefician sin duda a los institutos de educación superior, aunque esas tecnologías no han sustituido a las modalidades tradicionales de aprendizaje y enseñanza en las aulas. Es indudable que las TIC pueden ampliar el acceso de ciertos estudiantes y que se han convertido en medios para realizar experiencias pedagógicas más vastas, especialmente cuando alumnos y educadores se encuentran separados en tiempo y espacio.

El uso de las TIC en la educación ha evolucionado desde los años sesenta, con los laboratorios de idiomas, hasta la actualidad con propuestas que toman lo más avanzado del mundo de las TIC y la informática para brindar soluciones realmente transformadoras. Los Sistemas Tutores Inteligentes (STI) comenzaron a desarrollarse en los años ochenta con la idea de poder impartir el conocimiento usando alguna forma de inteligencia para poder asistir y guiar al estudiante en su proceso de aprendizaje. Se buscó emular el comportamiento de un tutor humano, a través de un sistema que pudiera adaptarse al comportamiento del estudiante, identificando la forma en que el mismo resuelve un problema a fin de poder brindarle ayudas cognitivas cuando lo requiera.

En el mundo se han desarrollado diversos STI, entre los que se encuentran el STICircSim, el cual fue desarrollado en conjunto por el Departamento de Ciencias de la Computación del “Illinois Institute of Technology” y el Departamento de Fisiología del “Rush College of Medicine”. Este tutor es el más avanzado actualmente en su tipo y se utiliza en el “Rush College of Medicine” para complementar las clases teóricas sobre problemas cardiovasculares. Otro ejemplo es el STI para la enseñanza en niños con dificultades intelectuales y cognitivas (STI PENDIC). En él, se trabaja con dos tipos de patologías: Síndrome de Down y Dislexia.

Cuba es un pionero en esta disciplina por la poca utilización, difusión y uso de este tipo de medios educativos, pues sólo se conoce el uso de dos de ellos en el país: El Sistema Tutorial Inteligente para

Diagnóstico y Tratamiento de Infecciones de Transmisión Sexual (STIITS) y el Sistema Tutorial Inteligente Multimedia generalizado (STI MD) en las disciplinas de Física, Química, Matemática y Biología, desarrollado por el Instituto Superior Politécnico "José Antonio Echeverría".

La Universidad de las Ciencias Informáticas (UCI) se encuentra en la vanguardia de los procesos que se llevan a cabo para potenciarla educación cubana con el empleo de las TIC. La asignatura de Idiomas Extranjeros es una de las que se encuentra inmersa en el proceso de crear nuevos métodos y medios que ayuden a desarrollar habilidades en los estudiantes, mediante metodologías activas que le permitan a estos profundizar en temas de su interés en el aprendizaje del idioma, para esto se han utilizado diferentes herramientas, un ejemplo, fue el Proyecto de Innovación Pedagógica Centro Virtual de Autoaprendizaje de Lenguas Extranjeras (CEVALE), este surgió a partir del re-diseño de la estrategia curricular de la disciplina de Idioma Inglés, el cual no pudo ser terminado por múltiples razones, las cuales no son objetivos de esta investigación.

En la escuela se encuentra en uso el Entorno Virtual de Aprendizaje (EVA), montado en el sistema de gestión de cursos Moodle, en el que estudiantes y docentes planifican, acuerdan, desarrollan y participan activamente en experiencias orientadas a lograr un mayor aprendizaje de los estudiantes. Sin embargo, el EVA no permite la integración de diferentes módulos que permitan dar seguimiento de forma única el proceso enseñanza –aprendizaje, tampoco posee la retroalimentación de las evaluaciones hechas a los estudiantes con la funcionalidad de identificar las debilidades y/o potencialidades de los estudiantes como un STI.

En este contexto se desarrolla en el año 2009 el Proyecto de Innovación Pedagógica: Tutor Virtual para la Evaluación del Aprendizaje Autónomo de Inglés (VIRTEVALL). Con la creación de esta plataforma se perfeccionará el método de estudio tradicional, de forma tal, que el estudiante será capaz por sí mismo a través de las distintas funcionalidades que brinda la aplicación de ver el avance obtenido, señalando las principales debilidades y favoreciendo así el trabajo autónomo para el aprendizaje del idioma Inglés. El Proyecto de Innovación Pedagógica está estructurado sobre la base del sustento teórico provisto por una tesis doctoral sobre evaluación del aprendizaje autónomo de Idiomas Extranjeros. Como aporte práctico, ofrece el STI para implementar la metodología propuesta. Además está sustentado por el desarrollo de dos tesis de maestría que abordarán la integración de las tecnologías en la gestión de la evaluación del aprendizaje autónomo de Idiomas Extranjeros y la integración de algoritmos de inteligencia artificial en la gestión de la evaluación para aprendizaje autónomo de Idiomas Extranjeros.

En una primera etapa, por la complejidad del sistema a desarrollar, se definió por la dirección del proyecto la división en nueve módulos estructurados en dependencia de las diferentes funcionalidades identificadas, entre ellos se encuentra el módulo Diseñador de Consultas y Generador de Reportes Dinámicos. Actualmente el sistema no cuenta con el módulo que permita diseñar consultas en tiempo real, así como la generación de reportes de manera dinámica.

En todo sistema informático es fundamental contar con una herramienta complementaria cuya función sea la de generar consultas sobre la información almacenada en una base de datos para obtener reportes a partir de esta. En este sentido los diseñadores de consultas y los generadores de reportes son herramientas adicionales a los sistemas de información. A diferencia de las consultas tradicionales, con los generadores de consultas dinámicos se tiene mayor control sobre el aspecto que tendrá la salida de la información, en otras palabras, el usuario podrá controlar los aspectos a mostrar en el reporte que desea generar. En la actualidad existen generadores de reportes comerciales como el Crystal Reports y el Jasper Report. El problema con estos y otros generadores de reportes es que su costo es elevado, y por esta razón no pueden ser explotados por la mayoría de los usuarios que necesitan una herramienta de este tipo.

El actual STI no cuenta con la determinación necesaria de la información relevante para la evaluación del aprendizaje, como este sistema está dirigido a profesores no tienen forma de acceder a los datos, actualmente se necesita de conocimientos de base de datos para la obtención de datos persistentes relacionado con los alumnos, se tiene carencia de información sobre el estado de los procesos de evaluación en diferentes formatos.

Teniendo en cuenta lo anteriormente planteado, queda como **problema a resolver** la siguiente interrogante: ¿Cómo generar información estadística del proceso de evaluación del aprendizaje autónomo de idiomas extranjeros que facilite la obtención de evidencia de los avances de los estudiantes en este proceso?

Siendo el **objeto de estudio**: Procesos de desarrollo de software en la generación de información estadística.

Presentando como **campo de acción** en esta investigación: Implementación de los procesos de generación de información estadística.

Determinándose como **objetivo general**: Implementar el módulo diseñador de consultas y generador de reportes dinámicos que permita obtener información estadística del proceso de evaluación de aprendizaje autónomo de idiomas.

A partir de un análisis del objetivo general se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Desarrollar el diseñador de consultas y generador de reportes dinámico para la aplicación VIRTEVALL.
- Validar el diseñador de consultas y generador de reportes dinámico de la aplicación VIRTEVALL.

Después del problema referenciado anteriormente se puede enunciar la siguiente **idea a defender**: Implementando el módulo de diseño de consultas y generación de reportes dinámicos se facilitará la obtención de información estadística sobre los estudiantes durante el proceso de evaluación del aprendizaje autónomo de idiomas extranjeros.

Para darle cumplimiento al objetivo trazado se ha decidido desarrollar las siguientes **tareas de la investigación**:

- Revisión de los antecedentes de otros Sistemas Tutores Inteligentes.
- Valoración de distintos gestores de reportes que potencien y agilicen el proceso de obtención de información estadística.
- Estudio de componentes existentes que agilicen el desarrollo de software educativo utilizando la tecnología web.
- Diseño de los artefactos correspondientes al módulo Diseñador de Consultas y Generador de Reportes.
- Implementación de las funcionalidades diseñadas para el módulo Diseñador de Consultas y Generador de Reportes.
- Desarrollo de Casos de Prueba que certifiquen la veracidad de los algoritmos empleados.

Métodos científicos de investigación

Para el desarrollo de la investigación se utilizaron métodos teóricos y empíricos.

Métodos teóricos:

- **Análisis histórico-lógico:** Este método permitirá realizar un análisis de los elementos que se utilizan en la implementación de sistemas Web, dígame, origen y evolución de los diferentes lenguajes de programación, Framework, Entornos de Desarrollo Integrado (IDE) y Sistemas Gestores de Bases de Datos (SGBD), para determinar los más idóneos para desarrollar la aplicación.
- **Analítico - Sintético:** Este método permitirá después de realizar un análisis de las tendencias actuales en torno a la investigación para el diseño de consultas y generación de reportes, determinar los principales métodos y algoritmos que son usados a nivel mundial y que pueden servir en la implementación del módulo.

Métodos empíricos:

- **Entrevista:** Se realizaron múltiples entrevistas no formales a los clientes, líderes de proyecto y analistas en aras de obtener información referente al funcionamiento del sistema.
- **Experimento:** Este método se utilizó para realizar los casos de prueba en un marco cerrado permitiendo obtener el grado de calidad de la solución.

Posibles Resultados:

- Prototipo funcional del módulo Diseñador de Consultas y Generador de Reportes Dinámicos del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Estructuración del contenido

El presente trabajo está conformado por 3 capítulos.

El **Capítulo 1** recoge el análisis de la información existente acerca del tema a tratar y las tendencias actuales que existen en el mundo. También incluye una descripción dando un sustento teórico a la selección de las herramientas a utilizar para resolver el problema planteado.

El **Capítulo 2** recoge la descripción y análisis de la solución propuesta, se describen las clases que modelan la solución y las principales funcionalidades de ellas, además de algunos algoritmos a implementar y la complejidad de los mismos.

El **Capítulo 3** recoge la validación de la solución propuesta, refiriéndose al diseño de las pruebas de unidad que permitan validar la solución propuesta, detallando de las mismas su objetivo, alcance y tipo, al igual que los valores utilizados para la ejecución de dichas pruebas consumando un sistema robusto.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se describen algunos sistemas de generación de reportes en Cuba y en el resto del mundo. Se analizan varias herramientas de generación de reportes. Además se hace referencia a las técnicas de programación, tecnologías y tendencias actuales que pueden ser útiles en el desarrollo de la propuesta de solución, tomando en cuenta lo definido anteriormente en la tesis de análisis y diseño que precede el presente trabajo y centrando un poco más la investigación en el proceso de generación de reportes.

1.2 Sistemas de generación de reportes

Actualmente, en diversos sistemas de generación de reportes a nivel mundial existen problemas con el dinamismo a la hora de brindar la información necesaria. Las herramientas que permiten este trabajo son elaboradas estrictamente para generar un tipo de reporte determinado, lo cual obliga al usuario a redefinir los requisitos cada vez que necesite añadir uno nuevo. Existen múltiples herramientas en el mundo asociadas a este problema, pero en su mayoría tienen un elevado costo de adquisición y, por lo general, están diseñados de una forma tal, que son más bien orientadas al programador y no al usuario.

Cuba, se encuentra en pleno proceso de desarrollo en el campo de la informática y las telecomunicaciones. Para desarrollarlo ha creado un conjunto de instituciones, entre ellas, la Universidad de las Ciencias Informáticas, que con sólo 8 años de fundada, ha logrado informatizar importantes áreas que inciden de manera trascendental en la calidad de vida de la población, como la medicina y la educación.

A pesar de estos importantes logros, aún no se cuenta con una herramienta multipropósito que posibilite al usuario explotar la información que posee almacenada en su base de datos, de una manera ágil y sencilla, pues las herramientas existentes se dedican a la generación de un tipo de reporte específico.

1.3 Análisis de herramientas de generación de reportes

Para lograr una mejor comprensión y conocimiento del estado del arte en el tema tratado, se realizó un breve análisis de algunas herramientas de generación de reportes como es el caso de Crystal Reports y Jasper Report, también se hace referencia al Sistema Automatizado para la Gestión Académica (Akademos) de la UCI.

Jasper Report



Jasper Report (1) es una herramienta para la generación de informes que está implementada en Java y es libre. El funcionamiento consiste en escribir un XML donde se recogen las particularidades del informe. El mismo es tratado por las clases del Jasper y es exportado a otros formatos como: PDF, XML, HTML, XLS o TXT. Jasper Report se integra perfectamente con el JFreeChart que es una librería libre para la generación de todo tipo de gráficas. Para generar el XML es recomendable el uso de la herramienta visual iReport que es un editor gráfico que también está implementado en Java y se integra perfectamente con el Jasper Report.

Para generar un reporte con Jasper Report se emplean una serie de pasos orientados más, al programador que al cliente, ya que tiene que trabajar con distintos tipos de ficheros, los cuales se muestran a continuación:

- Generar un fichero .jrxml en el que se configura cómo se quiere el informe
- Compilar el fichero .jrxml para obtener un fichero .jasper
- Rellenar los datos del informe. Esto generará un fichero .jrprint
- Exportar el fichero .jrprint al formato que se necesite (pdf, etc.). Esto generará el fichero en cuestión.

Crystal Reports



Crystal Reports (2) (3) es una herramienta que ha formado parte de Visual Studio desde 1993, y ahora es el estándar de creación de informes de Visual Studio, se incluye en todas sus copias y se integra directamente esta

plataforma. Además, incorpora la posibilidad de crear contenido interactivo con calidad de presentación al entorno de Windows. Brinda al usuario, entre otras ventajas, la posibilidad de elegir el tipo de base de datos en que desea trabajar y facilita los cambios en el diseño, en otras palabras, a la hora de editar el informe, en la ventana de vista previa, se pueden cambiar formatos, insertar o eliminar objetos, moverlos, etc.

Esta herramienta sólo utiliza una jerarquía para agrupar los datos, o sea, clasifica los datos según la jerarquía otorgada y así define la estructura de todo el informe. Como consecuencia, si se desea generar reportes con estructura compleja, el usuario está obligado a crear varios sub reportes. Además, elabora los reportes en tiempo de diseño ya que está dirigida más al programador que al usuario.

Sistema Automatizado para la Gestión Académica “Akademos”.



La aplicación Akademos, desarrollada en la Universidad de las Ciencias Informáticas en el año 2006, brinda en la actualidad importantes servicios académicos, tanto a los departamentos docentes, como a los estudiantes. 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 los reportes 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.

1.4 Arquitectura Cliente-Servidor

Cliente-Servidor (4): Desde sus inicios el modelo de administración de datos a través de computadoras se basaba en el uso de terminales remotas, que se conectaban de manera directa a una computadora central. Dicha computadora central se encargaba de prestar servicios caracterizados por que cada servicio se prestaba sólo a un grupo exclusivo de usuarios.

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como servidores, estos últimos responden a la demanda del cliente que la produjo.

Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es la Internet.

Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes, locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

- **Cliente:** es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.
- **Servidor:** es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes.

