Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Componente tabla cruzada para la representación de datos cruzados en GDR v2.0

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

Autor:
Julien Michel Carrillo Smith

Tutores:
MsC. Yadira Robles Aranda

Ing. Juan Miguel Pérez Almaguer

La Habana, Junio 2015
"Año 57 del Triunfo de la Revolución"

Nunca consideres el estudio como una obligación, sino como una oportunidad

para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser 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 _____.

Julien Michel Carrillo Smith

_____

Firma del Autor

MsC. Yadira Robles Aranda

_____

Firma del Tutor

Ing. Juan Miguel Pérez Almaguer

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: MsC.Yadira Robles Aranda.

Formación Académica: Ingeniero en Ciencias Informáticas (2007).

Centro Laboral: Universidad de las Ciencias Informáticas (UCI).

Correo Electrónico: yrobles@uci.cu

Tutor: Ing. Juan Miguel Pérez Almaguer

Formación Académica: Ingeniero en Ciencias Informáticas (2014)

Centro Laboral: Universidad de las Ciencias Informáticas (UCI).

Correo Electrónico: jmalmaguer@uci.cu

AGRADECIMIENTOS

En algún momento Francisco de Quevedo escribió: "el agradecimiento es la parte principal de un hombre de bien", reconozco entonces a todo aquel que ha contribuido con mi formación como persona y como profesional…

A mi madre por sus excesivas preocupaciones y ocupaciones, pero siempre ahí como un ángel guardián.
A Domitila y Teresita por el sustento los fines de semana.
A Belkys por sus consejos.
A mis tutores.
A Leopo y Carol por su apoyo incondicional.
A Eduar por su picardía.
A mi abuelo Mike por sus dulces.
A mis vecinos más cercanos que son parte de mi familia.
A mis pocos amigos por su aliento oportuno.

A todos, de verdad y de corazón, MUCHAS GRACIAS.

DEDICATORIA

A mis padres por la darme la oportunidad de existir.

A mi abuelo que en gloria esté, por tener siempre luz larga.

A mi abuela.

A mi hermano porque comprenda que el sacrificio es la clave del éxito.

RESUMEN

Los generadores de reportes son herramientas complementarias de los sistemas de información que utilizan un lenguaje transparente al usuario, capaces de realizar consultas a la base de datos para obtener información de ella en forma de reportes. El Centro de Tecnologías de Gestión de Datos (DATEC) en la Universidad de las Ciencias Informáticas (UCI), cuenta actualmente con el Generador Dinámico de Reportes v2.0. La herramienta en su primera versión generaba reportes de forma dinámica auxiliándose del motor de reporte *PhpReport*, permitiendo visualizar información mediante tablas cruzadas a través de dos o más variables. En la versión actual se utiliza el motor de reporte *JasperReport* y a pesar de que este cuenta con la capacidad de generar información cruzada a partir de un modelo de datos, la versión del diseñador de reporte para GDR v2.0 no brinda la posibilidad de trabajar con estas opciones limitando su uso en los reportes de salida. Este trabajo se desarrolló con el objetivo de desarrollar un componente tabla cruzada que permita la representación de datos cruzados en el sistema GDR v2.0. Para el desarrollo del mismo se utilizó un conjunto de herramientas y tecnologías que permitieron diseñar e implementar las clases del componente siguiendo las pautas que propone la metodología OpenUP para construir un software con buenas prácticas. Como resultado se logró la integración del componente tabla cruzada al Generador Dinámico de Reportes v2.0 para generar reportes que contengan datos cruzados.

Palabras claves: datos cruzados, componente, tabla cruzada, reportes.

ABSTRACT

The report generators are complementary tools for information systems using transparent to the user, able to query the database for information it reports as language. The Technology Centre Data Management (DATEC) at the University of Information Sciences (UCI), currently has Generator v2.0 Dynamic Reports. The tool in its first version dynamically generated reports leaning PhpReport report engine, allowing display information by cross through two or more variables tables. In the current version JasperReport report engine and although this has the ability to generate cross information from a data model, version of the report designer for GDR v2.0 does not provide the possibility to work with is used these options limiting its use in the output reports. This work was developed with the aim of developing a cross-table component that allows cross-representation in the GDR v2.0 system data. For its development a set of tools and technologies that enabled design and implement component classes along the lines proposed by the OpenUP methodology to build software with best practices used. As a result the cross-component integration table was achieved v2.0 Dynamic Report Generator to generate reports that contain data cross.

Key words: cross data, component, crosstab, reports.

ÍNDICE

Introducción 1
Capítulo I: Fundamentación Teórica 8
Introducción 8
1.1. Tabla 8
1.2. Tabla cruzada 8
1.3. Componente 9
1.4. Componente tabla cruzada 10
1.5. Reporte 10
1.6. Sistemas generadores de reportes 11
1.7. Ambiente de desarrollo 16
1.7.1. Metodología de desarrollo de software 16
1.7.2. Lenguaje Unificado de Modelado 18
1.7.3. Herramienta CASE 18
1.7.4. Lenguajes de Programación 19
1.7.5. Marcos de trabajo 21
1.7.6. Entorno de Desarrollo 22
1.7.7. Servidor de Aplicaciones 23
1.7.8. Sistema Gestor de Base de Datos 24
Conclusiones Parciales 25
Capítulo 2: Análisis y diseño 26
Introducción 26
2.1. Modelo de Dominio 26
2.2. Requisitos del sistema 28
2.2.1. Requisitos Funcionales 28
2.2.2. Requisitos No Funcionales 30
2.3. Modelo de casos de uso del sistema 32
2.4. Patrones arquitectónicos 38
2.5. Patrones utilizados en la solución 39

ÍNDICE

2.5.1. Patrones de diseño GRASP 39
2.6. Modelo del Diseño 40
2.6.1. Diagramas de clases del diseño 41
2.7. Diagrama de Despliegue 43
Conclusiones Parciales 44
Capítulo 3: Implementación y prueba 45
Introducción 45
3.1. Implementación 45
3.1.1. Diagrama de Componentes 45
3.2. Código fuente 47
3.2.1. Estándares de codificación 47
3.3. Pruebas del software 48
3.3.1. Estrategia de Prueba 49
3.4. Diseño de Caso de Prueba 50
3.5. Resultado de pruebas 55
Conclusiones parciales 57
Conclusiones generales 58
Recomendaciones 59
Referencias bibliográficas 60
Bibliografías 62
Anexos 65
Glosario de términos 66

Índice de Tablas

Tabla 1. Comparación entre SGDR. 15
Tabla 2. Descripción del Caso de Uso Administrar componente tabla cruzada 34
Tabla 3. Sección de prueba para el CUS_Administrar componente tabla cruzada. 51
Tabla 4. Descripción de las variables 52
Tabla 5. Descripción de las variables del escenario Seleccionar grupo de filas. 55

Índice de Figuras

Figura 1. Adicionar componente tabla cruzada en el iReport. 13
Figura 2. Componente tabla cruzada generado. 14
Figura 3. Modelo de dominio. 27
Figura 4. Diagrama de Casos de Uso del Sistema. 34
Figura 5. Diagrama de clases del diseño CUS Administrar componente tabla cruzada. 42
Figura 6. Diagrama de despliegue. 43
Figura 7. Diagrama de componentes del CUS Administrar componente tabla cruzada. 46
Figura 8. Resumen de las no conformidades. 56
Figura 9. Componente tabla cruzada integrado al sistema GDR v2.0. 65

INTRODUCCIÓN

La información está destinada a resolver determinados problemas, sirviendo para el desarrollo individual y corporativo, estando presente en todos los niveles de actividad y ramas de la economía, la política y la sociedad.

En la actualidad existen empresas que tratan una gran cantidad de información, la cual, si no es almacenada de manera cuidadosa, puede perderse o deteriorarse en momentos que sea necesaria su utilidad para un óptimo desempeño de la empresa. En la mayoría de los casos estas organizaciones requieren de una alternativa que les proporcione información útil en tiempo real para ayudar a la toma de decisiones, así como para mejorar los procesos, productos y servicios que se llevan a cabo en las mismas; por esta razón es recomendable representar los datos mediante tablas cruzadas en los reportes para mostrar los análisis y resultados de las empresas; por lo que se consideran un importante requisito dentro de la lógica del negocio.

Existen herramientas que permiten obtener dichos informes, ejemplo de ello son los generadores de reportes, que permiten a los usuarios acceder de forma sencilla y rápida a los datos guardados en forma de reportes.

En Cuba, muchas empresas y proyectos realizan sus análisis a través de generadores de reportes, un ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI), la cual, cuenta con una amplia infraestructura productiva, donde dispone de varios centros de desarrollo, entre los que se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC). Su objetivo principal es proveer soluciones integrales, brindar soporte y asesorías relacionado con tecnologías de bases de datos y análisis de información, auxiliándose de sistemas como el Generador Dinámico de Reportes (GDR).

El GDR es uno de los sistemas desarrollados en dicho centro, tiene entre sus propósito garantizar la generación dinámica de reportes partiendo de datos persistentes en algún origen de datos soportado por el sistema (*PostgreSQL*, *MySQL*, *Sqlite*, *Microsoft SQL Server*). El mismo está compuesto por un conjunto de módulos que brindan funcionalidades para el trabajo con los reportes brindándoles soporte durante todo el ciclo de vida. En la primera versión del sistema se utilizó como motor de reportes *PhpRep*ort, el cual, soportaba el trabajo con tablas, permitiendo visualizar información de forma lineal o cruzada por dos o más

variables. En la versión actual se utiliza el motor de reportes Jasper Report; a pesar de que este cuenta con la capacidad de crear documentos, preparados para imprimir información cruzada en una forma simple y flexible, la versión del diseñador de reporte para GDR v2.0 en desarrollo no brinda la posibilidad de trabajar con estas opciones limitando su uso en los reportes de salida.

A raíz de la problemática planteada, surge el **Problema de la investigación:**

¿Cómo construir reportes que permitan visualizar datos cruzados mediante tablas en el Generador Dinámico de Reportes v2.0?

Este problema se define con el **Objeto de estudio**: Desarrollo de componentes en los sistemas generadores de reportes, el cual, está delimitado por el **Campo de acción:** Componente tabla cruzada para la representación de datos cruzados en el sistema Generador Dinámico de Reportes.

Para resolver el problema planteado surge como **objetivo general:** Desarrollar un componente tabla cruzada que permita la representación de datos cruzados en el sistema GDR v2.0.

Para guiar la investigación se definen las siguientes **preguntas de la investigación**:

- ¿Cuáles son los fundamentos teóricos que soportan el proceso de desarrollo del componente tabla cruzada en el Sistema Generador Dinámico de Reportes en la versión 2.0 para visualizar datos cruzados?
- ¿Cuáles características debe poseer el componente tabla cruzada para que los reportes que se generen contengan tablas con datos cruzados en el Generador Dinámico de Reportes en su versión 2.0?
- ¿Cómo estructurar el proceso de desarrollo para que el sistema Generador Dinámico de Reportes permita la visualización de los datos cruzados mediante tablas en su versión 2.0?
- ¿Cómo comprobar el funcionamiento correcto del componente tabla cruzada en el Generador Dinámico de reportes en su versión 2.0?

INTRODUCCIÓN

Tareas de la investigación:

- Análisis de los conceptos principales relacionados con los sistemas generadores dinámicos de reportes y el componente tabla cruzada para la confección del marco teórico que guiará la investigación.
- Definición de las herramientas y metodologías a utilizar en el desarrollo del soporte del componte tabla cruzada para la representación de datos cruzados en el Generador Dinámico de Reportes v2.0.
- Definición de los requisitos funcionales y no funcionales para el correcto funcionamiento del componente tabla cruzada del Generador Dinámico de Reportes v2.0.
- Definición de los principios arquitectónicos del componente tabla cruzada del Generador Dinámico de Reportes v2.0 para definir su estructura y composición.
- Definición de los principios de diseño del componente tabla cruzada del Generador Dinámico de Reportes v 2.0 para orientar el proceso de implementación.
- Implementación del componente tabla cruzada para cumplir con los requisitos funcionales y no funcionales en el Generador Dinámico de Reportes v2.0.
- Realización de pruebas de software para validar el correcto funcionamiento del componente tabla cruzada del Generador Dinámico de Reportes v2.0.

Métodos científicos:

Para dar cumplimiento a la idea planteada, y en correspondencia con el objetivo y las tareas investigativas propuestas, se aplican los siguientes métodos científicos durante el desarrollo de la investigación.

Teórico:

Analítico -Sintético : Se emplea para analizar las fuentes de información consultadas para la elaboración del marco teórico de la investigación vinculado con el desarrollo del componente tablacruzada que permita

la representación de datos cruzados en los reportes de salida generados por el Generador Dinámico de Reportes v2.0.

Modelación: Se emplea para representar a través de los modelos y diagramas parte del desarrollo del componente tabla cruzada para la representación de datos cruzados en el Generador Dinámico de Reportes v2.0.

Empíricos:

Análisis Estático: se emplea para la revisión del código fuente del componente tablacruzada del iReport 5.1.0, con el propósito de entender la estructura y funcionamiento de los elementos por los cuales está conformado.

Técnicas de recopilación de información

Tormenta de ideas: Se emplea con el objetivo de identificar las características y funcionalidades que debe poseer el componente tablacruzada del Generador Dinámico de Reportes v2.0, se tuvo en cuenta los criterios de los tutores de la investigación y los desarrolladores de las versiones anteriores al GDR v2.0.

El presente documento se estructura en tres capítulos que abarcan todo lo relacionado con el trabajo investigativo, la modelación y la implementación del sistema.

CAPÍTULO 1. Fundamentación teórica

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

CAPÍTULO 2. Análisis y diseño

En el capítulo se describe el proceso llevado a cabo durante la fase de análisis y diseño del componente tabla cruzada. Se muestra la estructura del componente a desarrollar a través de una representación visual de las clases conceptuales en el modelo de dominio. Se definen los requisitos funcionales y no funcionales

del software, lo cual, es un paso de gran importancia para su éxito. Se especifica la estructura física de la solución que se propone mediante el diagrama de casos de uso del sistema y se describe la arquitectura propuesta y los artefactos generados de acuerdo a la metodología utilizada, permitiendo asegurar una organización del componente para alcanzar resultados satisfactorios.

CAPÍTULO 3: Implementación y prueba

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

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se abordarán temas relacionados con los sistemas generadores de reportes, haciendo énfasis en la utilización del componente tabla cruzada en los mismos. Se profundizará en sus características y se realizará un análisis crítico a través de una comparación entre las mismas, teniendo en cuenta los aspectos más relevantes. Se selecciona la metodología, los distintos lenguajes, herramientas y tecnologías a utilizar para el desarrollo de la solución propuesta.

1.1. Tabla

Una tabla es una herramienta de organización de información que es incluida en bases de datos u hojas de cálculo, pero también pueden incorporarse a documentos de texto y otros programas.

Está compuesta por filas horizontales y columnas verticales. El campo es el nombre de cada columna, debe ser único y con un tipo de dato asociado. El registro, por otro lado, es cada fila que compone la tabla y que incluye datos (o también puede ser nulo).

Las tablas son estructuras útiles y a menudo fáciles de interpretar para relacionar datos e información de manera pertinente. Son aplicables a la hora de organizar actividades o cronogramas, pero también sirve para llevar cuentas y cálculos financieros. Una tabla puede ser empleada para algo tan sencillo como organizar la información familiar o cuestiones complejas como llevar la contabilidad de una empresa.(Definición ABC, 2008)

1.2. Tabla cruzada

Es un tipo de tabla que pueden relacionarse con tantas como se desee, brindando un nivel de flexibilidad y de acceso a los datos muy elevado. En principio pueden parecer un sistema cerrado y precario de organizar la información; pero cuando se combinan con conceptos tales como las referencias (muy utilizado en bases de datos de gran envergadura), el nivel de complejidad crece de forma exponencial.(Definicion.De, 2008)

El procedimiento de Tabla Cruzada está diseñado para resumir dos columnas de datos. Esta construye una tabla de dos-caminos mostrando las frecuencias de ocurrencia de cada uno de los pares formados de las dos columnas. Estadísticas son construidas para cuantificar el grado de asociación entre las columnas, y pruebas son corridas para determinar si hay dependencia estadísticamente significante entre las columnas o no.(Inc, 2005)

Las tablas cruzadas, permiten describir el comportamiento de los datos de dos o más variables de acuerdo a la combinación o cruce de sus categorías. Para realizar una tabla de contingencia por medio del generador de tablas, se debe ingresar una variable categórica a cada una de las dimensiones de la tabla (fila y columna).(Martínez, 2011)

Posteriormente al análisis de las definiciones anteriores, se define que las tablas cruzadas son una herramienta que se utiliza para comprender una determinada información, representando la mismas mediante una serie de filas y columnas, posibilitando registrar y analizar la relación entre dos o más variables, habitualmente de naturaleza cualitativa.

1.3. Componente

Bloque de construcción modular para software de computadora que puede ser utilizado desde 3 puntos de vista:

Orientado a Objetos: un componente contiene un conjunto de clases que colaboran entre sí. El diseño de un componente implica añadir a la definición de clases en el análisis (dominio del problema) información para su implementación en software.(Pressman, 2010)

Convencional: un componente es un elemento funcional de un programa que incluye lógica de procesamiento, estructuras de datos internas requeridas para implementar dicha lógica y una interfaz que permite que el componente sea invocado y que se le puedan pasar datos.(Pressman, 2010)

Relacionado a procesos: Reutiliza software. Cuando se desarrolla la arquitectura, se escogen componentes o patrones de diseño de un catálogo, los cuales, fueron creados para ser reutilizados. (Pressman, 2010)

Otros conceptos plantean que un componente es una parte modular, entregable y reemplazable de un sistema que encapsula su implementación y expone un conjunto de interfaces. (OMG Unified Modeling Language Specification, 2010)

Luego de analizar los conceptos anteriores se define que un componente es un elemento funcional de un programa que contiene un conjunto de interfaces que permiten que el mismo sea invocado para realizar una determinada actividad de acuerdo a los datos que se le transfieren.

1.4. Componente tabla cruzada

Es un informe en el cual se representa una tabla sencilla en la que las filas, columnas y valores son recuperados dinámicamente a partir de un determinado *datasource* o conexión a base de datos.(Blanco Criado, 2009)

Permiten crear diferentes vistas de los datos de acuerdo a las variables que se desean integrar. Ofreciendo una gran cantidad de posibilidades de comparación de manera que se puede hacer un análisis muy exhaustivo de la información sin necesidad de estar creando reportes individuales.(EXCELTOTAL, 2010)

Luego de analizar los conceptos anteriores se define que el componente tabla cruzada hace referencia al modelado o recopilación de datos cruzados[1] en un reporte por parte de una aplicación de un programa que permite operar con los mismos organizándolos y poniéndolos en relación de diversas maneras, teniendo en cuenta el origen de datos y la función estadística seleccionada.

1.5. Reporte

Un reporte es un informe o una noticia que puede ser impreso, digital o audiovisual, pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos, persuasivos y

[1] «datos cruzados: Es la información complementaria que permite establecer una determinada idea con otra, donde hace alusión a un elemento que se ubica en un ambiente diferente del actual y se utiliza para referirse a la dependencia o semejanza que pudiera existir entre diferentes hechos. »

específicamente en el ámbito de la informática los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y fácil de interpretar por los usuarios. Los reportes tienen diversos niveles de complejidad, desde una lista o enumeración hasta gráficos mucho más desarrollados. Según la herramienta informática y la base de datos en cuestión, los reportes permiten la creación de etiquetas y la elaboración de facturas. (Brito Rodriguez, 2012)

Posteriormente al análisis del concepto anterior se tiene como conclusión que los reportes son de gran importancia pues permiten al usuario realizar de forma transparente, consultas a las bases de datos y obtener información de ellas en un determinado formato, con una mayor rapidez y un mayor nivel de detalle y flexibilidad.

1.6. Sistemas generadores de reportes

Los generadores de reportes son herramientas complementarias de los sistemas de información incluidos en la mayoría de los productos de software empresariales. Proveen una forma transparente al usuario para realizar consultas a las bases de datos y obtener información de ella en forma de reporte. Además, tienen la capacidad de interactuar con los resultados obtenidos basándose en los datos para tomar sus propias decisiones. (Orihuela Sánchez, 2013)

iReport

Es un generador de informes visuales y un diseñador para *JasperReports*[2]. Herramienta que posibilita entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV y XML. Está escrito en el lenguaje de desarrollo Java.

Esta aplicación de escritorio permite que los usuarios corrijan visualmente informes complejos y subinformes. iReport está además integrado con JFreeChart, marco de software OpenSource para el

[2] « JasperReport: Es una librería que funciona como un motor de reportes implementada completamente en Java brindando el máximo nivel de portabilidad, con un amplio y expandible grupo de posibles fuentes de datos.».

lenguaje de programación Java, el cual permite la creación de gráficos complejos de forma simple. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, JavaBeans, XML. (Orihuela Sánchez, 2013)

Características

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

Sistema Operativo: Windows y Linux.