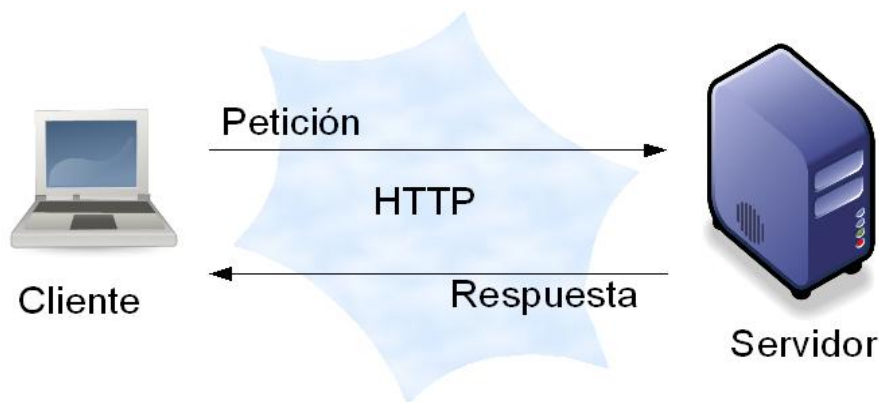


Figura 1.0 Arquitectura Cliente-Servidor

Características del modelo cliente-servidor

En el modelo cliente-servidor podemos encontrar las siguientes características:

- El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
- Las funciones de cliente-servidor pueden estar en plataformas separadas, o en la misma plataforma.
- Un servidor da servicio a múltiples clientes en forma concurrente.

- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un sólo sistema a las estaciones Clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un sólo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.
- Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propios datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los recursos de este host central y otros sistemas que la organización pone a su servicio.

1.5 Programación Orientada a Objetos

Programación Orientada a Objetos (POO) (5) (6): el elemento fundamental de la POO es, como su nombre lo indica, el objeto. Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización. Esta definición especifica varias propiedades importantes de los objetos. En primer lugar, un objeto no es un dato simple, sino que contiene en su interior cierto número de componentes bien estructurados. En segundo lugar, cada objeto no es un ente aislado, sino que forma parte de una organización jerárquica o de otro tipo.

Estructura de un objeto.

Un objeto puede considerarse como una especie de cápsula dividida en tres partes:

- Relaciones: permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos.

- **Propiedades:** distinguen un objeto determinado de los restantes que forman parte de la misma organización y tiene valores que dependen de la propiedad de que se trate. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.
- **Métodos:** son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia.

Muchos definen las siguientes características como las principales de la POO:

- **Abstracción:** es básicamente la capacidad de separar los objetos (al menos mentalmente) para poder verlos de forma singular y poder capturar sus comportamientos. Como cuando describimos el cuerpo humano y decimos cabeza, brazo(s), pierna(s), etc.
- **Encapsulación:** se encarga de mantener ocultos los procesos internos, lo que se necesita llevar a cabo, lo que se desea, dándole al programador acceso sólo a lo que necesita. Esto da dos ventajas iniciales: lo que hace el usuario puede ser controlado internamente (incluso sus errores), evitando que todo colapse por una intervención indeseada. La segunda ventaja es que, al hacer que la mayor parte del código esté oculto, puedes hacer cambios y/o mejoras sin que eso afecte el modo de cómo los usuarios van a utilizar tu código. Sólo tienes que mantener igual la forma de acceder a él. Estas puertas de acceso que das a los usuarios, son lo que se conoce como interfaz.
- **Herencia:** es la capacidad que tiene una clase de derivar las propiedades y métodos de otra. La herencia nos permite, entre otras cosas, evitar tener que escribir el mismo código una y otra vez, puesto que al definir que una categoría (que en programación llamaremos clase) pertenece a otra, automáticamente estamos atribuyéndoles las características generales de la primera, sin tener que definir las de nuevo.

1.6 Lenguajes de Programación

HTML (7) (8): (Hyper Text Markup Language) Lenguaje de Marcas Hipertextuales. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una

página web, fue diseñado con el objetivo de estructurar textos y mostrarlos en el formato estándar de las páginas web, o sea, hipertexto. Es compatible con navegadores reconocidos como: Internet Explorer, Opera, Firefox, Netscape o Safari. Se ha convertido en uno de los formatos con mayor popularidad y facilidad de aprendizaje que existen en la elaboración de documentos para web.

JavaScript (9): es un lenguaje que no requiere de la compilación, muy utilizado en páginas web y posee una sintaxis similar a la de los lenguajes Java y C. No se considera un lenguaje orientado a objetos por no disponer de herencia; más bien se basa en prototipos, donde las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. El código JavaScript que se encuentra dentro de las páginas web puede ser interpretado por todos los navegadores. Permite crear páginas web interactivas (DHTML) con relativa facilidad y se utiliza tanto para programación en el lado del cliente.

La extensión para cliente permite acceder a los objetos que utilizan los navegadores y el Document Object Model (DOM), además la jerarquía de objetos de un documento HTML. Añade soporte para control de eventos, de tal forma que el programa puede interactuar con el usuario.

XML (10) (11): son las siglas en inglés de Extensible Markup Language (lenguaje de marcas extensible), constituye una mezcla de especificación y lenguaje de programación, desarrollada por el W3C. Es una versión de SGML, diseñado especialmente para los documentos de la web. Permite que los diseñadores creen sus propias etiquetas, posibilitando la definición, transmisión, validación e interpretación de datos entre aplicaciones y organizaciones.

XML posee varias ventajas con respecto a otros lenguajes como:

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan errores y se acelera el desarrollo de la aplicación.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

En la implementación de esta herramienta se decidió por parte del equipo de analistas utilizar la tecnología montada en la plataforma de desarrollo PHP.

PHP 5.2.5 (12): es un lenguaje de script interpretado en el lado del servidor, se utiliza en la generación de páginas Web dinámicas; es similar al ASP de Microsoft o el JSP de Sun, está embebido en páginas HTML y se ejecuta en el servidor. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl, aunque tiene algunas características específicas. Su meta es permitir a los desarrolladores la rápida creación de páginas web dinámicas. Es un potente lenguaje, que se interpreta, tanto en el servidor Web como módulo o ejecutado como un binario CGI; puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor.

Este lenguaje libre tiene características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas:

- Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, SybasemSQL, Informix, entre otras.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

1.7 Navegador recomendado

Mozilla Firefox 3.5.x o superior: es el navegador web recomendado para la realización de pruebas y posterior despliegue de la propuesta desarrollada ya que es un programa multiplataforma, que está disponible en versiones para Microsoft Windows, Mac OS X y GNU/Linux. El código ha sido portado por terceros a FreeBSD, OS/2, Solaris, SkyOS, BeOS y más recientemente, Windows XP Professional x64

Edition. El código fuente de Firefox está disponible libremente, bajo la triple licencia de Mozilla como un programa libre y de código abierto.

1.8 Servidor web

Un servidor Web es un programa que permite acceder a páginas Web contenidas en un ordenador, así como guardar los registros de acceso al sitio (logs) o páginas web contenidas en él.

Apache2.2.6 (13) (14): es un sistema de código abierto para plataformas Windows, Unix, Macintosh y otras que implementa el protocolo HTTP. Este servidor posee varias características que lo favorecen, como por ejemplo:

- Es una tecnología gratuita de código fuente abierta, lo cual le atribuye transparencia a este.
- Corre en varios Sistemas Operativos, lo que provoca que sea una herramienta prácticamente universal.
- Es un servidor altamente configurable de diseño modular. Permite aumentar fácilmente su capacidad e instalar cualquier módulo para cumplir una función específica. Otra cosa importante es que cualquiera que posea experiencia en la programación de C o Perl puede escribir un módulo para realizar una acción determinada.
- Permite personalizar la respuesta ante posibles errores. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

Tiene un alto nivel de configuración en la creación y gestión de logs. Apache permite la creación de ficheros de log facilitando, de este modo, el control de las acciones realizadas en el servidor.

1.9 Sistemas gestores de bases de datos

Los **Sistemas de Gestión de Base de Datos** (15) (16): son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. El propósito general de los SGBD es manejar de manera clara, sencilla y ordenada un conjunto de datos

que posteriormente se convertirán en información. Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información:** los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- **Independencia:** la independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Redundancia mínima:** un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- **Consistencia:** en aquellos casos en los que no se ha logrado que la redundancia sea casi nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, en otras palabras, que todos los datos repetidos se actualicen de forma simultánea.
- **Seguridad:** los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad:** se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. En otras palabras, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
- **Respaldo y recuperación:** los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- **Control de la concurrencia:** en la mayoría de los entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos y en muchos casos,

estos accesos ocurren de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

- Tiempo de respuesta: lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

1.9.1 PostgreSQL

PostgreSQL 8.3.4 (17) (18): es un sistema de gestión de base de datos relacional orientada a objetos de software libre. Posee varias ventajas como:

- Ágil navegación y administración de base de datos.
- Herramientas de manipulación de datos avanzada.
- Administración efectiva de seguridad.
- Excelentes herramientas visuales y de texto para elaboración de consultas.
- Acceso al servidor PostgreSQL a través del protocolo HTTP.
- Conexión vía reenvío de puerto local a través del túnel SSH.
- Impresionantes opciones de exportación e importación de datos.
- Poderoso diseñador visual de base de datos.
- Asistentes fáciles de usar que realizan mantenimiento de tareas de PostgreSQL.

1.10 Herramientas

1.10.1 NetBeans

NetBeans 6.8 (19): es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de

programación. Existe además un número importante de módulos para extender el Entorno de Desarrollo Integrado (IDE¹) NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

Principales características

- Nos provee de una estructura para los proyectos que podemos crear junto a este IDE, nos propone un esqueleto para organizar nuestro código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS. Además posee un sistema para examinar todos los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación.
- Integración con framework de PHP como Symfony y ZenFramework. Además brinda un entorno amigable para ejecutar comandos de Symfony.
- El editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP.
- NetBeans integra muy bien la utilización Xdebug, gracias a esto, podemos inspeccionar y examinar cada variable local, establecer puntos de interrupción y evaluar el código en nuestra lógica.
- El IDE de NetBeans para PHP también ofrece la línea de comandos de depuración: La salida del programa PHP aparece en una pantalla de línea de comandos en el IDE de sí mismo y se puede inspeccionar el código HTML generado sin tener que cambiar a un navegador.
- Integración de sistemas de control de versiones, tales como SVN, CVS, Mercurial y Git.
- Desde el editor es posible realizar la administración de estos sistemas versionados, sus commit, branch, importar, exportar, revert, clonar, etc.

1.10.2 WAMP (Wamp Server)

WAMP 2.0 (20) (21) (22): es un PC con Windows que dispone de un servidor Apache, un gestor de bases de datos MySQL y el lenguaje de programación PHP. Las siglas WAMP son un acrónimo de Windows +

¹ IDE acrónimo en inglés de Integrated Development Environment

Apache + MySQL + PHP. Esta aplicación instala una interfaz residente en la barra de tareas que permite iniciar, supervisar y detener los distintos servicios. Una de las ventajas de usar Wamp Server es que la instalación modificará los archivos de configuración (*.conf) con la ruta donde finalmente se ubicará el programa. También crea un directorio denominado 'www' que será la raíz para tus documentos. Un icono en la barra de tareas nos indicará al instante el estado de los diferentes servicios.

1.10.3 EMS SQL Manager para PostgreSQL

EMS SQL Manager para PostgreSQL 4.1.0.1 (23) (24): es una poderosa herramienta gráfica para la administración y, de manera general, para el trabajo con servidores PostgreSQL. Funciona con cualquier versión de PostgreSQL y soporta todas sus nuevas características, incluyendo espacios de tablas, argumentos nombrados en funciones y otras más. Ofrece una gran variedad de herramientas poderosas para usuarios avanzados, tales como diseñador visual de base de datos, constructor visual de consultas y un poderoso editor de objetos binarios para satisfacer todas sus necesidades. PostgreSQL Manager cuenta con una nueva y avanzada interfaz gráfica de usuario con un sistema asistente bastante descriptivo, tan claro en su uso que ni un principiante podría confundirse.

Características principales:

- Soporte completo de PostgreSQL.
- Nueva interfaz gráfica de usuario último modelo.
- Ágil navegación y administración de base de datos.
- Administración sencilla de todos los objetos PostgreSQL.
- Herramientas de manipulación de datos avanzada.
- Administración efectiva de seguridad.
- Acceso al servidor PostgreSQL a través del protocolo HTTP.
- Impresionantes opciones de exportación e importación de datos.
- Asistentes fáciles de usar que realizan mantenimiento de tareas de PostgreSQL.

1.11 Tecnologías y Framework

1.11.1 Symfony

Symfony 1.2.8 (25) (26): es un framework PHP que facilita el desarrollo de las aplicaciones web. Symfony se encarga de todos los aspectos comunes y aburridos de las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto.

Symfony, es además, el framework más documentado del mundo, ya que cuenta con miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales.

Características:

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).
- Compatible solamente con PHP 5 desde hace años, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Flexible hasta cualquier límite y extensible mediante un completo mecanismo de plugins.
- Publicado bajo licencia MIT de software libre y apoyado por una empresa comprometida con su desarrollo.
- Escalable: infinitamente escalable si se disponen de los recursos necesarios. Yahoo! utiliza Symfony para programar aplicaciones con 200 millones de usuarios.
- Probado con éxito durante años en varias aplicaciones gigantescas (Yahoo! Answers, Dailymotion, delicious) y en otros miles de sitios pequeños y medianos.
- Su código fuente incluye más de 9.000 pruebas unitarias y funcionales.

1.11.2 ExtJS

ExtJS 3.0.0 (27) (28): es una librería JavaScript que permite construir aplicaciones complejas en Internet.

Esta librería incluye:

- Componentes para la interfaz de usuario o UI² de alto performance y personalizables.
- Modelo de componentes extensibles.
- Una interfaz de programación de aplicaciones o API³ fácil de usar.
- Licencias open source y comerciales.

Principales características

- Nos permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto, provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).
- La ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla sólo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros.
- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- Necesita una plataforma. En este caso dependemos de ExtJS para mostrar los componentes y hacer el render de la aplicación.

² UI acrónimo en inglés de User Interface

³ API acrónimo en inglés de Application Programming Interface

- El JavaScript no es tan rápido como quisiéramos.
- Problemas con los motores de búsqueda. Los motores de búsqueda indexan el contenido web estático por lo que los textos cargados de manera dinámica no serán encontrados.
- Accesibilidad. Estas aplicaciones tienen problemas con los programas de accesibilidad, pues, al igual que los motores de búsqueda, no trabajan bien con texto cargado dinámicamente.
- No se pueden usar fuera de línea. Por su naturaleza web estas aplicaciones no pueden ser usadas en el cliente como cualquier otra aplicación.

1.11.3 AJAX

AJAX (29), acrónimo de Asynchronous JavaScript And XML (en inglés «JavaScript y XML asíncronos»). Técnica de desarrollo web para crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes:

- HTML (o XHTML) y Hojas de Estilo en Cascada (CSS) para presentar la información.
- Document Object Model (DOM) y JavaScript, para interactuar dinámicamente con los datos.
- XML y XSLT, para intercambiar y manipular datos de manera asíncrona con un servidor web (aunque las aplicaciones AJAX pueden usar otro tipo de tecnologías, incluyendo texto llano, para realizar esta labor).

Como el DHTML o LAMP, AJAX no constituye una tecnología en sí, pero es un término que engloba a un grupo de éstas que trabajan conjuntamente. Las aplicaciones AJAX usan navegadores web que soportan las tecnologías mencionadas más arriba. Entre estos se incluyen Mozilla, Firefox, Internet Explorer, Opera, Konqueror y Safari.