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

Componente tabla cruzada en Ireport

Los datos a visualizar en el iReport pueden ser escogidos por diferentes vías como la creación de una consulta SQL para el reporte, donde luego de realizar dicha operación, se debe pulsar sobre el icono del componente tabla cruzada correspondiente, y arrastrar el mismo hacia el área del reporte donde se desea colocar. (Ver Figura 1)

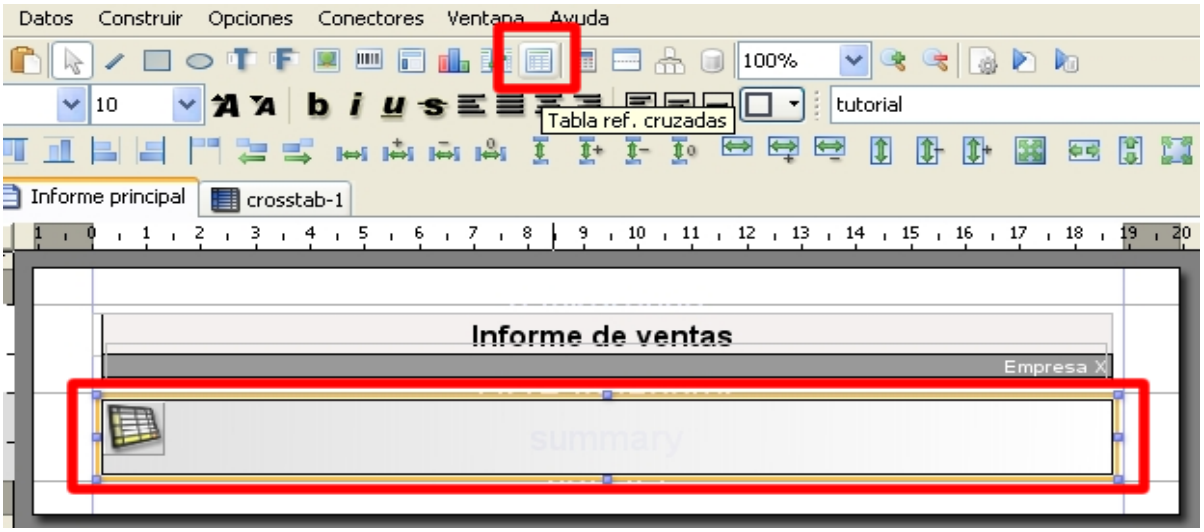
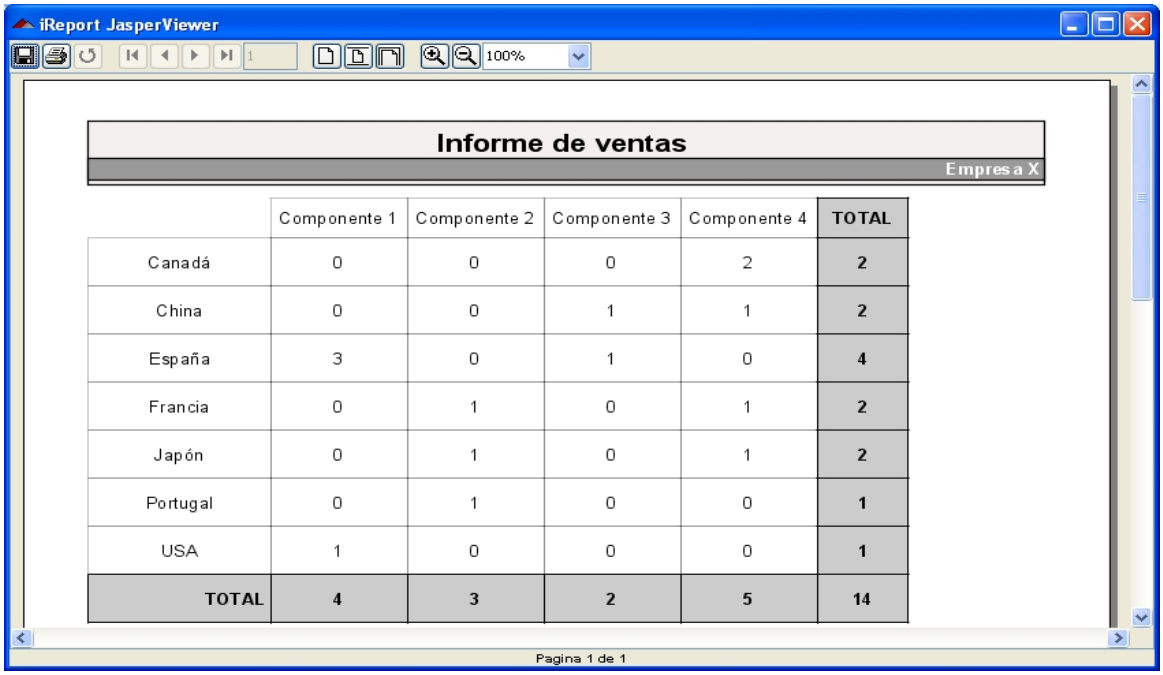


Figura 1. Adicionar componente tabla cruzada en el iReport.

Luego aparecerá una ventana donde se deberá configurar el componente tabla cruzada o *crosstab*; para ello se definen una serie de pasos:

- Elegir un modelo de datos (subdataset) o crear uno nuevo
- Definir una agrupación (*group by*) para las filas de la tabla.
- Definir una agrupación (*group by*) para las columnas de la tabla.
- Establecer los datos que se visualizarán en el interior de la tabla mediante una función estadística.
- Establecer filas/columnas exteriores con la suma total/parcial.

Una vez concluida esta configuración, se visualiza el componente tabla cruzada con los datos requeridos.



	Componente 1	Componente 2	Componente 3	Componente 4	TOTAL
Canadá	0	0	0	2	2
China	0	0	1	1	2
España	3	0	1	0	4
Francia	0	1	0	1	2
Japón	0	1	0	1	2
Portugal	0	1	0	0	1
USA	1	0	0	0	1
TOTAL	4	3	2	5	14

Figura 2. Componente tabla cruzada generado.

Generador Dinámico de Reportes

Generador Dinámico de Reportes (GDR) es una aplicación web que tiene como objetivo generar reportes de forma rápida, interactiva y con una amplia gama de alternativas para los usuarios. Permite a sus clientes consultar las bases de datos de sus organizaciones y poder generar reportes con la información que estos manejan, independientemente del sistema gestor de base de datos que utilicen ya sea MySQL, Sqlite o PostgreSQL. El desarrollo de dicho sistema está basado en tecnología web, lo cual, hace del sistema una gran ventaja para las organizaciones, puesto que lo mantiene disponible desde cualquier estación de trabajo donde se encuentra desplegado el GDR v2.0, las 24horas del día. Tiene una interfaz gráfica amigable y está orientado al usuario final. Se encuentra en constante perfeccionamiento con el objetivo de que cada día, logre satisfacer las expectativas del mercado. Durante su implementación se emplearon tecnologías novedosas en el desarrollo web actual como: PHP, JavaScript, Symfony, Doctrine, Ext-JS, OpenJacob, XML, PostgreSQL. Es una aplicación multiplataforma y soporta imágenes, gráficas y varios orígenes de datos. (Rodriguez Durán, 2011)

Ventajas

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

Sistema Operativo: El GDR v2.0 se encuentra disponible para todo tipo de sistema operativo.

Licencia: Está bajo licencia GPL (Licencia Pública General).

Comparación entre ambas herramientas de generación de reportes

Tabla 1. *Comparación entre SGDR.*

Sistemas generadores de reportes	iReport	GDR v2.0
Multiplataforma	Sí	Sí
Sistema privativo	No	No
Motor de reporte	JasperReport	JasperReport
Aplicación	Escritorio	Web
Componente tabla	Sí	No

Después de analizar estos sistemas de generación de reportes, se puede apreciar que poseen diversos aspectos que determinan sus potencialidades; donde ambos poseen un motor de reporte (*JasperReport*) que permite basado en sus características visualizar datos cruzados, función que se puede observar en el sistema iReport a través de su componente tabla cruzada, no siendo así para el GDR v2.0.

Basándose en lo anteriormente dicho, se determina que el componente tabla cruzada debe incorporarse dentro del sistema Generador Dinámico de Reportes en su versión 2.0 para entender y comprender la realidad y el entorno de una determinada organización; utilizando la información que se manipula con simplicidad y diseño a través de tablas para facilitar la toma decisiones.

1.7. Ambiente de desarrollo

Las herramientas y tecnologías son programas o aplicaciones que se crean y se diseñan para efectuar una o varias funciones determinadas. Con el objetivo de definir ventajas, oportunidades y con la disposición de potenciar al máximo las prestaciones de las mismas se realiza el siguiente estudio.

1.7.1. Metodología de desarrollo de software

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

La finalidad de una metodología de desarrollo es garantizar la eficacia, cumpliendo los requisitos iniciales y la eficiencia, minimizando las pérdidas de tiempo en el proceso de generación de software.(Orihuela Sánchez, 2013)

OpenUP

Es una metodología ágil dirigida a la gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.(Bernal Rojas, 2013)

El objetivo de OpenUP es ayudar al equipo de desarrollo a través de todo el ciclo de vida de las iteraciones, de modo que este sea capaz de añadir valor de negocio para los clientes de una forma predecible: con la entrega de un software operativo y funcional al final de cada iteración. El ciclo de vida del proyecto provee

a los clientes de una visión del proyecto, transparencia y les dota de los medios para que les permitan controlar la financiación, el riesgo, el ámbito y el valor de retorno esperado.(Bernal Rojas, 2013)

Mantiene las características esenciales de RUP, en el cual se incluyen las siguientes características:

- Desarrollo incremental.
- Uso de casos de uso y escenarios.
- Manejo de riesgos.
- Diseño basado en la arquitectura.

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

1. **Concepción**: primera fase en el proyecto del ciclo de vida, acerca del entendimiento del propósito y objetivos obteniendo suficiente información para confirmar que el proyecto debe hacer. El objetivo de ésta fase es capturar las necesidades de los *stakeholder* (clientes) en los objetivos del ciclo de vida para el proyecto.(Bernal Rojas, 2013)

2. **Elaboración**: se trata los riesgos significativos para la arquitectura. El propósito de esta fase es establecer la base la elaboración de la arquitectura del sistema.(Bernal Rojas, 2013)

3. **Construcción**: esta fase está enfocada al diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito de esta fase es completar el desarrollo del sistema basado en la arquitectura definida.(Bernal Rojas, 2013)

4. **Transición**: es la última fase, cuyo propósito es asegurar que el sistema es entregado a los usuarios, evalúa la funcionalidad y rendimiento del último entregable de la fase de construcción.(Bernal Rojas, 2013)

Partiendo de que es un proyecto de baja escala, el tiempo de desarrollo es corto, el equipo de trabajo y la asignación de responsabilidades es limitado, no existe un ambiente de alto riesgo técnico y es necesario la documentación escrita para el estudio de la situación problemática, se decide utilizar la metodología OpenUp para guiar el proceso de desarrollo del componente tabla cruzada, ya que permite dirigir cualquier tipo de

proyecto, cuenta con pocos artefactos y posibilita la detección y corrección anticipada de errores en el ciclo iterativo de vida de un sistema, evitando la elaboración de documentos, diagramas e iteraciones innecesarias, basándose en el prefijando de las mismas por parte del equipo de desarrollo con el objetivo de incrementar las funcionalidades del producto para optimizar su salida .

1.7.2. Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado o UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos, información que se utiliza o produce mediante un proceso de software. Este lenguaje de modelado no es una guía para realizar el análisis y diseño orientado a objeto, es decir, no es un proceso, es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos informáticos, de arquitectura o de cualquier otra rama. Además ayuda a una rápida construcción de aplicaciones con mejor calidad y promueve la reutilización.(Larman, 2006)

1.7.3. Herramienta CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadoras), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costos de las mismas en términos de tiempo y de dinero. Estas herramientas son de mucha ayuda en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores.(Orihuela Sánchez, 2013)

Visual Paradigm 8.0

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

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

1.7.4. Lenguajes de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; permitiendo al desarrollador comunicarse con los dispositivos de hardware y software existentes. Los programas creados pueden ser usados para controlar el comportamiento físico y lógico de una máquina.(Orihuela Sánchez, 2013)

PHP

PHP 5.3 (Procesador de Hipertexto) es un lenguaje interpretado de propósito general ampliamente utilizado en el mundo, puede ser incrustado dentro de código HTML (Lenguaje de Marcado de Hipertexto). Este lenguaje está diseñado especialmente para incrementar el dinamismo de las páginas web, aprovechando los recursos de las redes informáticas y los escasos requerimientos de hardware que solicita el lenguaje para que sus aplicaciones funcionen correctamente.(Orihuela Sánchez, 2013)

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

Para el desarrollo de la solución se seleccionó la versión 5.3 del lenguaje de programación PHP porque se adapta a las necesidades del usuario siendo un lenguaje rápido, robusto, seguro, sencillo en la sintaxis y de fácil acceso a las bases de datos, posibilitando trabajar en conjunto con otros componentes, disponiendo de una amplia gama de paquetes totalmente autoinstalables que optimizan varios procesos en el desarrollo de software.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

JavaScript

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

Se emplea JavaScript para la creación de interfaces de usuarios más complejas e interactivas haciendo uso de las bibliotecas de ExtJS. Puede combinarse con otras herramientas de desarrollo web como Symfony 2.0.

Lenguaje para la transferencia de información JSON

JSON, (Notación de Objetos de JavaScript), genera un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso del lenguaje de marcas extensible (XML). Es muy sencillo de usar, especialmente como alternativa a XML en AJAX. Una de sus ventajas sobre XML como formato de intercambio de datos es que prevalece un formato mucho más sencillo a la hora de escribir un analizador semántico del mismo y los datos en JSON ocupan mucho menos que en XML. Además de que es fácil de leer para los humanos y máquinas y su procesamiento por parte de los ordenadores es rápido, pues necesita librerías muy pequeñas para trabajar con él. (json.org,2010)

Se emplea JSON ya que basado en sus características permite una mayor simplicidad e interoperabilidad en la trasferencias de datos; siendo usado por distintas tecnologías, lo cual, posibilita su lectura por varios lenguajes como PHP y XML, y garantiza un alto nivel seguridad en la integración de la información.

Lenguaje para la representación del modelo de datos XML

XML (Lenguaje de Etiquetado Extensible), conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades y contiene tres características muy importantes que

son: extensibilidad, estructura y validación. Este lenguaje facilita la integración desde fuentes de datos heterogéneas, por ejemplo, páginas web y distintas bases de datos; permite proporcionar diferentes vistas sobre los datos (HTML, PDF), dependiendo de quién sea el cliente. (O'Reilly, 2010)

Se utiliza XML ya que proporciona una independencia respecto a las herramientas de software que se utilicen; proporcionando uniformidad en los paquetes de datos, permitiendo estructurar la información de la manera más adecuada, posibilitando generar la misma en diferentes formatos con el objetivo de ofrecer una representación visual para el usuario.

1.7.5. Marcos de trabajo

Dentro del ambiente de desarrollo de software, un marco de trabajo es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, sirviendo de base para que otro proyecto de software pueda ser fácilmente organizado y desarrollado. Puede incluir soporte de programas, librerías y otras herramientas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Los principales objetivos de los marcos de trabajo son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. El cumplimiento de estos objetivos, facilita el desarrollo del software, permitiendo a diseñadores y programadores concentrarse en los requerimientos del proyecto, reduciendo los posibles problemas con las tecnologías utilizadas, así como facilitando ciertas funcionalidades básicas y comunes.(Bernal Rojas, 2013)

Symfony 2.0.18

Es un marco de trabajo que facilita y automatiza una gran cantidad de funcionalidades que resultan comunes a la hora de realizar algunos tipos de software. Permite optimizar el desarrollo de las aplicaciones web y está desarrollado completamente con PHP 5, siendo sencillo de usar pero lo suficientemente flexible como para adaptarse a los casos más complejos. Es una herramienta fácil de instalar y configurar en la mayoría de las plataformas ya que se puede ejecutar tanto en sistemas Windows y Unix estándares. Independiente del sistema gestor de bases de datos, su capa de abstracción y el uso de Doctrine, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto. (Gazquez Martinez, 2011)

Se emplea Symfony como framework porque se basa en el patrón arquitectónico Modelo-Vista-Controlador. Está desarrollado completamente en PHP. Ofrece un código fuente transparente que permite el entendimiento y soporte del mismo. Adopta un estado multiplataforma permitiendo no simplemente la independencia con sistemas de aplicaciones y de base datos, sino además que ejerce los patrones de diseño y las buenas prácticas establecidas para el desarrollo web.

Ext-JS 3.4.0

Es un framework JavaScript para la creación de aplicaciones enriquecidas del lado del cliente haciendo un uso intensivo de las tecnologías AJAX, XHTML, DHTML y DOM. Sus características principales son: gran desempeño, componentes de interfaz de usuario personalizables, con buen diseño y documentación. Sirve de puente entre las librerías JS más usadas (Prototype, JQuery, YUI), debido a que se inició como una extensión de YUI (Yahoo User Interface) esta presenta una cierta ventaja de compatibilidad respecto a las otras dos.(Orihuela Sánchez, 2013)

Se emplea ExtJS porque contiene un conjunto de componentes que permiten desarrollar interfaces que están integradas en la misma librería, lo cual, es idóneo para desarrollar aplicaciones de gestión con apariencia uniforme verdaderamente profesional. Además de ello, prescinde de la programación orientada a objetos prototípica clásica y se basa en la arquitectura MVC, posibilitando organizar un código que depende de innumerables transacciones de datos posiblemente reutilizables.

1.7.6. Entorno de Desarrollo

NetBeans 8.0

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

comunes en las aplicaciones web como son: la refactorización, la validación de código así como el soporte para lenguajes o script.(Hernandez Carvajal, 2012)

Se utiliza NetBeans 8.0 porque es una herramienta multiplataforma que permite desarrollar aplicaciones en PHP, poseer un excelente balance entre una interfaz con múltiples opciones, un aceptable completamiento de código y poseer soporte para el framework Symfony.

Lycan GENESIS - Component Builder

Es un entorno para el desarrollo de aplicaciones enriquecidas para la web sobre las tecnologías Ext JS,SQLite y Symfony soportado sobre el lenguaje PHP. Permite el desarrollo de componentes de una forma rápida y cómoda, se reutiliza y configura para desarrollos más especializados en otros dominios. Es una herramienta multiplataforma que posee el respaldo institucional del Centro DATEC de la UCI. (García Quintela, 2013)

Se emplea para el desarrollo del componente tabla cruzada por permitir la administración de componentes de manera sencilla y personalizable para el usuario así como su fácil integración con el sistema GDR v2.0.

1.7.7. Servidor de Aplicaciones

Tipo de servidor que permite el procesamiento de datos de una aplicación de cliente.

Las principales ventajas de la tecnología de los servidores de aplicación es la centralización y la disminución de la complejidad del desarrollo de aplicaciones, dado que las aplicaciones no necesitan ser programadas; en su lugar, estas son ensambladas desde bloques provistos por el servidor de aplicación.(Alegs, 2008)

Apache 2.4

Constituye una tecnología gratuita de código abierto. Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Permite la creación de sitios web dinámicos mediante el uso de Server Side Includes (SSI), de lenguajes de scripting como PHP y JavaScript. Se ejecuta en varios sistemas operativos. Posee gran cantidad de extensiones para diversas tecnologías, además de

una amplia documentación, es libre, modular y multiplataforma. Se señala como desventaja que no posee interfaz gráfica que facilite su configuración.(Lobo, 2009)

Se utiliza la versión Apache 2.4 porque permite una fácil integración con diversos lenguajes de programación como PHP. Ofrece un alto nivel de soporte en términos de uso de caché; es multiplataforma y extensible, lo cual, es idóneo para su uso en tecnologías webs.

1.7.8. Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD), es una interfaz para que el usuario se pueda comunicar dinámicamente con la base datos. Estos sistemas manejan de forma clara la información de modo que su uso es bastante transparente, disminuyendo el tiempo de desarrollo y aumentando la calidad del sistema. Entre los principales SGBD que existen en la actualidad a nivel mundial se encuentra el privativo Oracle y los libre MySQL y PostgreSQL, tales gestores brindan similares prestaciones las cuales son necesarias para la transformación de BD.(Medinilla Santo y Álvarez Rojas, 2013)

PostgreSQL 9.3

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

Se emplea PostgreSQL en su versión 9.3 por ser un gestor de base de datos estable, robusto, con fuerte coherencia, seguro y conforme con el estándar SQL, así como por incluir entre sus nuevas características vistas actualizables automáticas que mejoraran la confiabilidad y disponibilidad del mismo.

PgAdmin III 1.8

Es una herramienta para la administración del gestor de base de datos PostgreSQL. Esta permite crear simples consultas SQL o consultas de una alta complejidad de una manera más fácil, debido a que se le ha incorporado una nueva interfaz gráfica, la cual, resulta de gran ayuda en el momento de gestionar las bases de datos. Es una herramienta de código abierto respaldada por una amplia comunidad de desarrolladores que encaminan sus esfuerzos al perfeccionamiento de la misma. Esta aplicación incluye un editor de resaltado de sintaxis SQL, una interfaz administrativa gráfica y una herramienta de consulta SQL. La conexión al servidor se puede hacer a través de protocolo TCP/IP. No se requieren controladores adicionales para comunicarse con el servidor de base de datos.(Bernal Rojas, 2013)

Se utiliza la versión 1.8 de PgAdmin III por ser estable, confiable y poseer un conjunto facilidades que permiten de una forma profesional la elaboración y manipulación de las bases de datos.

Conclusiones Parciales

En este capítulo se realizó un estudio detallado de los diferentes generadores de reportes y el componente tabla cruzada en los mismos, mostrando como resultado características claves que servirán de base para el diseño de la propuesta de trabajo. Se analizaron las herramientas y tecnologías que posibilitarán la inclusión de dicho componente en los reportes generados del sistema GDR v2.0 partiendo de las premisas de software libre y código abierto.

Basado en lo anteriormente dicho se decidió utilizar como metodología de desarrollo a OpenUP por ser una metodología objetiva que permite adaptarse a la solución con carácter eficiente y como herramienta de modelado Visual Paradigm para elaborar los artefactos de la misma. Para el desarrollo de la aplicación se utilizan como lenguajes de programación del lado del servidor PHP y JavaScript del lado del cliente. Como marcos de trabajo para el desarrollo de aplicaciones Web se seleccionaron a ExtJS del lado del cliente y Symfony, que implementa el estilo arquitectónico Modelo-Vista-Controlador. El gestor de base de datos elegido es PostgreSQL y como administrador de PostgreSQL se utiliza PGAdmin III. En aras de lograr la organización del código se seleccionó el entorno integrado de desarrollo Netbeans con soporte para los lenguajes de programación a utilizar, cumpliendo con el estándar de codificación definido por el centro de desarrollo DATEC.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

Introducción

En el presente capítulo se profundiza en las actividades específicas del proceso de análisis y diseño, permitiendo así asegurar una organización del software para alcanzar resultados satisfactorios, donde para ello, se muestra la estructura del sistema a desarrollar a través de una representación visual de las clases conceptuales en el modelo de dominio. Se definen los requisitos funcionales y no funcionales, el diagrama de casos de uso del sistema y el diagrama de clases del diseño para mostrar la estructura estática del sistema. Además, se especifica la estructura física de la solución que se propone mediante el diagrama de despliegue.