1.12 Metodología de desarrollo

RUP (30) (31): metodología llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- Inicio: el objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: en esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción: en esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- Transmisión: el objetivo es llegar a obtener el realce del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Los elementos del RUP son:

- Actividades: son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: vienen siendo las personas o entes involucrados en cada proceso.
- Artefactos: un artefacto puede ser un documento, un modelo, o un elemento de modelo.

El RUP presenta 3 características que constituyen la esencia de todo el proceso de desarrollo:

- Dirigido por los casos de uso.
- Centrado en la arquitectura.
- Ciclo de vida iterativo.

Características de RUP

- Reconoce que las necesidades del usuario y sus requerimientos no se pueden definir completamente al principio.
- Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema.
- Acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan para obtener resultados claros a corto plazo.

- Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración.

1.13 Características e importancia del software a crear

La propuesta posee gran flexibilidad a la hora del diseño de las consultas, ya que permite al usuario abstraerse de los conocimientos relacionados con SQL, pues el mismo trabaja con un entorno amigable en el cual la transparencia del proceso para el usuario es primordial. En el diseñador de reportes el usuario puede configurar a gusto como se mostrará, siendo una tarea sencilla en comparación con otras herramientas para el mismo fin.

La aplicación es portable, en el sentido, que puede ser ejecutada en cualquier versión del sistema operativo Linux que cuente con interfaz gráfica, Windows XP o superior. El usuario podrá confeccionar los reportes de forma dinámica, en otras palabras, tendrá la libertad de agrupar los datos del reporte, establecer un orden jerárquico según su conveniencia, así como introducir criterios para un atributo seleccionado.

1.14 Conclusiones de capítulo

En este capítulo se realizó un estudio de los sistemas de generación de reportes en Cuba y el mundo, entre los sistemas analizados se encuentran el Crystal Report, el Jasper Report y el generador de reportes del Akademos y, aunque todos presentan ventajas, ninguno se ajusta a las necesidades del sistema a implementar, porque se necesita una aplicación que permita generar reportes de manera dinámica y que sea de fácil entendimiento para personas sin conocimientos de BD. Se valoró la importancia del software a crear, fundamentando la necesidad de su desarrollo. Se caracterizaron las principales tecnologías, metodologías, frameworks, herramientas y lenguajes que son empleados en el desarrollo de la aplicación.

Capítulo 2: Descripción y análisis de la solución propuesta

2.1 Introducción

En este capítulo se abordará acerca de la implementación del sistema y la solución que se le propone en este trabajo de diploma, para lo cual, se realizará un valoración crítica del diseño propuesto por los analistas, teniendo en cuenta los requerimientos funcionales y no funcionales de la aplicación. Se hará referencia también a los componentes a reutilizar (clases o bibliotecas previamente desarrolladas). Además, se describirán los algoritmos no triviales a implementar y el análisis de la complejidad de los mismos, los estándares de codificación definidos, para finalmente describir las nuevas clases y operaciones necesarias para darle solución a la situación problemática.

2.2 Análisis de los diagramas

En este epígrafe se procederá a analizar los diagramas que intervienen en la solución propuesta. Los cuales tienen una gran importancia para la realización de la implementación del sistema.

2.2.1 Diagramas de Clases del Diseño

En los diagramas de clases del diseño (DCD), ver en Anexos: figura 2.1, figura 2.2 y figura 2.3, se refleja la estructura propuesta por los analistas. Estos cumplen con las especificaciones del patrón arquitectónico Modelo Vista Controlador (MVC) que implementa Symfony.

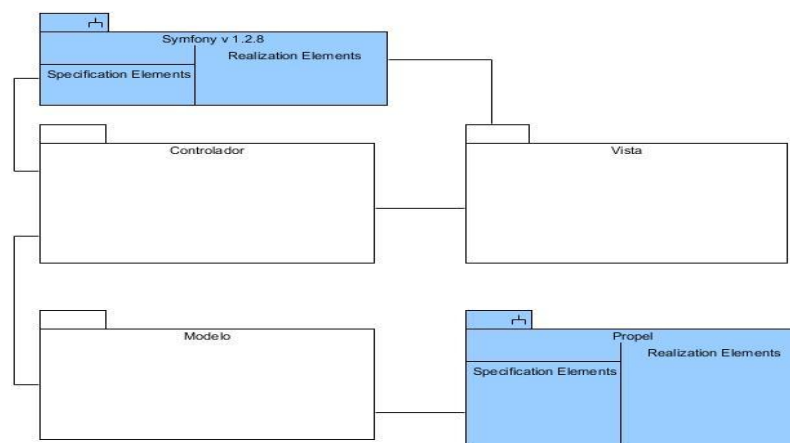


Figura 2.1: Patrón arquitectónico MVC que implementa Symfony.

En la figura anterior se muestran las tres capas del patrón MVC, este pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Los DCD propuestos, siguen el flujo del patrón, de manera que el usuario interactúe con la interfaz de la aplicación de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.), el controlador recibe (por parte de los objetos de la interfaz- vista) la notificación de la acción solicitada por el usuario, el controlador gestiona el evento que llega, accede al modelo, actualizando, eliminando o posiblemente modificando datos de este, de forma adecuada a la acción solicitada por el usuario, el modelo no debe tener conocimiento directo sobre la vista, luego se genera un mensaje en el controlador y finalmente el usuario recibe una notificación del éxito o fracaso de la acción indicada, la interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente. Para mantener este flujo es necesario que haya una coherencia entra la relación de las capas del patrón, asignando a cada clase las responsabilidades específicas de ella.

Posterior al análisis de la propuesta de diseño del analista, que tiene una gran repercusión, puesto que decide qué se asume y qué se modifica para la implementación del sistema, se analizará de igual manera los diagramas de componentes. La reutilización de componentes o módulos ya existentes, que pueden ser reutilizados, así como las estrategias de integración.

2.2.2 Diagramas de componentes

A continuación se muestran los diagramas de componentes propuestos. Los mismos están acorde con los requerimientos del sistema a implementar y con el patrón arquitectónico MVC propuesto por el framework Symfony.

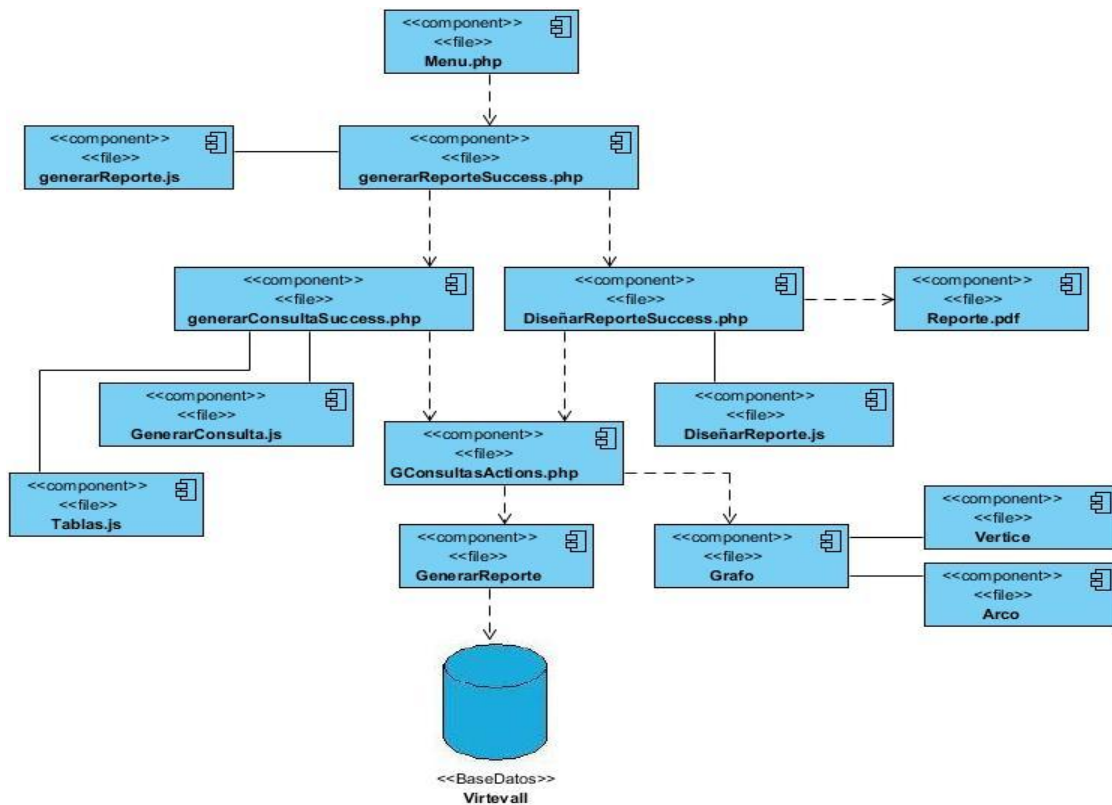


Figura 2.4 DC Generar Reporte

Posterior al análisis de la propuesta de diseño del analista, y de los diagramas de componentes, se analizará de igual manera, la reutilización de componentes o módulos ya existentes, que pueden ser reutilizados, así como las estrategias de integración.

2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes que puedan ser rehusados

Symfony

Anteriormente, se expuso que el framework de PHP seleccionado para el desarrollo de la aplicación fue Symfony, debido a que el mismo posee las características y funcionalidades necesarias que permitirían desarrollar de manera eficiente. Una de las características expuestas fue el uso del patrón de arquitectura Modelo-Vista-Controlador (MVC) que separa la lógica de negocio (el modelo) y la presentación (la vista), por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

MVC es una aproximación al software que separa la lógica de la aplicación de la presentación. En la práctica, permite que las páginas web contengan mínima codificación, ya que la presentación es separada del código PHP.

El *modelo* representa la información con la que trabaja la aplicación, en otras palabras, su lógica de negocio. Típicamente las clases del modelo contendrán funciones para recuperar, insertar y actualizar información en la BD. Symfony con el ORM brinda abstracción a la hora de acceder a los datos, ya que se manejan como objetos.

La *vista* transforma el modelo en una página web que permite al usuario interactuar con ella.

El *controlador* se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

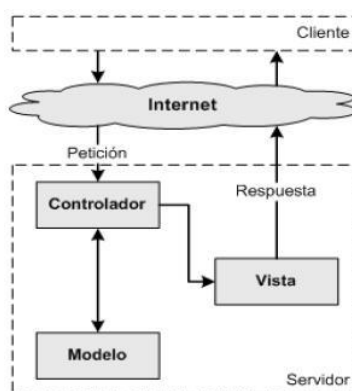


Figura 2.5 El patrón MVC.

2.3.1 La implementación del MVC que realiza Symfony

El controlador frontal y el layout son comunes para todas las acciones de la aplicación, se pueden tener varios controladores y varios layouts, con la posibilidad de cambiarlos mediante la configuración del framework, esto es más utilizado con los layout, ya que muchas veces se hace necesario tener varios diseños de interfaces, pero solamente es obligatorio tener uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática.

Las clases de la capa del modelo también se generan automáticamente, en función de la estructura de datos de la aplicación. La librería Propel se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario, sólo es necesario crear las consultas manejando objetos y el framework se encarga del acceso a la BD, este acceso es completamente invisible al programador, porque lo realiza un componente específico llamado Creole. Así, si se cambia el SGBD en cualquier momento, no se debe reescribir ni una línea de código, ya que tan sólo es necesario modificar un parámetro en un archivo de configuración.

La lógica de la vista se puede transformar en un archivo de configuración sencillo, sin necesidad de programarla.

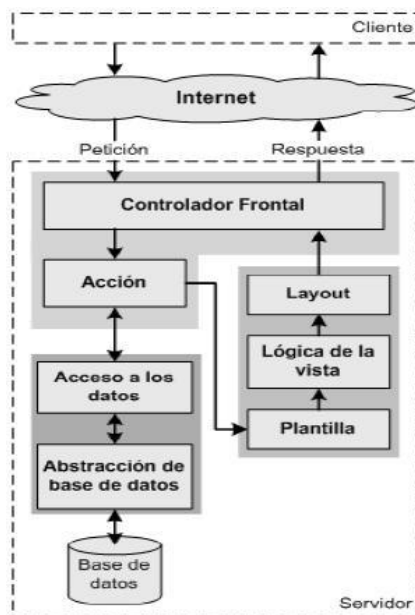


Figura 2.6 El flujo de trabajo de Symfony

2.4 Funciones y clases reutilizados

Método Json

Gestiona la conversión de cadenas Json para uso en el JavaScript. Json es un formato de intercambio de datos creado a partir de un subconjunto de la notación de literales de objetos en JavaScript. Aunque JavaScript acepta una sintaxis para valores literales muy flexible, es importante tener en cuenta que JSON posee reglas mucho más estrictas, en versiones anteriores a php 5.2 el lenguaje no traía esta

función implementada, había que crearla si se deseaba utilizar, ya en las versiones posteriores el lenguaje incorpora esta funcionalidad.

Clase Vértice

Permite modelar disímiles objetos del mundo real, esta clase es mayormente utilizada como complemento de la clase Grafo para modelar uno de los elementos de este, estos vértices pueden ser objetos, personas, etc. Cada uno de estos objetos tiene sus atributos, los cuales lo definen como un objeto único.

Clase Grafo

Permite modelar infinidad de sistemas del mundo real en los que diferentes elementos de un conjunto están relacionados entre sí: ciudades conectadas por carreteras, proyectos divididos en tareas que dependen unas de otras, relaciones familiares en diagramas genealógicos, aeropuertos conectados por vuelos directos, etc. En este caso se utiliza para modelar las relaciones entre las tablas de la BD en las cuales se realizaran las consultas. Para modelar los nodos o vértices se utiliza la clase Vértice, estos vértices serían, para este caso en particular: las tablas.

Existen varias implementaciones de esta clase, algunas de ellas utilizan la clase Arco para definir las relaciones entre los objetos, en este caso, se utilizó la clase con una matriz de adyacencia que es una equivalencia a lo que se hace al utilizar la clase Arco.

Floyd-Warshall

Este algoritmo se utiliza para hallar la matriz de predecesores con la cual se tienen todos los posibles caminos que pueden existir entre las tablas, tomando como peso para las aristas del grafo la cantidad de tuplas de las tablas, lo que se utiliza para el cálculo del camino mínimo.

Librería FPDF

FPDF es una biblioteca escrita en lenguaje de programación PHP que permite crear archivos en formato PDF sin ningún requerimiento adicional. Ofrece distintas funcionalidades como la elección de la unidad de medida, formato de página y márgenes, gestión de cabeceras y pie de página, si las páginas van a tener salto de página automático, salto de línea y justificación del texto automático, admisión de imágenes, colores y enlaces.

Está desarrollado con orientación a objetos, siendo el Objeto FPDF el encargado de ir almacenando la estructura, y mostrándolo con la función Output, teniendo diferentes salidas, tanto por pantalla como por impresora o simplemente ofreciendo la posibilidad de descargar el archivo. FPDF ofrece la ventaja de permitir crear PDF desde PHP con relativa sencillez, entre sus funciones más utilizadas se encuentra Cell(), que es la base de todo el muestreo, creando celdas, las cuales, pueden contener texto.