2.1. Modelo de Dominio

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

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

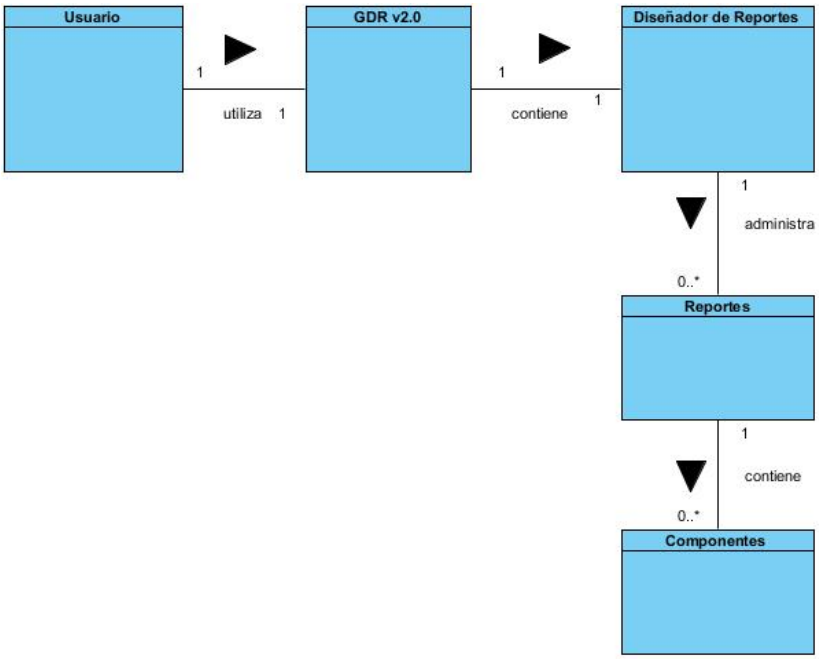


Figura 3. Modelo de dominio.

Descripción de las clases del dominio

Usuario: se comporta como el diseñador de reportes que interactúa con el sistema Generador Dinámico de Reportes, específicamente con el módulo Diseñador de Reportes.

GDR v2.0: entidad que representa al sistema Generador Dinámico de Reportes en su versión 2.0.

Diseñador de Reportes: representa uno de los módulos que contiene el sistema GDR v2.0, el mismo, funciona como administrador de reportes permitiendo crear, modificar y eliminar reportes.

Reporte: Informe que transmite una información mediante un conjunto de elementos

Componentes: elementos contenidos en un reporte que permiten ofrecer una serie de servicios o funcionalidades a través de interfaces definidas.

2.2. Requisitos del sistema

Los requisitos son condiciones o capacidades que el sistema debe cumplir, definiendo que es lo que debe hacer. Para esto se identifican las funcionalidades requeridas y las restricciones que se imponen, clasificándose en: funcionales y no funcionales.(Orihuela Sánchez, 2013)

2.2.1. Requisitos Funcionales

Los requisitos funcionales (RF) definen el comportamiento interno de un software, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software.(Orihuela Sánchez, 2013)

El componente tabla cruzada debe cumplir con los requisitos funcionales que a continuación se describen:

RF1: Adicionar componente tabla cruzada en el reporte.

Descripción: El sistema permitirá al usuario crear un componente tabla cruzada en el reporte.

Entradas: se especifican los detalles del nuevo componente tabla cruzada que se va a adicionar (modelo de datos, agrupación por filas, columnas, función estadística y diseño).

Salidas: se actualizan los cambios y se el nuevo componente tabla cruzada creado.

RF2: Modificar propiedades del componente tabla cruzada**.**

Descripción: El sistema permitirá al usuario modificar las propiedades de los componentes tablas cruzadas existentes en el reporte.

Entradas: se selecciona el componente tabla cruzada que se desea modificar del reporte.

Salidas: se actualizan los cambios y se muestra las propiedades que se modificaron.

RF3: Transformar visualmente el componente tabla cruzada en el reporte.

Descripción: El sistema permitirá al usuario transformar los componentes tablas cruzadas existentes en el reporte**.**

Entradas: se selecciona el componente tabla cruzada que se desea transformar del reporte.

Salidas: se transforma el componente tabla cruzada seleccionado.

RF4: Posicionar componente tabla cruzada en el reporte.

Descripción: El sistema permitirá al usuario posicionar componentes tablas cruzadas existentes en el reporte.

Entradas: se selecciona el componente tabla cruzada que se desea posicionar en el reporte.

Salidas: se posiciona el componente tabla cruzada seleccionado.

RF5: Eliminar componente tabla cruzada en el reporte.

Descripción: El sistema permitirá al usuario eliminar componentes tablas cruzadas existentes en el reporte**.**

Entradas: se selecciona el componente tabla cruzada que se desea eliminar del reporte.

Salidas: se elimina el componente tabla cruzada seleccionado.

RF6: Generar vista XML del componente tabla cruzada.

Descripción: permitirá visualizar un diseño XML del componente tabla cruzada.

RF7: Generar vista previa del componente tabla cruzada.

Descripción: permitirá visualizar una vista previa del componente tabla cruzada.

2.2.2. Requisitos No Funcionales

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

Software

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

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 14.04 .
- Paquetes:

✓ **Comunes**: apache2, php5, libapache2-modphp5, php5-cli, php5-xsl, php5-gd.
✓ **PostgreSQL:** php5-pgsql.
✓ **SQLite:** php5-sqlite.
✓ **MySQL:** php5-mysql.

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

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 14.04.
- PostgreSQL versión 9.3.
- PgAdmin III 1.18.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP.

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

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 hasta 14.04, Debian 4 GNU/Linux, Microsoft Window XP.
- Navegador Web Mozilla Firefox 2.0 a la versión 35.0.

Hardware

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

- Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- Como mínimo requiere 1GB de memoria RAM.
- Necesita espacio en disco duro igual o superior a 40 GB.

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

- Procesador Intel Pentium 4 1.7 GHz, o AMD similar.
- 1 GB RAM o más.
- 20 GB de espacio en disco duro o más.

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

- Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- Necesita memoria RAM igual o superior a 1 GB.
- Como mínimo requiere 40 GB de espacio en disco duro.

Restricciones de Diseño e Implementación

- El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.3 o superior.
- Se utilizará el framework de desarrollo Symfony en su versión 2.0.18 y la librería de JavaScript ExtJS versión 3.4.0.
- Se empleará la herramienta de desarrollo NetBeans 8.0 y el sistema gestor de base de datos PostgreSQL 9.3.

Interfaz

- Las interfaces de usuario serán diseñadas a modo de aplicaciones RIA (Rich Internet Application), lo que permite a los usuarios contar con aplicaciones web con una experiencia de usuario similar a la de las aplicaciones de escritorio.

- El sistema deberá contar de manera general, con una interfaz sencilla, que sea intuitiva y de fácil comprensión garantizando la correcta interacción con el sistema.
- El usuario deberá acceder a la aplicación a través del protocolo HTTP usando el navegador Firefox en su versión 4.0 hasta la 35.
- El sistema debe tener indicadores que permitan al usuario conocer las acciones que debe realizar, por ejemplo: botones con íconos sugerentes y alternativa textual.
- El sistema debe permitir al usuario transitar de una tarea a otra sin necesidad de obligarlo a realizar acciones innecesarias o no deseadas, por ejemplo, para llegar de una tarea a otra el usuario no debe dar más de 3 clic

Usabilidad

- La aplicación garantiza una fácil interacción entre cliente y PC, de tal forma que no haya conflictos de usabilidad entre ambos.
- El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
- El usuario debe poder diseñar una consulta SQL de manera ágil y sencilla, sin ser un experto en el proceso de diseño de un reporte.
- La información deberá estar disponible en todo momento, limitada solamente por las restricciones de acuerdo a las políticas de seguridad definidas.

Eficiencia

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

2.3. Modelo de casos de uso del sistema

Un modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema. Los casos de uso (CUS) son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores, estos especifican una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus

actores, incluyendo alternativas dentro de la secuencia. Los casos de uso son el componente clave del modelado. Su propósito es ilustrar como un sistema permite a un actor cumplir una meta, ilustrando todos los posibles caminos apropiados que ellos pueden tomar para cumplirla, así como las situaciones que podrían hacerlo fallar.(Bernal Rojas, 2013)

Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema (DCUS) documenta el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar. Para el modelado del diagrama de casos de uso del sistema, se tuvo en cuenta el uso de patrones de casos de uso. Estos patrones son un conjunto de comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer y cómo este interactúa con los usuarios. Generalmente son utilizados como plantillas que describen cómo debería ser estructurados y organizados los casos de uso, capturando así mejores prácticas para modelar casos de uso. Entre estos patrones, el que se evidencia en el diagrama de casos de uso del sistema es: CRUD Parcial, este consiste en definir un subconjunto de casos de uso (adicionar, modificar, eliminar, transformar y posicionar) y agrupar estos flujos en un mismo caso (Administrar componente tabla cruzada) para que dichas operaciones estén presentes en el modelo.

A continuación se presenta el DCUS del componente tabla cruzada.

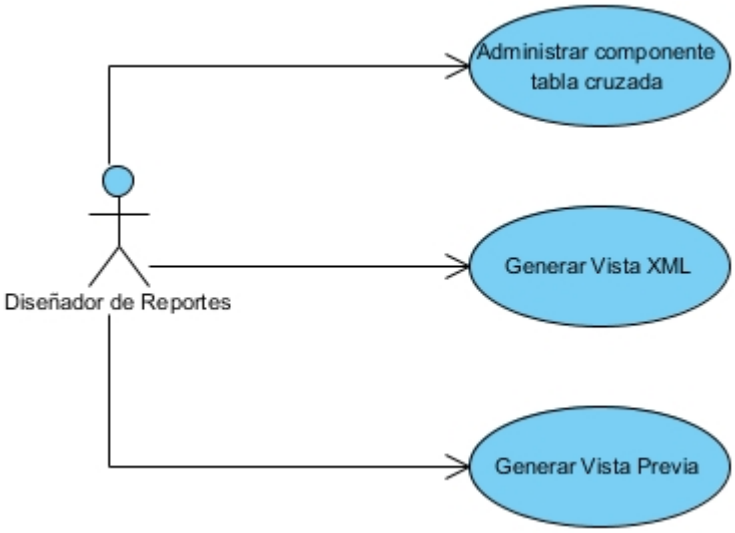


Figura 4. Diagrama de Casos de Uso del Sistema.

En este DCUS, el actor Diseñador de Reportes es el encargado de inicializar los caso de usos Administrar componente tabla cruzada, considerado crítico por su impacto en la arquitectura del componente, así como Generar Vista XML y Generar Vista Previa basándose en la estructura del mismo.