Se tiene ya una versión casi definitiva del diseño. Se ha decidido qué componentes utilizar y qué estrategia de integración tomar en consideración durante la implementación del sistema. Por tanto, en el epígrafe siguiente se analizarán los algoritmos que dan solución a algunos problemas, los cuales cobran cierto nivel de complejidad.

2.5 Descripción de los algoritmos no triviales a implementar. Análisis de complejidad de los mismos

Los algoritmos que se analizan a continuación tienen un nivel de complejidad superior a la media, por lo que se hace necesario su estudio, debido a que brindan la solución a problemáticas claves en la implementación del sistema.

Floyd-Warshall

El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución. Este compara todos los posibles caminos a través del grafo entre cada par de vértices. El algoritmo es capaz de hacer esto con sólo V^3 comparaciones (esto es notable considerando que puede haber hasta V^2 aristas en el grafo, y que cada combinación de aristas se prueba, siendo V la cantidad de vértices del grafo). Lo hace mejorando paulatinamente una estimación del camino más corto entre dos vértices, hasta que se sabe que la estimación es óptima.

Construir Consulta

Este método tiene una complejidad temporal de N^3 , ya que hace llamada a varios algoritmos auxiliares que hacen una búsqueda exhaustiva en el arreglo pasado por parámetro para construir la consulta, tomando de este los valores necesarios para concatenar, además hace llamada al método encargado de leer el archivo XML con la lista de las tablas y por último al encargado de realizar las consultas después que la llamada a los métodos auxiliares fuera completando la cadena que compone la consulta en sí.

Leer XML con Lista Tablas

Este método se encarga de leer el archivo *Lista_Tablas.xml* con la librería SimpleXML de PHP, este archivo contiene todas las tablas definidas para realizar consultas en la base de datos, después de realizar la lectura es encargado de crear el grafo con el contenido del xml y mandar a ejecutar el algoritmo Floyd-Warshall, este método sólo se llama, si en la consulta que va a realizar el usuario está involucrada más de una tabla, sino no es necesario hacer una llamada a este en el mejor de los casos, pero si se ejecuta, la complejidad temporal del mismo es de N^3 .

2.6 Descripción de las nuevas clases u operaciones necesarias

Siguiendo el enfoque MVC del Symfony, las nuevas clases a describir serán de tipo Controladora, Modelo o Vista.

Nombre: GConsultasActions	
Tipo de clase: Controladora	
Atributos:	Tipo:
Funciones:	
Nombre:	executeGenerarConsulta
Descripción:	Método que ejecuta la vista correspondiente.
Nombre:	executeObtenerDatosConsulta
Descripción:	Método que obtiene los datos de la consulta especificada por el usuario en la vista.
Nombre:	executeResultados
Descripción:	Método que se encarga de mostrar los resultados de la consulta realizada por el usuario.
Nombre:	executeExportarPDF
Descripción:	Método que se encarga de llamar a las funcionalidades necesarias para exportar los resultados de la consulta a formato PDF.

Tabla 2.1: Clase Controladora GConsultasActions.

Nombre: generarConsultaSuccess	
Tipo de clase: Vista	
Atributos:	Tipo:

Funciones:	
Nombre:	
Descripción:	

Tabla 2.2: Clase Vista generarConsulta.

Nombre: indexSuccess	
Tipo de clase: Vista	
Atributos:	Tipo:
Funciones:	
Nombre:	
Descripción:	

Tabla 2.3: Clase Vista index.

Nombre: resultadosSuccess	
Tipo de clase: Vista	
Atributos:	Tipo:
Funciones:	
Nombre:	
Descripción:	

Tabla 2.4: Clase Vista resultados.

Nombre: exportarPDFSuccess	
Tipo de clase: Vista	
Atributos:	Tipo:
Funciones:	
Nombre:	
Descripción:	

Tabla 2.5: Clase Vista exportarPDF.

Nombre: Vértice	
Tipo de clase: Modelo	
Atributos:	Tipo:
\$nombre	string
\$alias	string
\$id	string
\$llavesforaneas	Array()
\$peso	int

Funciones:	
Nombre:	Vertice(\$n, \$a, \$p, \$lf, \$i)
Descripción:	Constructor de la clase, recibe los parámetros que inicializan el objeto.
Nombre:	Nombre()
Descripción:	Método que retorna el nombre de la tabla.
Nombre:	Alias()
Descripción:	Método que retorna el alias de la tabla.
Nombre:	LlavesForaneas()
Descripción:	Método que retorna las llaves foráneas de la tabla.
Nombre:	Peso()
Descripción:	Método que retorna el peso de la tabla (como peso se toma la cantidad de tuplas).
Nombre:	Id()
Descripción:	Método que retorna la llave primaria de la tabla.

Tabla 2.6: Clase Modelo Vértice.

Nombre: Grafo	
Tipo de clase: Modelo	
Atributos:	Tipo:
\$listaVertices	Array()
\$matrizDistancia	Array(Array())
\$numeroVertices	int
\$matrizPredecesores	Array(Array())
\$listaCaminos	Array(Array())
\$costoCaminos	Array(Array())
\$tmp	Array()
Funciones:	
Nombre:	Grafo(\$lv, \$cm)
Descripción:	Constructor de la clase grafo, recibe los parámetros para inicializar las variables.
Nombre:	ListaVertices()
Descripción:	Método que retorna la lista de vértices del grafo.
Nombre:	ListaCaminos()
Descripción:	Método que retorna la lista de caminos del grafo.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN

Nombre:	MatrizDistancia()
Descripción:	Método que retorna la matriz de distancias del grafo.
Nombre:	NumeroVertices()
Descripción:	Método que retorna el número de vértices del grafo.
Nombre:	FloydWarshall()
Descripción:	Implementación actual del método Floyd-Warshall, de él se obtiene la matriz de distancias mínimas de todos a todos los vértices del grafo y la matriz de predecesores.
Nombre:	_obtenerRuta(\$i, \$j)
Descripción:	Método auxiliar para obtener una ruta de un vértice a otro.
Nombre:	ObtenerRuta(\$i, \$j)
Descripción:	Método que retorna la ruta del vértice \$i hasta el vértice \$j que recibe como parámetros.
Nombre:	ObtenerDistancia(\$i, \$j)
Descripción:	Método que retorna la distancia entre los vértices \$i, \$j que recibe como parámetros.
Nombre:	LlenarListaCaminos()
Descripción:	Método que obtiene todos los posibles caminos del grafo.

Tabla 2.7: Clase Modelo Grafo.

Nombre: CrearConsultas	
Tipo de clase: Controladora	
Atributos:	Tipo:
\$grafo	Grafo
Funciones:	
Nombre:	__construct()
Descripción:	Constructor por defecto de la clase.
Nombre:	getGrafo()
Descripción:	Método que retorna el atributo grafo de la clase.
Nombre:	construirConsulta(\$dC)
Descripción:	Método que se encarga de construir la consulta con los datos seleccionados por el usuario.
Nombre:	concatenarSelect(\$variablesSelect)
Descripción:	Método auxiliar para construir la consulta, se encarga de crear los select de la misma.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN

Nombre:	concatenarTablasJoin(\$t)
Descripción:	Método encargado de concatenar los <i>INNER JOIN</i> si es necesario para la consulta
Nombre:	ordenarCampos(\$arrego)
Descripción:	Método auxiliar que ordena los campos para después crear las condiciones de la cláusula <i>WHERE</i>
Nombre:	crearWhere(\$datosConsultaWhere)
Descripción:	Método encargado de concatenar las condiciones después de haber sido ordenadas por el método <i>ordenarCampos</i>
Nombre:	auxiliarCrearWhere(\$conds)
Descripción:	Método auxiliar de <i>crearWhere</i> , se hizo una separación del método principal para optimizar, esto se logra haciendo varias llamadas con distintos datos a este auxiliar.
Nombre:	concatenarOrderBy(\$datosConsulta)
Descripción:	Método encargado de buscar si existe algún campo por el que el usuario quiera ordenar ascendente o descendente y concatenar las cadenas necesarias, con los campos correspondientes.
Nombre:	concatenarConsulta(\$stringSelect, \$stringJoin, \$stringWhere, \$stringOrderBy)
Descripción:	Este método recibe por parámetros distintas cadenas las cuales une y conforma la consulta diseñada por el usuario.
Nombre:	buscarDatosConsulta(\$consulta)
Descripción:	Este método recibe como parámetro una cadena que es el retorno del método <i>concatenarConsulta</i> y se encarga de realizar la consulta a la base de datos retornando el resultado.
Nombre:	leerXMLListaTablas()
Descripción:	Método encargado de leer el archivo XML que contiene los parámetros de las tablas para construir el grafo.
Nombre:	auxiliarContarTuplas(\$tabla)
Descripción:	Método auxiliar que se encarga de hacer una consulta para obtener la cantidad de tuplas de una tabla que recibe por parámetro.

Tabla 2.8: Clase Controladora Crear Consultas.

Nombre: PDF	
Tipo de clase: Controladora	
Atributos:	Tipo:

Funciones:	
Nombre:	Header()
Descripción:	Método que redefine el método Header que trae por defecto la librería FPDF.
Nombre:	Footer()
Descripción:	Método que redefine el método Footer que trae por defecto la librería FPDF.
Nombre:	buscarHeader(\$datosGrid)
Descripción:	Método que se encarga de buscar los encabezados de las columnas de la tabla de acuerdo con el alias definido por el usuario.
Nombre:	calcularTamanoCeldas(\$numero)
Descripción:	Método que se encarga de calcular el tamaño de las celdas de acuerdo al tamaño de página y la cantidad de columnas necesitada para los datos.
Nombre:	ConstruirTabla(\$header, \$data, \$tamanoCell)
Descripción:	Método que se encarga de construir la tabla de forma dinámica de acuerdo a los parámetros resultantes en la consulta y los parámetros esperados como encabezado de las columnas (alias), los datos y un arreglo con el tamaño de las celdas.

Tabla 2.9: Clase Controladora PDF.

Las clases relacionadas anteriormente son las clases que componen las nuevas funcionalidades implementadas en la aplicación, responsables de la creación de las vistas y objetos necesarios para poder realizar las consultas y generar los reportes, para esto una de ellas particularmente tiene que leer un fichero .xml donde se encuentra la información relacionada con las tablas de la BD para poder construir un grafo, este refleja las relaciones entre tablas, este grafo solo se construye si es necesario, si la consulta tiene al menos dos tablas incluidas, sino, la se buscan los datos sin necesidad de utilizar algunos algoritmos más complejos.

Se encuentra la clase responsable de convertir los resultados de la consulta a formato PDF, si el usuario lo desea, esta clase hereda de la biblioteca FPDF, la cual es muy utilizada en el por los programadores cuando existe necesidad de exportar a este formato.

2.7 Patrones de Diseño utilizados por el framework Symfony

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. (32)

Los patrones de diseño se encuentran agrupados en dos grandes grupos fundamentalmente, estos son:

- ✓ Patrones GRASP⁴
- ✓ Patrones GoF⁵

Un grupo de desarrollo bien estructurado define cuáles serán los patrones de diseño a utilizar en la implementación del sistema. El uso de estos, hace que el sistema cuente con una arquitectura robusta y de fácil mantenimiento. El framework propuesto para el desarrollo de la aplicación utiliza muchos de estos patrones a continuación se relacionan algunos.

2.7.1 Patrones GRASP

- **Experto:** Este es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.
- **Alta Cohesión:** En cada clase Actions se definen las acciones para las plantillas, además, estas colaboran con otras para realizar diferentes operaciones, se instancian objetos, se acceden a las properties, en otras palabras, en una Actions se utilizan diferentes funcionalidades estrechamente relacionadas entre sí, lo que proporciona un software flexible ante los cambios.

⁴ GRASP es el acrónimo en inglés de "General Responsibility Assignment Software Patterns".

⁵ GoF es el acrónimo en inglés de "Gang of Four".

- **Bajo acoplamiento:** Este patrón está evidenciado en el framework ya que dentro de la capa modelo las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa vista ni con el controlador.
- **Controlador:** Todas las peticiones Web son manejadas por un sólo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado.
- **Creador:** Cada método perteneciente a una vista en las clases Actions tiene la responsabilidad de crear instancias de las clases mapeadas del modelo, a las cuales necesita acceder por que tiene datos que requiere su constructor o necesita acceder a sus datos. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos.

Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

2.7.2 Patrones GoF

En la categoría Creacionales:

- **Singleton** (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En las acciones se usan los métodos getRequest(), getUser(), esto se debe a que, en la acción, el método getContext(), guarda una referencia a todos los objetos del núcleo de Symfony, estos métodos pueden ser accedidos desde la vista y desde el controlador, sólo varía la forma de llamarlos.

En la categoría Estructurales:

- **Decorator** (Envoltorio): Añade funcionalidad a una clase, dinámicamente. En cada archivo layout.php se define el código HTML común de cada vista, evitando que este sea repetido en cada página.

2.8 Estándares de Codificación

Para facilitar el entendimiento del código y fijar un modelo a seguir, se establecieron estándares de codificación para todos los programadores. Estos estándares consisten en estilos de codificación a la

hora de escribir el código. Los aspectos para los que generalmente se establecen estándares son los siguientes:

- Identificadores.
- Indentación.
- Llaves
- Líneas y espacios en blanco.
- Comentarios.

En cada grupo de desarrollo se definen cuáles serán los aspectos a estandarizar y qué estilos se aplicarán a cada uno de ellos. El cumplimiento de estándares hace que todo el código lleve el sello personal del programador y en caso de ser varios los programadores, pues se busca que todo el código parezca que ha sido implementado por la misma persona. De esta manera se consigue mayor legibilidad y facilidad de mantenimiento. Los estándares deben responder además a acciones prácticas que acomoden al programador. A continuación se definirá que estándares usar para la actividad de implementación correspondiente a este trabajo. Cabe destacar que estos estándares se definen teniendo en cuenta el estilo personal del programador, las características propias del lenguaje de programación, los recursos que se utilizarán y el tipo de programa que se debe implementar.

2.8.1 Identificadores

En el caso de los identificadores existen estilos definidos mundialmente como el lowerCamelCase⁶ y el UpperCamelCase. Cada palabra interna en identificadores compuestos comienza con mayúsculas para ambos estilos, además ocurre, que no se colocan caracteres de separación entre las palabras que conforman un identificador compuesto en ninguno de los dos casos. Para el primero, el identificador comienza con minúscula y para el segundo, el identificador comienza con mayúscula.

- Espacio de nombre: para los espacios de nombres se escogió el UpperCamelCase.

⁶CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre CamelCase se podría traducir como Mayúsculas/Minúsculas Camello, aunque no es correcto en todos los contextos ya que la palabra inglesa Case no tiene traducción literal. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello.

- Clase: para las clases se escogió el UpperCamelCase.
- **Class** GConsultaActions **extends** sfActions
- Función: para las funciones se escogió el lowerCamelCase.
- **public function** getIdUsuario()
- Variable
- **protected** \$id_usuario;

2.8.2 Indentación

La indentación es una práctica de programación que consiste en comenzar a escribir cada línea de código a diferentes distancias desde el borde izquierdo del área de texto del editor. Esta distancia está determinada por la jerarquía que se forma al introducir sentencias dentro de bloques de estructuras. Esto brinda mayor legibilidad y entendimiento para el programador e igualmente depende del propio estilo de cada persona, del lenguaje y tipo de programa que se implementa. Se definió que la indentación se hará agregando dos <tab> al inicio de la línea que se desee escribir. Se escribirá sólo una sentencia por línea de código y en el caso de cortar las líneas, se hará luego de una coma o antes de un operador. La sección de la derecha de la línea que se corte se ubicará en la línea siguiente indentada al nivel de la expresión correspondiente en la línea superior.

```
36 class Arco(  
37  
38     var $verticeOrigen;  
39     var $verticeDestino;  
40  
41     public function Arco($vO, $vD, $p)  
42         (...)  
43  
44     public function VerticeOrigen()  
45         (...)  
46  
47     public function VerticeDestino()  
48         (...)  
49  
50     public function PesoVertice()  
51         (...)  
52  
53 )
```

Figura 2.7 Ejemplo de indentación.

2.8.3 Llaves

Existe diversidad de criterios en cuanto a la ubicación de las llaves que delimitan el cuerpo de los bloques de código en los lenguajes que contienen este tipo de estructuras. Algunos programadores prefieren hacerlo ubicando la llave de apertura inmediatamente detrás de la línea cabecera del bloque, mientras otros apuestan por ubicarlas de forma solitaria en la línea siguiente a la línea cabecera. Para este último estilo existen además diferencias en cuanto al nivel de indentación de las mismas. Algunos lo hacen al nivel de la línea cabecera y otros al nivel de las líneas del cuerpo del bloque. Para este trabajo: Las llaves de apertura de las funciones se colocarán solitarias en la línea siguiente e indentadas al nivel de la línea cabecera del bloque y las llaves de aperturas de las estructuras *for*, *if*, *while*, *else*, se colocarán inmediatamente detrás de la línea cabecera del bloque. Las llaves de cierre se colocarán solitarias en la línea que sigue a la última línea dentro del bloque e indentadas al nivel de la línea cabecera del bloque. Es prudente señalar que este estilo agrega más líneas de código al programa al ubicar las llaves solitarias en una línea, pero a su vez se gana en legibilidad del código.

```
48 public function VerticeOrigen()
49 {
50     $archivo='Lista_Tablas.xml';
51     $listaVertices = array();
52     if(file_exists($archivo)) {
53         $xml = simplexml_load_file($archivo);
54         if($xml) {
55             foreach ($xml->tabla as $tabla) {
56                 $n = $tabla->nombre;
57                 $a = $tabla->alias;
58                 $listaVertices[] = new Vertice($n,$a, "10", $tabla->llaves_foraneas);
59             }
60         } else {
61             echo "Sintaxis XML invalida";
62         }
63     } else {
64         echo "Error abriendo $archivo";
65     }
66 }
```

Figura 2.8 Ejemplo de uso de llaves.

2.8.4 Líneas y espacios en blanco

Para mejorar la legibilidad y organización del código muchas veces se utilizan líneas en blanco para separar segmentos de código que pueden corresponder a clases, funciones, declaraciones, implementaciones, comentarios, bloques o sencillamente secciones críticas que se deseen despejar. Así mismo sucede con los espacios en blanco cuando se utilizan para separar elementos dentro de las sentencias de código. En ocasiones se separan con espacios cada operador de su respectivo operando, paréntesis, identificadores, símbolos y algunos lenguajes exigen que se separen las palabras propias del vocabulario de las adyacentes para ser comprendidas por los compiladores. En este trabajo se ha definido emplear líneas en blanco:

- Entre funciones.

```
48 | public function VerticeOrigen()
49 | {
50 |     return $this->verticeOrigen;
51 | }
52 |
53 | public function VerticeDestino()
54 | {
55 |     return $this->verticeDestino;
56 | }
57 |
58 | public function PesoVertice()
59 | {
60 |     return $this->pesoVertice;
61 | }
```

Figura 2.9 Ejemplo de línea en blanco entre funciones

- Entre declaraciones de variables e implementaciones dentro del cuerpo de las funciones.

```
7 | private $nombre;
8 | private $alias;
9 | private $llavesforaneas;
10 | private $peso;
11 |
12 | public function Vertice($n, $a, $p, $llf){
13 |     $this->nombre = $n;
```

Figura 2.10 Ejemplo de espacio entre declaraciones de variables e implementaciones.

Se colocarán espacios en blanco:

- Entre las palabras reservadas y los elementos adyacentes a las mismas.
- `protected $id_usuario;`
- Después de las comas en la lista de argumentos de las funciones.

```
41 | public function Arco($vO, $vD, $p)
42 | {
```

Figura 2.11 Ejemplo de espacios en blanco en las listas de argumentos de las funciones.

- Después de cada punto y coma (;) en las estructuras **for**.

```
for ($i = 0; $i < count($ids); $i++) {
    $incisos[$i] = TincisoEncuestaPeer::ObtenerInciso($ids[$i]);
}
```

Figura 2.12 Ejemplo de espacio en blanco en las estructuras for.

2.8.5 Comentarios

El uso de comentarios durante la codificación ha demostrado que es beneficiosa por varias razones:

- Ayuda al programador a entender cada elemento o sección de código.
- Hace más fácil el proceso de adaptación del código durante su reutilización.
- Sirve de guía en los casos en que varios programadores trabajen sobre las mismas secciones del código.
- Disminuye el esfuerzo de análisis ya que el lenguaje natural es más legible que cualquier lenguaje de programación.

Todo esto se aprecia claramente cuando se escribe gran cantidad de código en largos intervalos de tiempo donde generalmente el o los programadores olvidan lo que se pensó en un momento. Los comentarios se pueden utilizar para varios fines:

- Para explicar el propósito de las funciones.
- Para explicar las características fundamentales de las clases.
- Para sintetizar las acciones de los algoritmos complejos.
- Para aclarar los datos que representan las variables.
- Para dividir secciones de código en dependencia de los diferentes contextos y funciones.
- A veces se usan comentarios temporales para recordar cosas que faltan, cosas que se deben modificar o analizar en otro momento.

Es necesario tener en cuenta algunos detalles al escribir comentarios:

- La capacidad de síntesis.

- El uso de lenguaje técnico.
- No repetir exactamente paso por paso lo que hace el algoritmo sino expresar un resumen de su propósito.
- Usar un estilo uniforme de comentario definido en estándares para todo el equipo.

En este trabajo se decidió colocar los comentarios encima de la línea a la que se le quiera aplicar y encima de la línea cabecera de los bloques. La indentación se hará al nivel de la línea en cuestión. Se utilizarán en funciones y clases. Se puede utilizar en algoritmos no triviales y secciones de diferentes contextos dentro de los métodos.

2.9 Conclusiones del capítulo

Después de analizar y realizar una valoración crítica del diseño propuesto por los analistas, se arribó a un diseño acorde con las restricciones del sistema, que deben tenerse en consideración a la hora de implementarlo.

Se realizó un análisis a un conjunto de métodos, clases y librerías previamente implementadas a reutilizar, que repercuten en la dinámica de la implementación porque brindan muchas facilidades.

Se estudiaron los algoritmos no triviales presentes en la solución, realizándole a cada uno un análisis de su complejidad y se describieron las nuevas clases y funciones necesarias para solucionar el problema. Todos los aspectos tratados tienen repercusión directa en el resultado final, la aplicación.

Capítulo 3: Validación de la solución propuesta

3.1 Introducción

En el presente capítulo se realiza la validación de la solución propuesta, a través de diferentes pruebas de unidad. Dentro de estas pruebas, están las pruebas de caja blanca y pruebas de caja negra, definiéndose el objetivo de cada una, así como su alcance y detalles de las mismas.

La prueba del software es un elemento fundamental para garantizar la calidad del mismo, esta representa una última revisión de las especificaciones del diseño y de la codificación. El objetivo de esta etapa es garantizar la calidad del producto terminado. Es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

3.2 Diseño de las pruebas unitarias y funcionales

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto (33).

- *Interfaz:* se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada.
- *Estructuras de datos locales:* se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente, conservan su integridad durante todos los pasos de ejecución del algoritmo.
- *Condiciones límites:* se prueban las condiciones límites para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento.

- *Caminos independientes*: se ejercitan todos los caminos independientes de la estructura de control para asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez.
- *Caminos de manejo de errores*: se prueban todos los caminos de manejo de errores.

Las pruebas unitarias son capaces de aislar una parte del código de manera que pueda ser analizado. Un ejemplo de esto es evaluar las funciones o métodos, a los cuales se les realiza una entrada con distintos juegos de datos para obtener los datos de salida correctos y los posibles caminos que pueden tomar algunos algoritmos con dichos datos. Este tipo de pruebas validan la forma en la que los algoritmos, funciones y métodos trabajan en cada caso particular. Las pruebas funcionales no sólo validan la transformación de una entrada en una salida, sino que validan a toda una característica completa. De modo que las pruebas funcionales validan procesos y requieren de un escenario. En Symfony se deberían crear pruebas funcionales para todas las acciones. Las pruebas deben simular la navegación del usuario, realizar peticiones y comprobar los elementos de la respuesta tal y como lo haría manualmente un usuario para validar que una determinada acción hace lo que se supone que debe hacer. En las pruebas funcionales se ejecuta un escenario correspondiente a lo que se denomina un *caso de uso*.

3.3 Tipos de pruebas de software

Para el cliente la calidad de un sistema está fundamentalmente determinada, entre otras cosas, por la coincidencia entre lo que se programó y los requisitos establecidos en las entrevistas con dicho cliente. Para comprobar el grado de cumplimiento de estos requisitos se usan las pruebas del sistema. La prueba de unidad es la prueba enfocada a los elementos probables más pequeños del software, consiste en una prueba estructural (*pruebas de caja blanca*), esta requiere conocer el diseño interno del producto puesto que verifica la lógica interna y el flujo de datos; y una prueba de especificación (*pruebas de caja negra*), basada sólo en la especificación del comportamiento externamente visible de la unidad. Con estas se comprueban las funcionalidades, diseñando casos de prueba que definen cómo proceder. Estos casos de prueba incluyen los juegos de datos a usar que son los válidos o esperados y los no válidos o no esperados por el programa. Las pruebas se deben aplicar durante todo el ciclo de vida del software e invariablemente se le debe dedicar una gran parte del esfuerzo total del desarrollo. Se deben planificar

correctamente desde el inicio y establecer qué hacer, cómo hacer, quién va a hacer y en qué condiciones hacer las comprobaciones.

El objetivo de los casos de prueba es forzar al máximo el sistema en los puntos críticos para encontrar fallos y detectar defectos. Se debe tener en cuenta muchos aspectos para escoger los tipos de pruebas que se adapten mejor al producto que se va a probar, ejemplo de estos son: el lenguaje de programación, las características de los desarrolladores, el proceso de desarrollo, el tipo de funcionalidad que se implementó, la plataforma de desarrollo, los principales errores, si la aplicación realiza conexiones a bases de datos, si es web o de escritorio, entre otros.

3.3.1 Pruebas de Caja Blanca

Son denominadas Pruebas de Caja Blanca porque revisan la parte interna del software, específicamente sobre el código fuente. Se basan en el examen minucioso de los detalles procedimentales. Se comprueban los caminos lógicos del sistema generando casos de prueba que ejerciten las estructuras condicionales y los bucles. Existen varios métodos que analizan diferentes partes del programa y se complementan entre sí para garantizar la calidad del sistema. Un resultado fundamental de estas pruebas es la complejidad ciclomática, esta acota la cantidad mínima de casos de prueba que se deben ejecutar. La técnica del camino básico permite diseñar casos de prueba para cubrir todas las sentencias de un programa a partir de la obtención de un conjunto de caminos independientes.

La prueba de las Condiciones es un método que se encamina hacia la ejercitación de las condiciones. Se basa en el principio de que si un conjunto de casos de prueba es capaz de ejercitar todas las condiciones contenidas en un bloque de código, este mismo conjunto serviría para encontrar más errores en el programa que no tengan que ver directamente con las condiciones. La prueba del Flujo de Datos verifica la validez en el uso de las variables para manipular los datos de la aplicación. La prueba de los Bucles se centra en la validez de las estructuras cíclicas o bucles. El objetivo es probar el comportamiento de estas estructuras en sus valores límites de iteración. Los bucles se clasifican en cuatro tipos: Bucles simples, Bucles anidados, Bucles concatenados y Bucles no estructurados. Aunque la esencia es la misma, cada tipo se prueba de forma diferente.

Es posible pensar que con las pruebas de Caja Blanca se logra detectar y corregir todos los errores de la aplicación, pero estas cuentan con una desventaja, ya que resulta muy costoso en tiempo y recursos lograr abarcar todo el código fuente de un sistema no muy grande.

3.3.2 Pruebas de Caja Negra

Las Pruebas de Caja Negra deben su nombre a los elementos que estas revisan y bajo qué condiciones se hace la revisión y desde el punto de vista de las entradas que recibe el elemento que es estudiado y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Estas se basan en los requerimientos funcionales del sistema y se llevan a cabo desde el exterior de la aplicación.

De una caja negra nos interesará su forma de interactuar con el medio que le rodea (en ocasiones, otros elementos que también podrían ser cajas negras) entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace. Por tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento.

Este tipo de prueba es importante a la hora de medir el grado de cumplimiento de los requerimientos solicitados por el cliente y se aplican sobre la interfaz de la aplicación observando las respuestas del sistema ante determinadas acciones y los datos de salida para determinados juegos de datos de entrada.

3.4 Herramientas para la realización de pruebas

Cuando se emplea php como lenguaje de programación existe diversidad de frameworks para crear pruebas unitarias y funcionales, siendo los más usados PHPUnit y SimpleTest. Symfony incluye su propio framework para pruebas unitarias llamado Lime. El mismo utiliza la librería Test::More de Perl y es compatible con TAP. Lo que significa que los resultados de las pruebas se muestran con el formato definido en el "Test Anything Protocol", creado para facilitar la lectura de los resultados de las pruebas. El Lime presenta además una serie de ventajas dentro de las cuales resaltan por su importancia las siguientes (34):

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas.
- Las pruebas son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados utilizan diferentes colores para mostrar de forma clara la información más importante.
- Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo llamado lime.php y no tiene ninguna dependencia.

3.5 Diseño de Casos de Prueba de Caja Blanca aplicados

A continuación se muestra la prueba del camino básico aplicada a un método de la clase CrearConsulta.php para la cual se siguieron una serie de pasos lógicos.

- Se construye el grafo de flujo a partir del código fuente del método a probar.
- Se determina la complejidad ciclomática (V) del grafo (G). Se puede calcular de tres formas:
 - ✓ $V(G)$ coincide con el número de regiones del grafo de flujo (internas y externa).
 - ✓ $V(G) = A - N + 2$, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
 - ✓ $V(G) = P + 1$, donde P es el número de nodos predicado contenidos en el grafo de flujo. Un nodo predicado es cada nodo que contiene una condición y se caracteriza gráficamente porque dos o más aristas emergen de él.
- Se construye una secuencia de $V(G)$ caminos linealmente independientes en G .
- Se prepara un caso de prueba por camino hallado en el paso anterior.
 - ✓ Determinar los datos a proporcionar como entrada para ejecutar el camino hallado.
 - ✓ Usando la especificación funcional del método, indicar cuál es el resultado esperado.