Tabla 2. Descripción del Caso de Uso Administrar componente tabla cruzada

Objetivo	Administrar los componentes tablas cruzadas en los reportes que genera el GDR v2.0
Actores	Diseñador de reportes
Resumen	El caso permite adicionar varios componentes tablas cruzadas a los reportes teniendo en cuenta aspectos esenciales como un modelo de datos, así como modificar, transformar, posicionar y eliminar los mismos una vez incorporados.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Debe existir al menos un modelo de datos.

Postcondiciones	Se adiciona un componente tabla cruzada en el reporte generado por el GDR v2.0
Flujo de eventos	
Flujo básico < Administrar componente tabla cruzada>	
Actor	
El usuario accede al módulo Diseñador de Reportes y realiza una de las siguientes acciones : a) Adicionar componente tabla cruzada en el reporte. b) Modificar propiedades del componente tabla cruzada en el reporte. c) Transformar visualmente el componente tabla cruzada en el reporte. d) Posicionar componente tabla cruzada en el reporte. e) Eliminar componente tabla cruzada del reporte.	En dependencia de la acción realizada por el usuario, el sistema ejecuta una de las siguientes funciones : a) Adicionar componente tabla cruzada en el reporte. Ver **sección "Adicionar componente tabla cruzada".** b) Modificar propiedades del componente tabla cruzada en el reporte. Ver **sección "Modificar propiedades del componente tabla cruzada".** c) Transformar visualmente el componente tabla cruzada en el reporte. Ver **sección "Transformar componente tabla cruzada".** d) Posicionar componente tabla cruzada en el reporte. Ver **sección "Posicionar componente tabla cruzada".** e) Eliminar componente tabla cruzada en el reporte. Ver **sección "Eliminar componente tabla cruzada. "**
Sección 1 : "Adicionar componente tabla cruzada"	
Flujo básico <Adicionar componente tabla cruzada>	
Actor	**Sistema**

1. Adicionar componente tabla cruzada.	2. Muestra una ventana de configuración del componente tabla cruzada, mostrando un listado de los modelos de datos existentes.
3. Selecciona un modelo de datos.	
4. Presiona el botón **Siguiente.**	5. Muestra una serie de campos para seleccionar los grupos de fila.
6. Selecciona uno o varios grupos de filas.	
7. Presiona el botón **Siguiente.**	8. Muestra una serie de campos para seleccionar los grupos de columnas.
9. Selecciona uno o varios grupos de columnas.	
10. Presiona el botón **Siguiente.**	11. Muestra listado con las funciones estadísticas disponibles.
12. Selecciona una de las funciones estadísticas.	
13. Presiona el botón **Siguiente.**	14. Muestra una serie de opciones: color, variación, seleccionar totales por fila, y por columna para configurar el diseño del componente.
15. Selecciona los elementos deseados para el diseño del componente tabla cruzada.	
16. Presiona el botón **Terminar**.	17. Cierra la ventana de configuración del componente tabla cruzada y muestra el componente tabla cruzada en el reporte.
Flujo alternos <Cancelar adicionar componente tabla cruzada>	
Actor	Sistema
16.1 Presiona el botón **Cancelar.**	Cierra la ventana de configuración del componente tabla cruzada.
Sección 2 : "Modificar propiedades del componente tabla cruzada"	

Flujo básico < Modificar componente tabla cruzada>	
Precondiciones	Que exista al menos un componente tabla cruzada en el reporte
Actor	Sistema
1. El usuario selecciona el componente tabla cruzada que desea modificar.	2. Muestra las propiedades del componente tabla cruzada seleccionado.
3. Selecciona una propiedad del componente tabla cruzada y modificar su valor.	4. Actualiza la propiedad modificada, mostrando el valor introducido.
Sección 3: "Transformar componente tabla cruzada en el reporte "	
Flujo básico < Transformar componente tabla cruzada>	
Precondiciones	Que exista al menos un componente tabla cruzada en el reporte
	Actor / **Sistema**
1. El usuario selecciona el componente tabla cruzada y cambia el tamaño del mismo.	2. Realiza la transformación del componente tabla cruzada en el reporte de acuerdo a la ubicación definida por el usuario y los límites de dimensión del componente.
Sección 4: "Posicionar componente tabla cruzada"	
Flujo básico < Posicionar componente tabla cruzada>	
Precondiciones	Que exista al menos un componente tabla cruzada en el reporte.
Actor	**Sistema**
1. El usuario selecciona el componente tabla cruzada y posiciona el mismo en un lugar específico del reporte.	2. Realiza el posicionamiento del componente tabla cruzada en el reporte de acuerdo a la ubicación definida por el usuario.
Sección 3: "Eliminar componente tabla cruzada"	

Flujo básico < Eliminar componente tabla cruzada>	
Precondiciones	Que exista al menos un componente tabla cruzada en el reporte.
Actor	**Sistema**
1. El usuario selecciona el componente tabla cruzada a eliminar.	2. Muestra una ventana emergente para verificar la eliminación del componente tabla cruzada seleccionado.
3. Presiona el botón **Sí.**	4. Elimina el componente tabla cruzada.
Flujo alternos <Cancelar eliminar componente tabla cruzada>	
Actor	Sistema
4.1 Presiona el botón **No.**	Cierra la ventana de configuración del componente tabla cruzada.

Tabla 2. Descripción del CUS Administrar componente tabla cruzada.

2.4. Patrones arquitectónicos

Los patrones son de gran utilidad para describir las mejores prácticas y los buenos diseños. Constituyen mecanismos cuyo objetivo es la solución de problemas que ocurren repetidamente dentro de un contexto muy bien definido. Definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

Para crear un diseño eficiente se decidió utilizar uno de los patrones arquitectónicos más conocidos para el diseño Web, el Modelo-Vista-Controlador (MVC), el cual, se aplica en la presente solución. El MVC permite separar las capas: Modelo (representa la información con la que trabaja la aplicación, es decir, su lógica de negocio), Vista (transforma el modelo en una página Web que permite al usuario interactuar con ella) y Controlador (se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista).

La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes pocos robustos y difíciles de mantener, entender, reutilizar o extender. Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos.(Lobo, 2009)

2.5. Patrones utilizados en la solución

La sociedad requiere sistemas más complejos y los recursos desarrollados son cada vez más escasos, por tanto se hacen imprescindibles los mecanismos de reutilización. Ante la necesidad de asimilar la reutilización en el desarrollo de software se han popularizado mecanismos como: componentes, framework, objetos distribuidos y patrones de diseño.

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

2.5.1. Patrones de diseño GRASP

En los patrones generales de software para asignación de responsabilidades (GRASP) se codifican algunos de los principios, que se aplican al diseñar los diagramas de interacción. Una de las principales ventajas a señalar en el uso del framework. Symfony es que está basado en patrones de diseño. Los patrones de diseño empleados en la solución pertenecen al conjunto de patrones GRASP, su utilización ayudó a refinar el diseño y a asignar las responsabilidades de las distintas clases de diseño, haciéndolas más sencillas, reutilizables y encapsuladas.(García Suárez del Villar, 2012)

A continuación se explican los patrones utilizados:

Experto: El patrón experto en información es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). La clase Crosstab.js

se encarga de realizar dicha operación ya que contiene las características necesarias para generar el componente tabla cruzada.

Creador: Identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Se brinda un soporte al bajo acoplamiento. La clase Crosstab.js es la encargada de realizar dicha operación ya que tiene la información requerida para realizar una instancia del componente tabla cruzada.

Alta Cohesión: Plantea que la información que almacena una clase debe ser coherente y estar en la mayor medida relacionada con la clase. Estas clases mejoran la claridad y la facilidad con que se entiende el diseño, además simplifican el mantenimiento y las mejoras en funcionalidad. La clase Crosstab.js se encarga de centralizar la información necesaria para generar el componente tabla cruzada.

Bajo acoplamiento: Este patrón guía la asignación de responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible, de manera que de producirse una modificación en algunas de ellas se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las mismas. La clase Crosstab.js que permite dicha operación encapsulando las posibles clases que se relacionen únicamente con el componente tabla cruzada.

Controlador: Asigna la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase. Patrón que establece el uso de una clase controladora, que funciona como intermediaria entre cada una de las clases interfaces y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y manda a ejecutar las acciones correspondientes. La clase DesignerController.js es la encargada de realizar dicha operación devolviendo un objeto de especificación del componente solicitado por parámetro.

2.6. Modelo del Diseño

El Modelo de Diseño se utiliza para documentar el diseño de un sistema. Es un modelo de objeto que describe la realización de los casos de uso, y sirve como una abstracción del Modelo de Implementación y del código fuente. Se utiliza como entrada esencial para las actividades en el flujo de trabajo Implementación y en el flujo de trabajo Prueba. Es un artefacto integral que abarca todas las clases del diseño y sus relaciones, e incluye los diagramas de clases y de interacción del diseño.(Pressman, 2010)

2.6.1. Diagramas de clases del diseño

En la fase de diseño del software se hace un refinamiento del análisis y se centra en cómo van a ser implementados los requerimientos, logrando adaptar el diseño para que se ajuste al entorno de implementación, persiguiendo así el desarrollo de una arquitectura estable y sólida.(Orihuela Sánchez, 2013)

El diagrama de clases del diseño (DCD) muestra los atributos y métodos de cada clase, representando de forma sencilla la colaboración y las responsabilidades de cada una de ellas, entorno al sistema que conforman.(Ennis Bouly, 2013)