Descripción del método concatenarOrderBy

Permite crear la cadena de la cláusula ORDER BY de la consulta.

```
private function concatenarOrderBy($datosConsulta)
{
    $stringOrderBy = '';
    $orden = '';
    $pos = null;
    for ($k = 0; $k < count($datosConsulta); $k++) {
        if (($datosConsulta[$k][3] == 'Ascendente' || ($datosConsulta[$k][3] == 'Descendente')) {
            $pos = $k;
        }
    }
    if (is_int($pos)) {
        if ($datosConsulta[$pos][3] == 'Ascendente') {
            $orden = ' ASC ';
        } else {
            $orden = ' DESC ';
        }
        $stringOrderBy = 'ORDER BY ' . $datosConsulta[$pos][1] . ' ' . $datosConsulta[$pos][0] . $orden;
    }
    return $stringOrderBy;
}
```

Figura 3.1: Código fuente del método a probar.

Complejidad Ciclomática	
$V(G) = (A - N) + 2$	
$V(G) = (14 - 11) + 2$	
$V(G) = 5$	
Posibles Caminos	
1. 1-2-3-4-5-2-6-7-8-10-11	
2. 1-2-3-4-5-2-6-7-9-10-11	
3. 1-2-3-5-2-6-11	

Figura 3.2: Grafo de Flujo.

Tabla 3.1: Escenario Concatenar ORDER BY.

Para cada camino se realiza un caso de prueba.

Caso de prueba para el camino básico #1:

Descripción: los datos se obtienen a través del método Post conteniendo un arreglo con los datos predefinidos por el usuario para generar la consulta.

```
$juego_datos_1 = array(array('nombre','usuario','Nombre','No Ordenar','Sin Condición','',''),
array('apellido1','usuario','Primer Apellido','Ascendente','Sin Condición','',''),
array('apellido2','usuario','Segundo Apellido','No Ordenar','Sin Condición','',''),
array('usuario','usuario','Usuario','No Ordenar','Sin Condición','',''),
array('correo','usuario','Correo Electronico','No Ordenar','Sin Condición','',''));
```

Figura 3.3: Juego de datos 1.

Condición de ejecución: los valores obtenidos no deben ser nulos.

Obtiene: **\$stringOrderBy= 'ORDER BY usuario.apellido1 ASC';**

Resultados esperados: cadena que contiene la cláusula ORDER BY de la consulta ordenando de manera ascendente por el primer apellido.

Caso de prueba para el camino básico #2:

Descripción: los datos se obtienen a través del método Post conteniendo un arreglo con los datos predefinidos por el usuario para generar la consulta.

```
$juego_datos_2 = array(array('nombre','tusuario','Nombre','Descendente','Sin Condición','',''),
array('apellido1','tusuario','Primer Apellido','No Ordenar','Sin Condición','',''),
array('apellido2','tusuario','Segundo Apellido','No Ordenar','Sin Condición','',''),
array('usuario','tusuario','Usuario','No Ordenar','Sin Condición','',''),
array('correo','tusuario','Correo Electronico','No Ordenar','Sin Condición','',''));
```

Figura 3.4: Juego de datos 2.

Condición de ejecución: los valores obtenidos no deben ser nulos.

Obtiene: `$stringOrderBy= 'ORDER BY tusuario.nombre DESC';`

Resultados esperados: cadena que contiene la cláusula ORDER BY de la consulta ordenando de manera descendente por el nombre.

Caso de prueba para el camino básico #3:

Descripción: los datos se obtienen a través del método Post conteniendo un arreglo con los datos predefinidos por el usuario para generar la consulta.

```
$juego_datos_3 = array(array('nombre','tusuario','Nombre','No Ordenar','Sin Condición','',''),
array('apellido1','tusuario','Primer Apellido','No Ordenar','Sin Condición','',''),
array('apellido2','tusuario','Segundo Apellido','No Ordenar','Sin Condición','',''),
array('usuario','tusuario','Usuario','No Ordenar','Sin Condición','',''),
array('correo','tusuario','Correo Electronico','No Ordenar','Sin Condición','',''));
```

Figura 3.5: Juego de datos 3.

Condición de ejecución: los valores obtenidos no deben ser nulos.

Obtiene: `$stringOrderBy = "";`

Resultados esperados: una cadena vacía, porque no existe criterio para definir un orden.

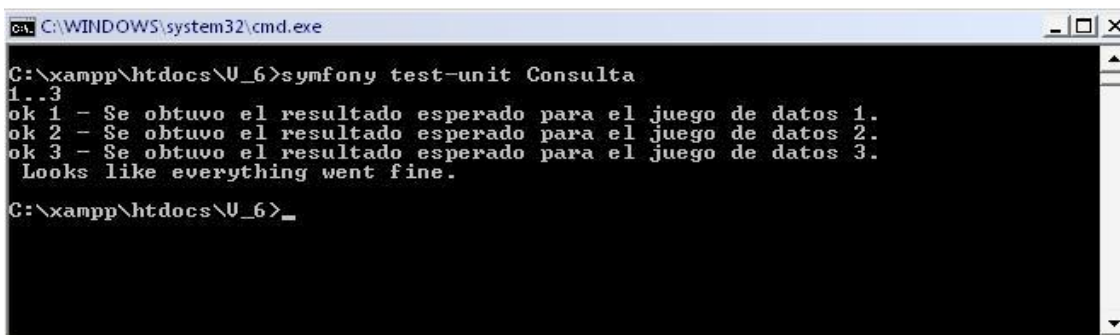


Figura 3.6: Resultado de los casos de prueba 1, 2 y 3 con la herramienta Lime.

Descripción del método calcularTamanoCeldas

Método que calcula el tamaño que van a tener las celdas del PDF a generar con la cantidad de columnas entrada por parámetro, además tiene en cuenta el tamaño de la página.

```

public function calcularTamanoCeldas($numero) {
    $total = 150;
    $tamanoCell[] = 40;
    if ($numero > 1) {
        $temp = $total / ($numero - 1);
        for ($k = 0; $k < $numero - 1; $k++) {
            $tamanoCell[] = $temp;
        }
    }
    return $tamanoCell;
}
    
```

Figura 3.7: Código fuente del método a probar.

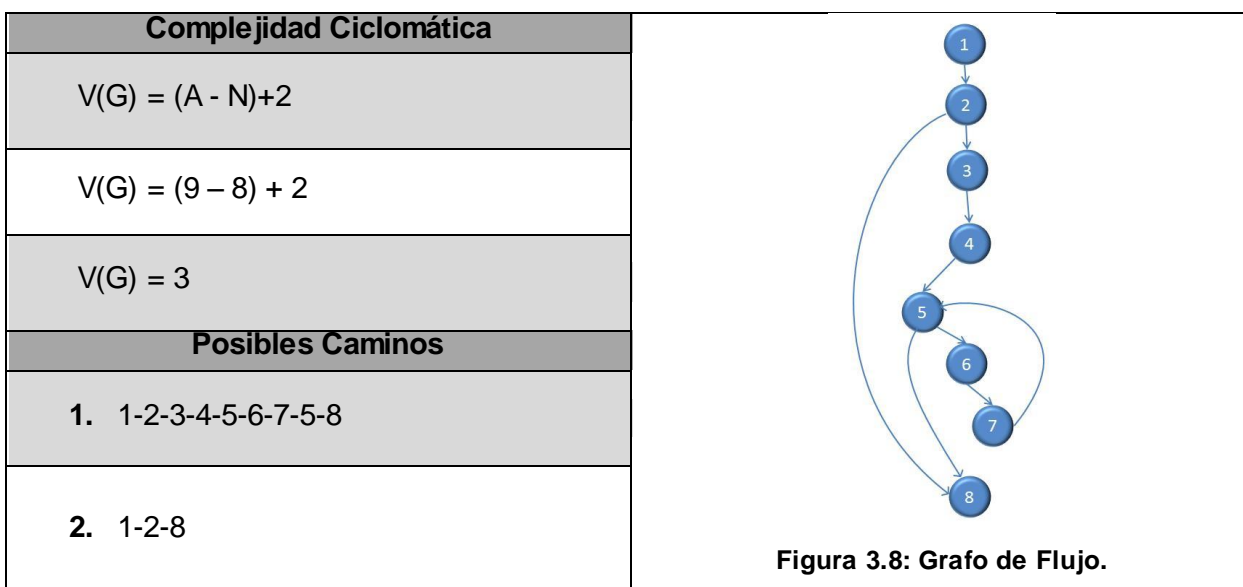


Figura 3.8: Grafo de Flujo.

Tabla 3.2: Escenario calcular tamaño de celdas.

Para cada camino se realiza un caso de prueba.

Caso de prueba para el camino básico #1:

Descripción: los datos se le pasan por parámetro desde un método de la clase actions.class correspondiente al módulo.

\$juego_datos_1 = '1';

Condición de ejecución: los valores obtenidos no deben ser nulos.

Obtiene: \$tamanoCell = [40];

Resultados esperados: arreglo con una sola posición en la cual se encuentra el tamaño de la celda a mostrar en el PDF, con los resultados de la consulta.

Caso de prueba para el camino básico #2:

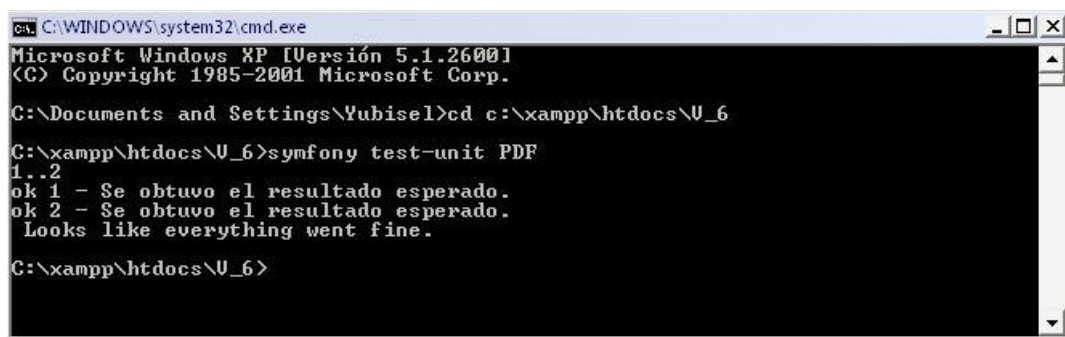
Descripción: los datos se le pasan por parámetro desde un método de la clase actions.class correspondiente al módulo.

\$juego_datos_2 = '5';

Condición de ejecución: los valores obtenidos no deben ser nulos.

Obtiene: \$tamanoCell = [40, 37.5, 37.5, 37.5, 37.5];

Resultados esperados: arreglo que contiene el tamaño de las celdas para mostrar la tabla con los resultados de la consulta en el PDF.



```
ca: C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Yubisel>cd c:\xampp\htdocs\U_6

C:\xampp\htdocs\U_6>symfony test-unit PDF
1..2
ok 1 - Se obtuvo el resultado esperado.
ok 2 - Se obtuvo el resultado esperado.
Looks like everything went fine.

C:\xampp\htdocs\U_6>
```

Figura 3.9: Resultado de los casos de prueba 1 y 2 con la herramienta Lime.

3.6 Diseño de Casos de Prueba de Caja Negra aplicados

Diseño de Casos de prueba correspondiente al Caso de Uso: Crear Consulta

Descripción de la funcionalidad:

El usuario selecciona las tablas a las cuales desea realizarle la consulta, especificando los campos que quiere mostrar al terminar de ejecutar la consulta, luego, en otra interfaz él puede definir los alias con los que se mostraran los datos, así como configurar si quiere que los resultados estén ordenados por un campo específico, además de especificar condiciones para los campos en las consultas.

Condiciones de ejecución:

- Se debe identificar y autenticar ante el sistema, y además de tener los permisos necesarios para ejecutar esta acción.
- Se debe seleccionar el subsistema de Configuración.
- Se debe seleccionar la opción “Generar Consulta”.

Secciones a Probar en el Caso de Uso

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Crear Consulta.	EC 1.1: Adicionar una nueva tabla.	El Caso de Uso comienza cuando el usuario accede a la interfaz “Seleccionar Tablas y Campos” y oprime la opción “Agregar Tabla”.	<ul style="list-style-type: none"> ✓ El sistema muestra la interfaz “Agregar Tabla”, mostrando todas las posibles tablas que se pueden adicionar. ✓ El usuario selecciona la tabla y presiona la opción Agregar. ✓ El sistema verifica que la tabla no esté visible. ✓ Si la tabla seleccionada no está visible, esta es añadida a la interfaz visual.
	EC 1.2: Datos incorrectos	Consiste en detectar que faltan datos por llenar.	<ul style="list-style-type: none"> ✓ El sistema detecta que no se seleccionó una tabla y muestra un mensaje de error.
	EC 1.3: Cancelar	Consiste en cancelarla selección	<ul style="list-style-type: none"> ✓ El usuario selecciona la opción “Cancelar Selección” en la tabla que

	una tabla.	de una tabla añadida al diseño de la consulta.	desea quitar. ✓ El sistema quita la tabla de la interfaz.
	EC 1.4: Cancelar selección.	Consiste en cancelar todas las tablas añadidas al diseño de la consulta.	✓ El usuario selecciona la opción “Cancelar Todo” en el menú de la interfaz. ✓ El sistema quita todas las tablas visibles en la interfaz.
	EC 1.5: Introducir criterios.	Consiste en que el usuario acepta la selección de tablas y campos y pasa a la próxima interfaz a especificar criterios sobre los campos seleccionados.	✓ El usuario selecciona la opción “Agregar Tabla”. ✓ El sistema muestra la interfaz “Agregar Tabla”, mostrando todas las posibles tablas que se pueden adicionar. ✓ El usuario selecciona la tabla y presiona la opción Agregar. ✓ El sistema verifica que la tabla no esté visible. ✓ Si la tabla seleccionada no está visible esta es añadida a la interfaz visual. ✓ El usuario selecciona los campos a tomar como criterios en la consulta. ✓ El usuario selecciona la opción “Introducir Criterios” en el menú de la interfaz. ✓ Si existen campos seleccionados en alguna tabla, el sistema muestra la interfaz “Especificar Criterios de Consulta” para introducir los criterios de la consulta.
	EC 1.6: Datos incorrectos	Consiste en detectar que no existe ninguna tabla seleccionada.	✓ El sistema detecta que no se seleccionó una tabla y muestra un mensaje de error.
	EC 1.7: Datos incorrectos	Consiste en detectar que no existe ningún campo seleccionado en alguna tabla.	✓ El sistema detecta que no se seleccionó algún campo y muestra un mensaje de error.
	EC 1.8: Especificar criterios de consulta.	Consiste en definir los criterios con los que se realizará la consulta (Alias de	✓ El sistema muestra la interfaz que permite especificar los distintos criterios respecto a la selección de campos hecha anteriormente.

		los campos seleccionados, condiciones de ordenamiento o comparación).	
	EC 1.9: Generar Consulta	El Caso de Uso termina cuando el usuario selecciona la opción en el menú "Generar Consulta".	<ul style="list-style-type: none"> ✓ El usuario selecciona la opción "Agregar Tabla". ✓ El sistema muestra la interfaz "Agregar Tabla", mostrando todas las posibles tablas que se pueden adicionar. ✓ El usuario selecciona la tabla y presiona la opción Agregar. ✓ Si la tabla seleccionada no está visible esta es añadida a la interfaz visual. ✓ El usuario selecciona los campos a tomar como criterios en la consulta. ✓ El usuario selecciona la opción "Introducir Criterios" en el menú de la interfaz. ✓ Si existen campos seleccionados en alguna tabla, el sistema muestra la interfaz "Especificar Criterios de Consulta" para introducir los criterios de la consulta. ✓ El usuario introduce los criterios para la consulta. ✓ El usuario selecciona la opción "Generar Consulta" en el menú. ✓ El sistema muestra la interfaz "Resultados de la Consulta" con los datos resultantes de la selección del usuario.

Tabla 3.3: Secciones a probar en el CU Crear Consulta.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
El usuario selecciona la tabla para agregarla.		El sistema muestra la tabla en la interfaz.	El sistema muestra la tabla en la interfaz.

	El usuario no selecciona la tabla para agregarla.	El sistema muestra un mensaje de error.	El sistema muestra el mensaje: "Tiene que seleccionar una tabla."
El usuario selecciona los campos dentro de la tabla para la consulta.		El sistema avanza a la siguiente interfaz.	El sistema muestra próxima interfaz "Especificar Criterios de Consulta".
	El usuario no selecciona ningún campo dentro de las tablas.	El sistema muestra un mensaje de error.	El sistema muestra el mensaje: "Tiene que seleccionar algún campo dentro de una tabla."

Tabla 3.4: Caso de prueba Crear Consulta.

Diseño de Casos de prueba correspondiente al Caso de Uso: Exportar PDF

Descripción de la funcionalidad:

El usuario después de haber construido la consulta, y obtenido los resultados, que el sistema muestre la interfaz "Resultados de la Consulta" y que seleccione la opción en el menú: "Exportar a PDF", el sistema construye el PDF con los datos resultantes de la consulta y se los muestra al usuario en una nueva ventana.

Condiciones de Ejecución:

- Se debe identificar y autenticar ante el sistema, y además de tener los permisos necesarios para ejecutar esta acción.
- Se debe seleccionar el subsistema de Configuración.
- Se debe seleccionar la opción "Generar Consulta".
- Se debe haber hecho una consulta previa para tener datos con los que construir el PDF.

Secciones a Probar en el Caso de Uso

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Exportar PDF.	EC 1.1: Exportar PDF.	El Caso de Uso comienza cuando el usuario termina de hacer la consulta, obtiene los resultados en la interfaz “Resultados de la Consulta” y selecciona la opción “Exportar a PDF”.	<ul style="list-style-type: none"> ✓ El usuario selecciona la opción “Agregar Tabla”. ✓ El sistema muestra la interfaz “Agregar Tabla”, mostrando todas las posibles tablas que se pueden adicionar. ✓ El usuario selecciona la tabla y presiona la opción Agregar. ✓ Si la tabla seleccionada no está visible, esta es añadida a la interfaz visual. ✓ El usuario selecciona los campos a tomar como criterios en la consulta. ✓ El usuario selecciona la opción “Introducir Criterios” en el menú de la interfaz. ✓ Si existen campos seleccionados en alguna tabla, el sistema muestra la interfaz “Especificar Criterios de Consulta” para introducir los criterios de la consulta. ✓ El usuario introduce los criterios para la consulta. ✓ El usuario selecciona la opción “Generar Consulta” en el menú. ✓ El sistema muestra la interfaz “Resultados de la Consulta” con los datos resultantes de la selección del usuario. ✓ El usuario selecciona la opción en el menú “Exportar a PDF”. ✓ El sistema construye el PDF con los resultados de la consulta y se lo muestra al usuario en una nueva ventana.

Tabla 3.5: Secciones a probar en el CU Exportar PDF.

3.7 Aplicación de Métricas

El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y

productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software. (35)

Dentro de los principales atributos de calidad que se incluyen en la aplicación de las métricas, se encuentran:

- **Responsabilidad:** se le asigna a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación:** grado de dificultad en la implementación de un diseño de clases determinado.
- **Reutilización:** nivel de reutilización que tiene una clase o estructura de clase, dentro de un diseño de software determinado.
- **Acoplamiento:** valor de dependencia de una clase o estructura de clase con otras. Este atributo está muy ligado al de Reutilización.
- **Complejidad del mantenimiento:** categoría de esfuerzo para realizar un arreglo, mejora o rectificación de algún error de un diseño de software.

Las clases constituyen la unidad básica y fundamental de un sistema orientado a objetos. Por tanto, y dado el caso de que este software se realizó bajo la POO, es indiscutible que la validación del mismo se centró en la aplicación de métricas dirigidas a sus clases de forma individual, sus jerarquías y colaboraciones, haciendo uso de los atributos de calidad descritos anteriormente. Dichas métricas se encuentran desarrolladas a continuación.

3.7.1 Tamaño Operacional de Clase

Tamaño Operacional de Clase (TOC): está dado por el número de métodos asignados a una clase, y evalúa los siguientes atributos de calidad:

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 3.6: Tamaño Operacional de la Clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Atributo	Categoría	Criterio
Responsabilidad	Baja	\leq Promedio
	Media	Entre Promedio y $2 \times$ Promedio
	Alta	$> 2 \times$ Promedio
Complejidad de implementación	Baja	\leq Promedio
	Media	Entre Promedio y $2 \times$ Promedio
	Alta	$> 2 \times$ Promedio
Reutilización	Baja	$> 2 \times$ Promedio
	Media	Entre Promedio y $2 \times$ Promedio
	Alta	\leq Promedio

Tabla 3.7: Rango de valores para los criterios de evaluación de la métrica TOC.

Al aplicar la métrica TOC se tomó como muestra 6 clases, estas constan de 87 procedimientos, reflejados en la siguiente tabla junto a los atributos de calidad de Responsabilidad, Complejidad de Implementación y Reutilización. Se obtuvo así mismo, un promedio aproximado de 14 procedimientos por clase, y los siguientes datos sirvieron para categorizar los atributos por cada clase:

Promedio de procedimientos por clase= 14.

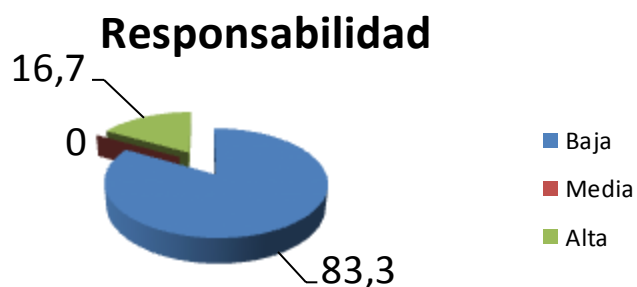
Responsabilidad: Baja ≤ 14 , $14 <$ Media < 28 , Alta > 28 .

Complejidad: Baja ≤ 14 , $14 <$ Media < 28 , Alta > 28 .

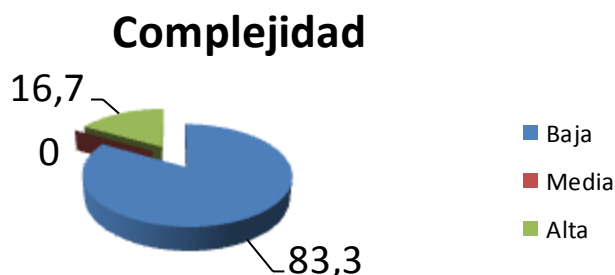
Reutilización: Baja > 28 , $14 <$ Media < 28 , Alta ≤ 14 .

Clases	Funciones	Responsabilidad	Complejidad	Reutilización
Actions.class	4	Baja	Baja	Alta
Grafo	10	Baja	Baja	Alta
Vértice	7	Baja	Baja	Alta
PDF	5	Baja	Baja	Alta
FPDF	47	Alta	Alta	Baja
Crear Consultas	14	Baja	Baja	Alta

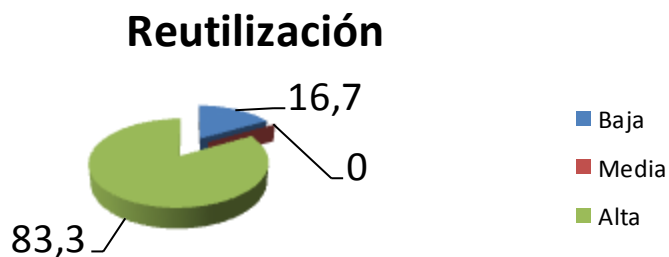
Tabla 3.8: Resultados de la evaluación de la métrica TOC.



Grafica3.1: Por ciento de clases por categorías del atributo Responsabilidad.



Grafica3.2: Por ciento de clases por categorías del atributo Complejidad.



Grafica3.3: Por ciento de clases por categorías del atributo Reutilización.

Tras un análisis de los resultados arrojados por la evaluación del sistema bajo los instrumentos de medición de la métrica TOC, se demuestra que se alcanzaron valores idóneos para cada uno de los atributos de calidad evaluados, puesto que, como se puede observar, el 83.3% de las clases del software contienen un número menor que el promedio de procedimientos para una clase, lo cual influye positivamente en el hecho de que predomine una responsabilidad baja de las clases en un 83.3%, y hace que carezcan de mucha complejidad, por lo que se tornan mucho más reutilizables. Solamente un 16.7 % de las clases tienen baja posibilidades de reutilización. Estos resultados demuestran la solidez del diseño de la implementación de este sistema, y garantizan que este trabaje de forma rápida y eficiente, permitiendo incluso la reutilización de gran parte de las clases y la futura inserción de nuevas funcionalidades dentro de otros módulos en desarrollo.

3.8 Conclusiones del Capítulo

Después de realizarle un conjunto de pruebas al sistema, se concluye que las pruebas constituyen un paso contundente en el ciclo de desarrollo del software que permiten verificar y revelar la calidad de un producto.

Para determinar el nivel de calidad se ha efectuado un conjunto de pruebas para comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema.

El capítulo se centró en la realización de Pruebas de Caja Blanca, de Caja Negra y la aplicación de la métrica del Tamaño Operacional de Clases, para esto fueron definidos diferentes Casos de Prueba por cada Caso de Uso del Sistema, que permitieron detectar la ocurrencia de errores en la aplicación y la posterior solución de los mismos.

Conclusiones

Durante el desarrollo del presente trabajo se llevaron a cabo una serie de fases que permitieron dar cumplimiento al objetivo planteado y que abarcaron todas las tareas investigativas propuestas.

Se hizo un estudio de los antecedentes de algunos sistemas diseñadores de consultas y generadores de reportes en la UCI y el mundo, lo que permitió evaluar las potencialidades de estos y sus debilidades, además de trazar una línea para el desarrollo de la aplicación, entre los sistemas analizados se encuentran el Crystal Report, el Jasper Report y el generador de reportes del Akademos, todos estos presentan ventajas, pero ninguno se ajusta a las necesidades del sistema a implementar, porque se necesita una aplicación que permita generar consulta y reportes de manera dinámica y que sea de fácil entendimiento para personas sin conocimientos de BD. Se valoró la importancia del software a crear, fundamentando la necesidad de su desarrollo. Se caracterizaron las principales tecnologías, metodologías, frameworks, herramientas y lenguajes que son empleados en el desarrollo de la aplicación.

Para implementar las funcionalidades, se estudiaron los contenidos teóricos referentes a las características de las posibles herramientas a usar, lo cual permitió realizar una selección que se ajustara a las especificaciones del sistema a desarrollar.

Se realizó una revisión bibliográfica de los principales algoritmos utilizados en el diseño de consultas de forma dinámica, las tecnologías actuales para el desarrollo de sistemas web, enfatizando en las tecnologías y herramientas libres, de lo que se concluyó utilizar para la implementación del sistema, como lenguaje de programación PHP 5 en su versión 5.2.5, con el framework Symfony versión 1.2.8, sobre el entorno de desarrollo Netbeans 6.8 para PHP y utilizando como SGBD PostgreSQL en su versión 8.3.4. Se implementó una solución cumpliendo con las características de las demás herramientas de edición gráfica existentes.

Se validaron las funcionalidades en el producto para comprobar que cumpliera con los requisitos funcionales, lográndose una aplicación de fácil manejo para personas sin conocimientos de base de datos, con una interfaz visual amigable y sencilla. La ejecución de las tareas en el sistema obtenido ocurre de forma aceptable.

Recomendaciones

Con vistas a dar continuidad al desarrollo de este proyecto se recomienda:

- Validar el funcionamiento de la herramienta a gran escala.
- Presentar este trabajo en futuros eventos científicos.
- Continuar investigando para estar a la par del desarrollo de nuevas tecnologías y herramientas para posibles futuras versiones.
- Realizar un análisis más profundo que permita incorporar nuevas funcionalidades para mejorar la calidad y robustez del producto.

Bibliografía

1. [En línea] [Citado el: 4 de Diciembre de 2010.]
http://chuwiki.chuidiang.org/index.php?title=Ejemplo_b%C3%A1sico_con_Jasper_Report.
2. [En línea] [Citado el: 8 de Diciembre de 2010.]
http://www.elguille.info/colabora/puntoNET/kagueto_crearInforme.htm.
3. MSDN. [En línea] [Citado el: 8 de diciembre de 2010.] <http://msdn.microsoft.com/es-es/library/aa287920%28v=vs.71%29.aspx>.
4. [En línea] [Citado el: 12 de Febrero de 2011.] <http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml>.
5. [En línea] [Citado el: 3 de Febrero de 2011.]
<http://www.monografias.com/trabajos/objetos/objetos.shtml>.
6. [En línea] [Citado el: 10 de Febrero de 2011.] <http://mawape.com.ar/blog/?p=10>.
7. [En línea] [Citado el: 12 de Diciembre de 2010.] <http://www.masadelante.com/faqs/html>.
8. [En línea] [Citado el: 12 de Diciembre de 2010.] <http://www.desarrolloweb.com/articulos/que-es-html.html>.
9. **Flanagan, David.** *JavaScript: The Definitive Guide*. 2002. ISBN.
10. [En línea] [Citado el: 12 de Enero de 2011.] <http://www.maestrosdelweb.com/editorial/flashxml/>.
11. [En línea] [Citado el: 13 de Enero de 2011.] <http://www.masadelante.com/faqs/xml>.
12. [En línea] [Citado el: 11 de Enero de 2011.] <http://www.php-es.com/>.
13. [En línea] [Citado el: 15 de Enero de 2011.] <http://httpd.apache.org/docs/2.0/es/>.
14. [En línea] [Citado el: 15 de Enero de 2011.] <http://www.desarrolloweb.com/articulos/1112.php>.
15. [En línea] [Citado el: 22 de Enero de 2011.] <http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/>.
16. [En línea] [Citado el: 23 de Enero de 2011.]
http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.
17. [En línea] [Citado el: 2 de Febrero de 2011.] <http://www.postgresql.org/about/awards>.
18. [En línea] [Citado el: 2 de Febrero de 2011.]
http://www.computerworld.com.au/article/62894/postgresql_affiliates_org_domain/.
19. [En línea] [Citado el: 20 de Febrero de 2011.] <http://www.elcodigok.com.ar/2010/09/7-caracteristicas-de-netbeans-6-9-1-integrado-a-php/>.
20. [En línea] [Citado el: 16 de Febrero de 2011.]
<http://recursostic.educacion.es/observatorio/web/es/home>.
21. [En línea] [Citado el: 16 de Febrero de 2011.] <http://wamp-server-wamp5.uptodown.com/>.
22. [En línea] [Citado el: 16 de Febrero de 2011.]
<http://recursostic.educacion.es/observatorio/web/es/software/servidores/800-monografico-servidores-wamp>.