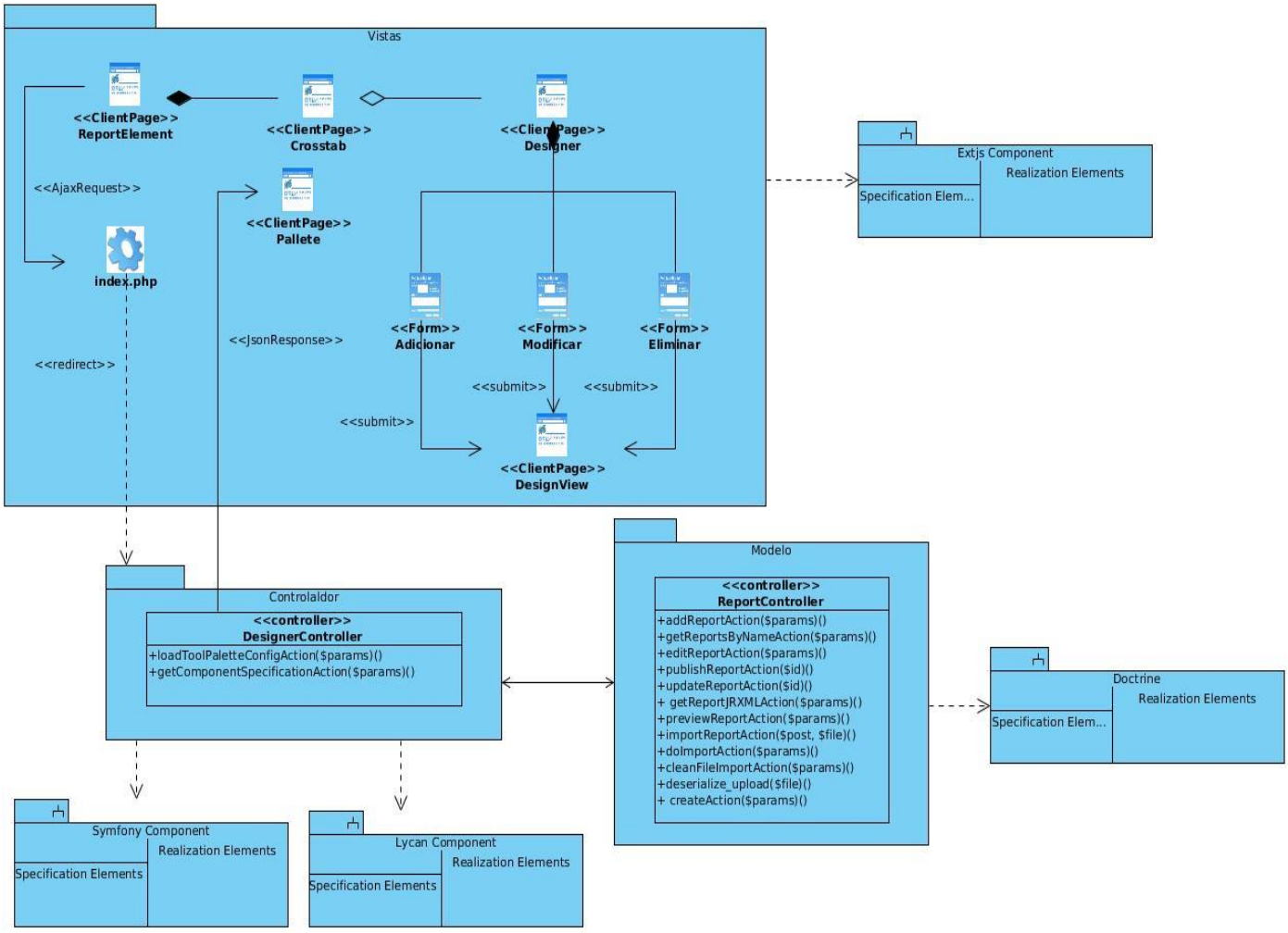


Figura 5. Diagrama de clases del diseño CUS Administrar componente tabla cruzada.

En este diagrama se representa la integración del componente tabla cruzada con el GDR v2.0, donde los elementos del paquete “Modelo” corresponden a los métodos generados por la clase ReportController que permiten recibir y dar respuesta a las solicitudes del paquete “Controlador”, auxiliándose de la herramienta de mapeo de objeto-relacional Doctrine, modelado por el subsistema de igual nombre. Las acciones son los elementos que construyen las respuestas del servidor de acuerdo a las peticiones realizadas por los elementos del paquete “Vista”, estas se encuentran agrupadas mediante la clase Designer Controller del paquete “Controlador”, con la dependencia de los subsistemas “Componentes de Symfony” y “Componentes de Lycan”. Las solicitudes se realizan por medio de la tecnología AJAX y las respuestas son devueltas en

formato JSON. Los elementos del lado del cliente corresponden a elementos específicos del negocio como paleta de componentes, paneles de trabajos y vista de diseño asociados a los elementos genéricos fuertemente reutilizables e implementados en lenguaje JavaScript utilizando los componentes del subsistema “Componentes Ext-JS”.

2.7. Diagrama de Despliegue

Los diagramas de despliegue muestran las relaciones físicas entre los componentes de hardware y software en el sistema final. Están compuestos por nodos, dispositivos y conectores, donde los nodos son elementos de procesamiento, los dispositivos son nodos estereotipados sin capacidad de procesamiento y los conectores expresan el tipo de conector utilizado entre el resto de los elementos del modelo.(Orihuela Sánchez, 2013)

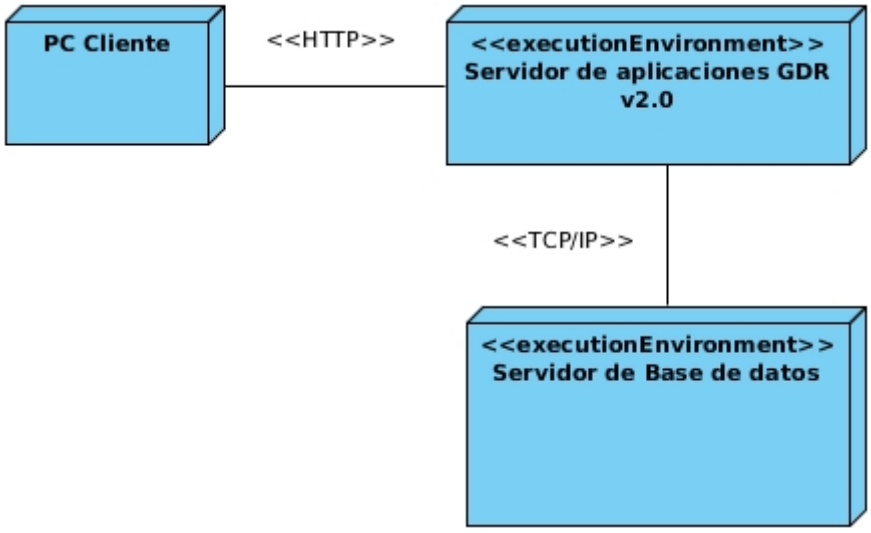


Figura 6. Diagrama de despliegue.

En la solución, el diagrama de despliegue se distribuye por una estación de trabajo para el cliente, desde la cual, se realizarán peticiones por el protocolo HTTP al servidor web representado por el nodo “Servidor de Aplicaciones GDR v2.0”, el cual, accederá a los datos de la aplicación almacenados en un Servidor de Base de Datos mediante el protocolo TCP/IP.

Conclusiones Parciales

En el desarrollo del presente capítulo se realizó el modelo de dominio donde se describieron las clases conceptuales que ayudarán a comprender los conceptos claves para un mejor dominio del problema. Además se identificaron 7 requisitos funcionales agrupados en 3 casos de uso y los requisitos no funcionales que el sistema debe cumplir, basados en software, hardware, interfaz, eficiencia, usabilidad y restricciones de diseño e implementación. También se realizó el diagrama de caso de uso del sistema, estableciendo el caso de uso Administrar componente tabla cruzada como crítico por su impacto en la arquitectura del componente. Se definieron los patrones arquitectónicos para darle solución a la problemática. De igual manera, se realizaron los diagramas de clases del diseño apoyadas en las particularidades que tiene la arquitectura de Symfony por el patrón Modelo-Vista-Controlador y el diagrama de despliegue que describió como se realizará el despliegue de los componentes a lo largo de la infraestructura del sistema, para así dar comienzo a la implementación de la aplicación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

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

3.1. Implementación

Esta disciplina explica cómo desarrollar, organizar, realizar pruebas de unidad e integrar los componentes implementados basándose en las especificaciones del diseño. Tiene la finalidad de definir la organización del código, en términos de los subsistemas de implementación, organizados en capas. Además se implementan los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables) y permite probar y desarrollar componentes como unidades, así como integrar los resultados producidos por los implementadores individuales, o equipos en un sistema ejecutable. (IBM, 2007)

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

3.1.1. Diagrama de Componentes

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

generalización, asociación, agregación o realización. Además contienen las interfaces que constituyen la unión entre varios componentes. Los componentes pueden agruparse en paquetes y pueden contener subsistemas. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se modelan por partes; cada diagrama describe un apartado del sistema. En él se sitúan librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Además contienen las interfaces que constituyen la unión entre varios componentes. Uno de los usos principales es que pueden servir para mostrar qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.(Orihuela Sánchez, 2013)

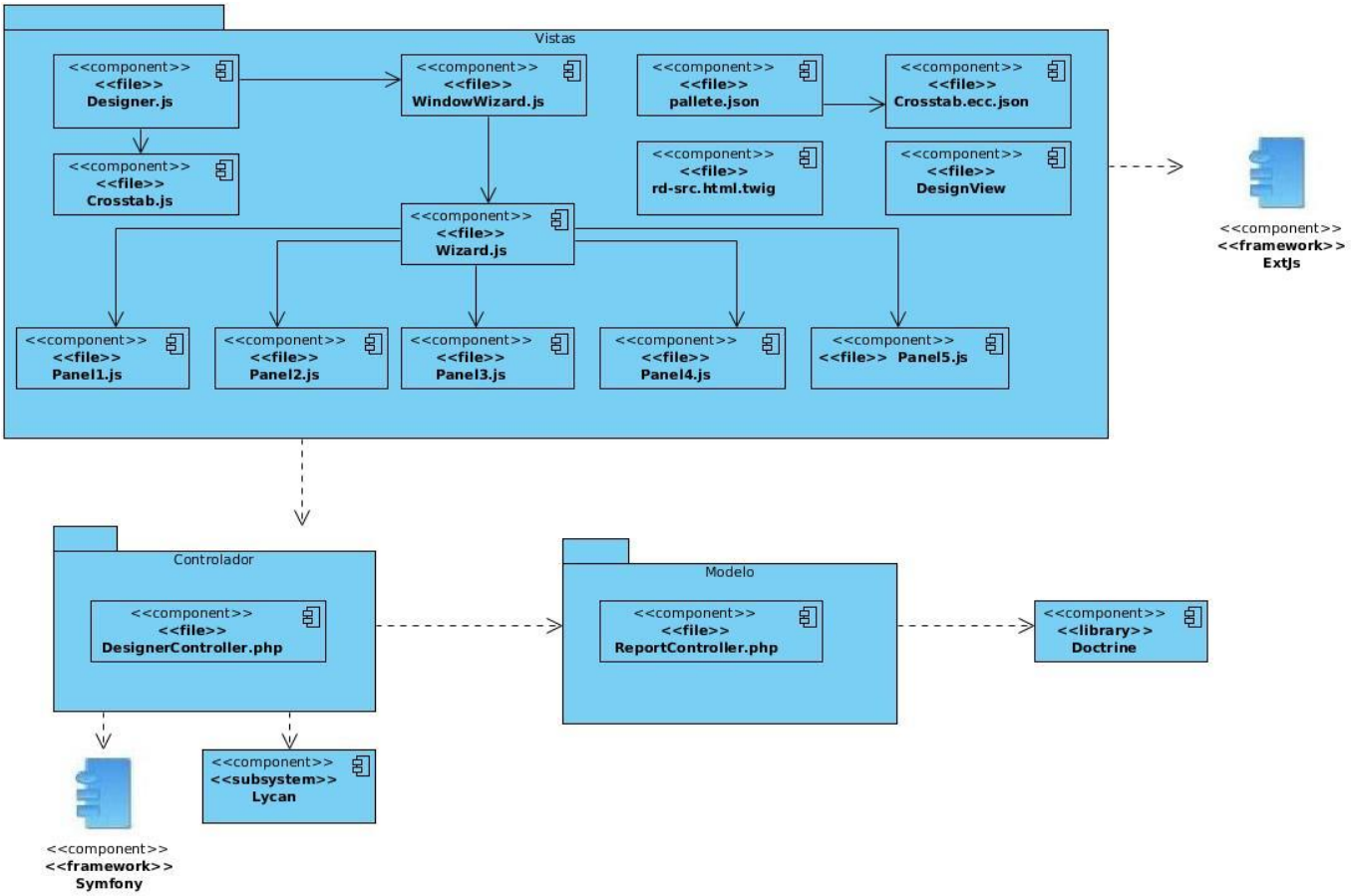


Figura 7. Diagrama de componentes del CUS Administrar componente tabla cruzada.