23. [En línea] [Citado el: 11 de Febrero de 2011.]
http://descargar.mp3.es/lv/group/view/kl46928/EMS_SQL_Manager_for_PostgreSQL_Lite.htm.
24. [En línea] [Citado el: 11 de Febrero de 2011.] http://www.bajame.net/EMS-SQL-Manager-2005-Lite-for-SQL-Server-2_3_0_4.htm.
25. [En línea] [Citado el: 14 de Febrero de 2011.] <http://www.symfony.es/>.
26. [En línea] [Citado el: 15 de Febrero de 2011.] <http://www.symfony.es/10-razones-para-utilizar-symfony/>.
27. [En línea] [Citado el: 13 de Febrero de 2011.] <http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/>.
28. [En línea] [Citado el: 13 de Febrero de 2011.] http://www.ecured.cu/index.php/Sencha_Ext_JS.
29. [En línea] [Citado el: 16 de Febrero de 2011.] <http://www.webtaller.com/maletin/articulos/que-es-ajax.php>.
30. *Rational Unified Process Best Practices for Software Development Teams*. 1998.
31. **Probasco, Leslee**. *The Ten Essentials of RUP. The Essence of an Effective Development Process*. 2000.
32. **Gamma, Erich , y otros, y otros**. *Design Patterns. Elements of Reusable Object-Oriented Software*.
33. *La Guía definitiva de Symfony 1.2*.
34. www.angelfire.com. [En línea] [Citado el: 10 de Abril de 2011.]
<http://www.angelfire.com/empire2/ivansanes/bywbox.htm..>
35. [En línea] [Citado el: 20 de Abril de 2011.]
<http://www.theserverside.com/tt/articles/content/TestFrameworkComparison/article.html..>
36. **Collin, Peter**. *Publishing Dictionary of Computing, 4th Edition*.
37. **Thomson, Laura y Welling, Luke** . *Programación Desarrollo Web con PHP y MySQL*.
38. FPDF. [En línea] [Citado el: 24 de Febrero de 2011.] <http://www.fpdf.org>.

Anexos

Anexo 1.

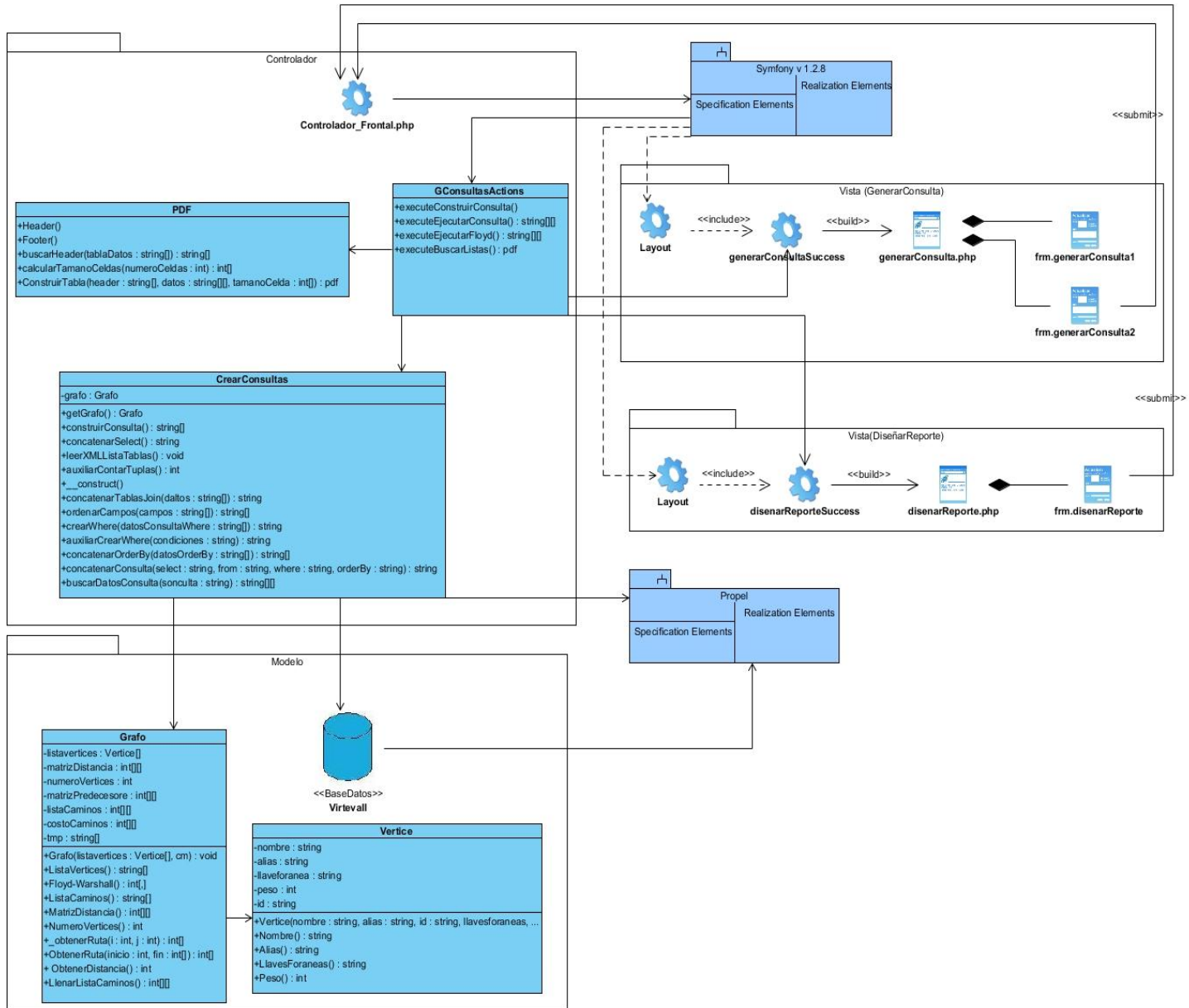


Figura 2.1 DCD Generar Reporte.

Anexo 2.

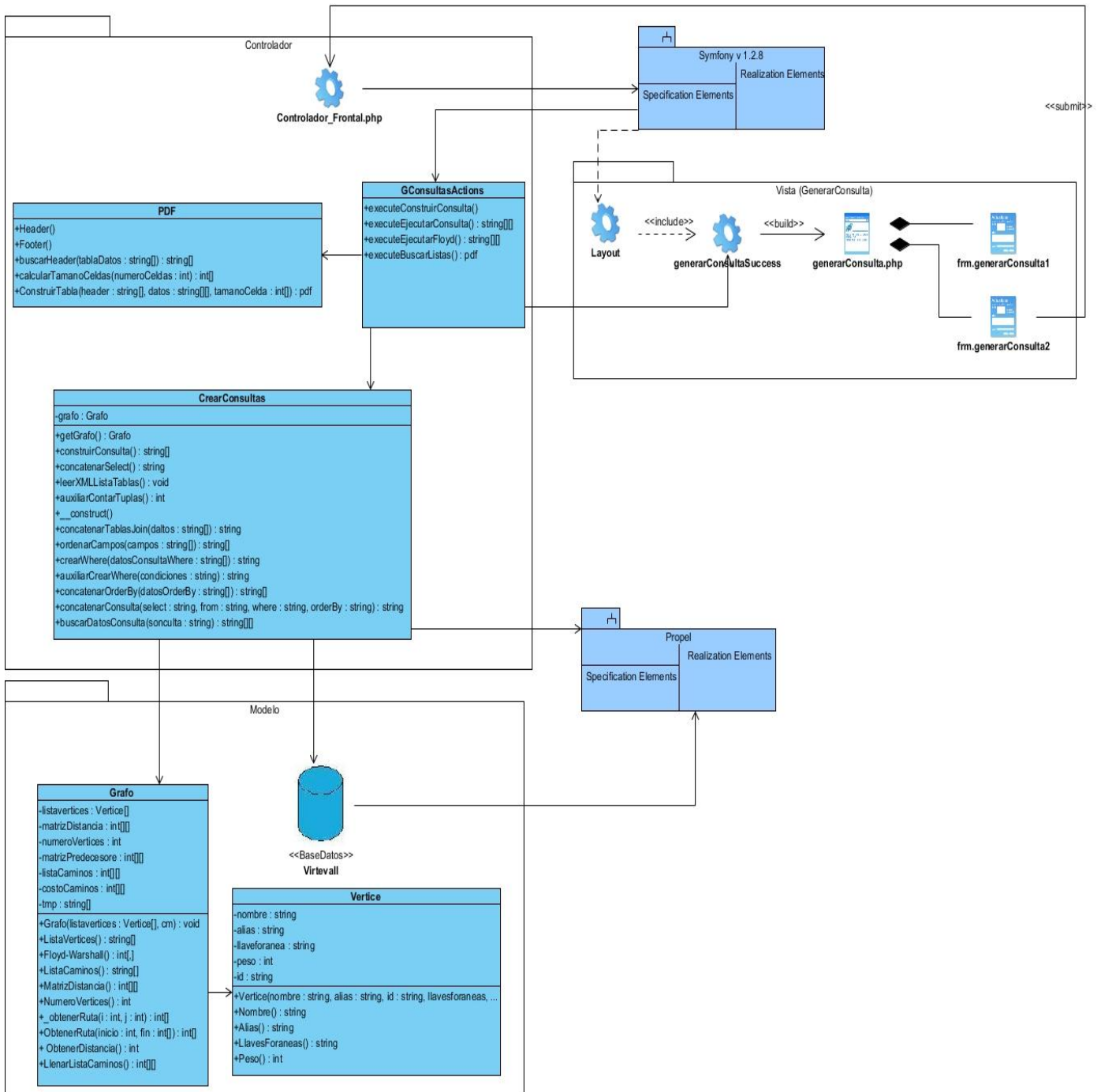


Figura 2.2 DCD Generar Consulta.

Anexo 3.

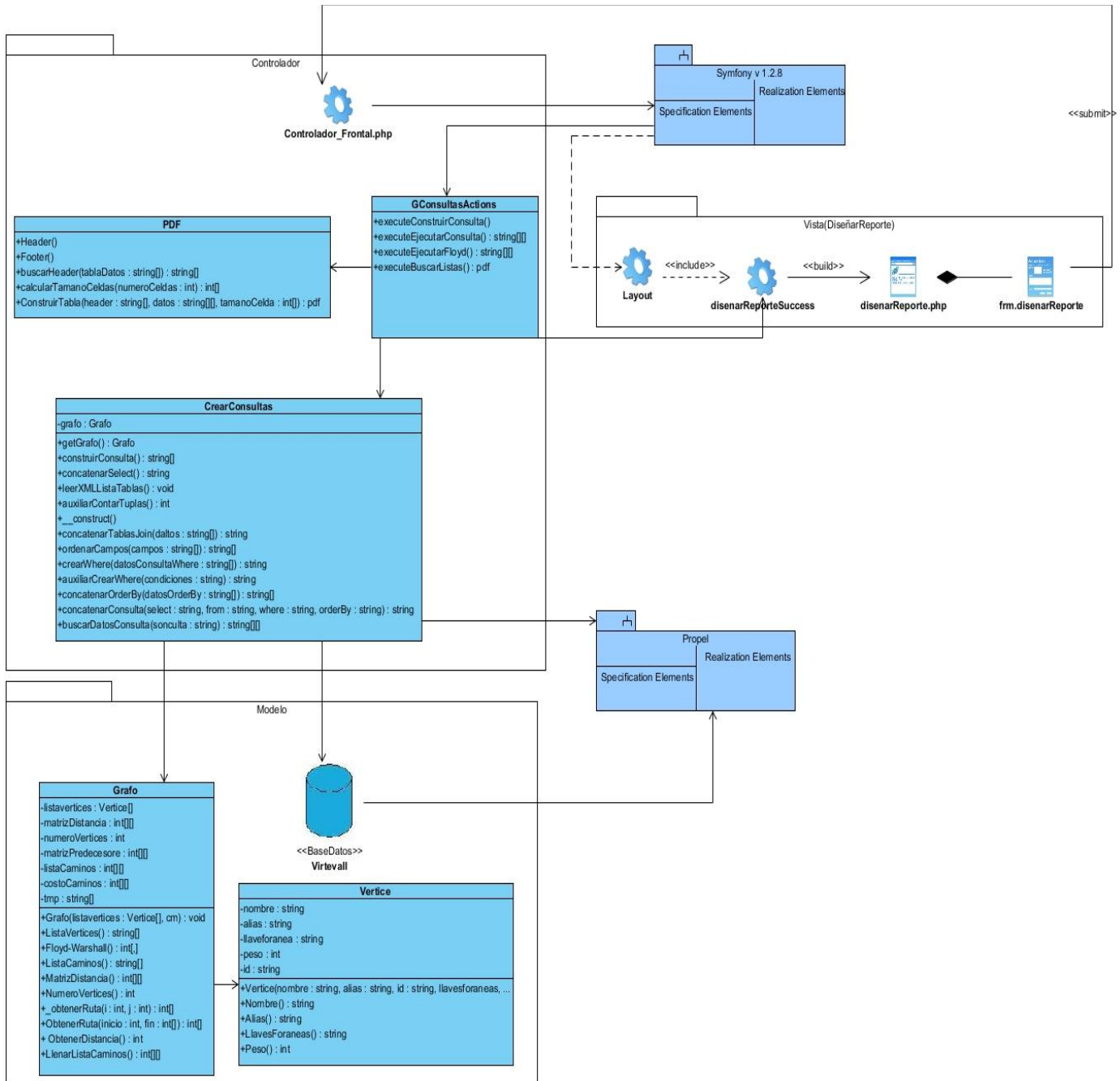


Figura 2.3 DCD Diseñar Reporte.

Glosario de términos

- 1- **STI:** Sistema Tutor Inteligente. Sistema que incorpora técnicas de Inteligencia Artificial a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa.
- 2- **POO:** Programación Orientada a Objetos es el paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora.
- 3- **HTML:** HyperText Markup Language.
- 4- **MVC:** Modelo Vista Controlador. También conocido como: patrón de llamada y retorno; es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.
- 5- **PHP:** Hypertext Preprocessor. Lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.
- 6- **RUP:** Proceso Unificado. Es un proceso de desarrollo de software, el cual es a su vez el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software.
- 7- **IDE:** Entorno de Desarrollo Integrado. Un IDE consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica; a grande rasgos es un entorno de programación empaquetado como un programa de aplicación.
- 8- **Framework:** En términos de desarrollo de software, una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.
- 9- **Pruebas:** La Fase Prueba bajo la metodología de desarrollo de Software RUP es la actividad en la que el sistema o componentes del sistema se ejecuta bajo requerimientos específicos o bajo condiciones, de ahí los resultados son observados y registrados permitiendo la evaluación del sistema o componente probado.
- 10- **JavaScript:** lenguaje de scripts, interpretado, multiplataforma y parcialmente orientado a objetos.
- 11- **API:** Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- 12- **Herramienta:** Aplicación empleada para la construcción de otros programas o aplicaciones.
- 13- **PEAR:** Es un framework y sistema de distribución de compones reutilizables de PHP.
- 14- **Plataforma:** sistema operativo o sistemas complejos, ya sea de hardware o software, sobre el cual un programa pueda ejecutarse. Ejemplos típicos incluyen: arquitectura de hardware, lenguajes de programación y sus librerías de tiempo de ejecución. Ejemplos de plataformas son PC (Windows) y Macintosh (Mac).