El diagrama de componente del CUS “Administrar componente tabla cruzada” incluye 3 paquete genéricos:

- **Paquete Vista:** agrupa los elementos que permiten la interacción directa con el usuario, representando la información que recibe del controlador.
- **Paquete Controlador:** contiene las clases que manipulan los eventos del usuario, escuchando las solicitudes del mismo mediante peticiones AJAX para devolver como respuesta un componente en formato JSON.
- **Paquete Modelo**: agrupa los elementos que interactúan con la base de datos con el objetivo de recibir las instrucciones del controlador, devolviendo los datos solicitados.

3.2. Código fuente

El código fuente, también llamado código base, es un texto que se ha escrito en un lenguaje de programación concreto, y que sólo puede ser leído por un experto o programador. Estos caracteres deben ser traducidos a un lenguaje que se denomina código máquina, el cual, podrá ejecutar cualquier ordenador. También puede ser traducido a un lenguaje llamado códigos de *bytes*, el cual podrá ser traducido por un intérprete. Este tipo de transferencias y traducciones se llaman compilación.(ethek, 2010)

3.2.1. Estándares de codificación

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

El estándar de codificación seleccionado para el desarrollo de la aplicación es el conocido como camelCase. Es un estilo de escritura que se aplica a frases o palabra compuesta, dentro de este estilo existen dos tipos lowerCamelCase que define la primera letra de cada palabra en mayúscula excepto la primera que la pone

en minúscula completa y el UpperCamelCase, el cual, define el comienzo de cada palabra en mayúscula; ambos permiten un estilo limpio, claro y comprensivo para mejorar el estilo de programación.

Para cumplir con estos paradigmas se establece el estilo siguiente:

Comentarios

Para los comentarios de más de una líneas se escriben comenzando con “/*” y terminando con “*/”, y “//” para los de una sola línea.

Declaraciones

Las variables y atributos deben ser explícitas para saber realmente lo que significan y utilizando letra minúscula.

Para declarar una clase se comienza con letra minúscula y en caso de necesitar escribir otra palabra para hacer la misma más entendible se comienza con mayúscula para lograr una mejor legibilidad de la palabra.

Deben escribirse comentarios al principio de cada clase y método brindando una breve descripción de los propósitos generales de cada funcionalidad.

Llaves

Colocar al lado de la declaración y al final del código, ya sean clases, métodos o instrucciones.

3.3. Pruebas del software

Las pruebas del software son el proceso que permite verificar, garantizar y mostrar la calidad de un producto, a través de resultados registrables que proporcionan una evaluación y representa una revisión final de las especificaciones, del diseño y de la codificación. Son empleadas para identificar posibles fallos durante el proceso de desarrollo. Los casos de prueba son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, permitiendo encontrar y documentar los defectos que puedan afectar la calidad del software.(Brito Rodriguez, 2012)

3.3.1. Estrategia de Prueba

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

Nivel de Prueba

Los niveles de prueba especifican diferentes ángulos para verificar y validar un producto de software. Existen varios niveles de pruebas como: pruebas de desarrollador, independiente, unidad, integración, sistema y aceptación, donde cada nivel contiene una técnica de prueba específica según los atributos de calidad que se deseen verificar. En el desarrollo del componente se aplicarán solamente las pruebas siguientes para comprobar que el sistema da respuesta a los requisitos funcionales definidos anteriormente.(Carrillo, 2011)

- Nivel de Desarrollador: es el primer nivel de pruebas, diseñadas e implementadas por el equipo de desarrollo, donde el programador es quien revisa su código fuente.

- Nivel de Integración: en este nivel se prueba los componentes combinados para ejecutar un CUS. Además se realiza la prueba para descubrir errores en las especificaciones de las interfaces de las clases.

Tipo de Prueba

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

Pruebas Funcionales: son aquellas que tienen por objetivo demostrar que los sistemas desarrollados, cumplen con las funciones específicas para los cuales han sido creados incluyendo la navegación, entrada

de datos, procesamiento y obtención de resultados. Es común que sea desarrollada por los analistas de pruebas con apoyo de algunos usuarios finales y están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.(Orihuela Sánchez, 2013)

Pruebas de Integración: son un conjunto de pruebas unitarias, funcionales, regresión y aceptación que se realizan para probar el software. Incluye también comprobar que lo programado por los diferentes desarrolladores funciona en un entorno real.(Orihuela Sánchez, 2013)

Método de Prueba

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

Este método intenta encontrar errores de las siguientes categorías:

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

La ventaja fundamental del método de Caja Negra es que permite identificar claramente las entradas y salidas además de estudiar las relaciones que existen entre ellas. Además maximiza la eficiencia de los sistemas sin tener que introducirnos en los todos los procesos del software, exceptuando cuando se presentan anomalías en las relaciones de entrada y salida.(Orihuela Sánchez, 2013)

3.4. Diseño de Caso de Prueba

Un caso de prueba se diseña a partir de las funcionalidades descritas en los casos de usos y se realizan con el objetivo de comprobar que las funcionalidades han sido implementadas según las peticiones del cliente. Para cada caso de uso debe haber una planilla de caso de prueba donde se recoge la especificación

del caso de uso, dividido en secciones y escenarios detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión.

A continuación se presentan las tablas de la sección probada para el caso de uso Administrar componente tabla cruzada.

Tabla 3. Sección de prueba para el CUS_Administrar componente tabla cruzada.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Adicionar componente tabla cruzada en el reporte.	EC 1.1: Seleccionar un dataset.	El diseñador de reportes decide seleccionar un modelo de datos y el sistema muestra los modelos de datos existentes en el sistema.
	EC 1.2: Seleccionar grupo de filas.	El diseñador de reportes decide seleccionar los grupos de filas del componente tabla cruzada y el sistema muestra los atributos del modelo de datos seleccionado.
	EC 1.2: Seleccionar grupo de columnas.	El diseñador de reportes decide seleccionar los grupos de columnas del componente tabla cruzada y el sistema muestra los atributos del modelo de datos seleccionado.
	EC 1.3: Seleccionar medida.	El diseñador de reportes decide seleccionar un elemento como medida y una función estadística asociada a la misma, el sistema

		muestra los atributos del modelo de datos seleccionado y las funciones estadísticas disponibles.
	EC 1.4 : Configurar diseño.	El diseñador de reportes define un tema y una estructura del componente tabla cruzada, el sistema muestra los temas y las estructuras existentes para el componente tabla cruzada.

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

Tabla 4. Descripción de las variables

No	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Dataset	Combobox	No	Constituye el modelo de datos que permite transferir elementos al componente tabla cruzada.
2	Grupo de fila	Combobox	No	Son los encabezados (rowHeader) que conforman las filas del componente tabla cruzada.

3	Grupo de columna	Combobox	No	Son los encabezados (columnHeader) que conforman las columnas del componente tabla cruzada.
4	Medida	Combobox	No	Representa el valor de las celdas interiores del componente tabla cruzada, de acuerdo a una función estadística.
5	Función estadística	Combobox	No	Permite realizar un análisis estadístico del atributo medida.
6	Color	Combobox	No	Es el color que representa las celdas del componente tabla cruzada.

7	Tono	Combobox	No	Variación que puede adquirir un color predefinido del componente tabla cruzada.
8	Grupo de filas totales	CheckBox	No	Es el valor total de las filas que conforman componente tabla cruzada.
9	Grupo de columnas totales	CheckBox	No	Es el valor total de las columnas que conforman componente tabla cruzada.

Esta descripción facilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el componente tabla cruzada, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se registraron los resultados de las pruebas, con el empleo de la técnica de partición de equivalencia.

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las

clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.(Orihuela Sánchez, 2013)

Matriz de datos

Tabla 5. Descripción de las variables del escenario Seleccionar grupo de filas.

Escenario	(Enumeradas según descripción de la variable)						Descripción	Respuesta del sistema	Resultado de la prueba	Flujo central
	1	2	3	4	5	6				
EC 1.2 Seleccionar grupo de filas correctamente	/	V	/	/	/	/	En este escenario se seleccionan los campos que conformarán los encabezados por fila en el componente tabla	El sistema muestra todos los atributos existentes del modelo de datos seleccionado	Satisfactorio	1.Módulo diseñador de reportes/Componente tabla 2.Se selecciona los grupos de fila. 3.El sistema visualiza los elementos seleccionados.

3.5. Resultado de pruebas

Caja negra

Una vez realizadas las pruebas de Caja Negra a través de los casos de prueba asociados a cada CUS, se demostró el correcto funcionamiento del componente tabla cruzada, así como la correcta validación de los campos. En el proceso de pruebas se detectaron 6 No Conformidades, las cuales fueron corregidas y gestionadas correctamente en cada iteración.

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

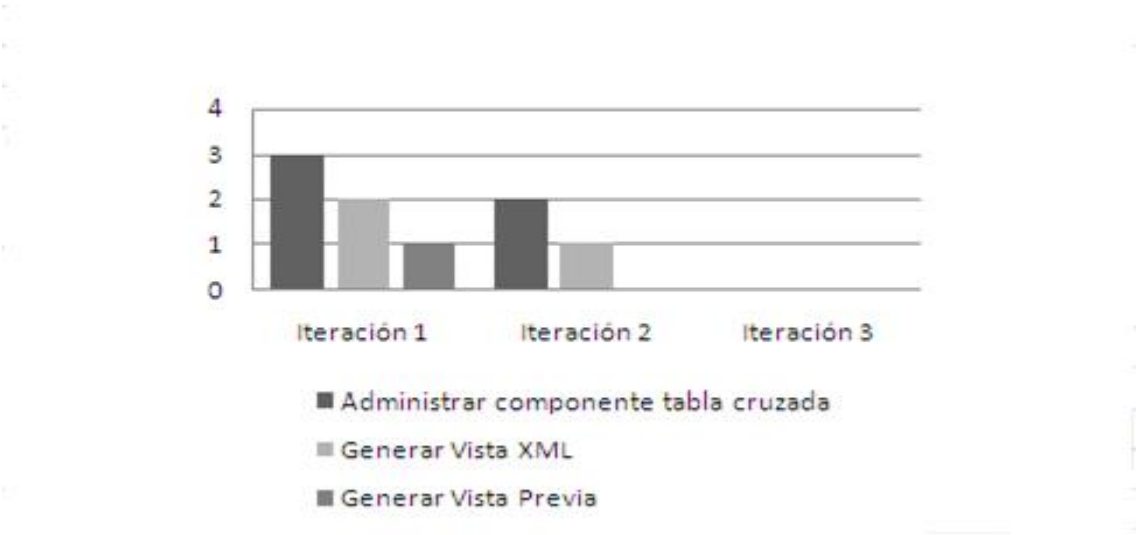


Figura 8. Resumen de las no conformidades.

Prueba de integración

Para la realización de dicha prueba se utilizó la integración descendente en forma de primero en profundidad, construyendo inicialmente la clase principal (WindowWizard) y posteriormente incorporando las clases subordinados a la misma (Panel 1, Panel 2, Panel 3, Panel 4 y Panel 5) con el objetivo de configurar el componente tabla cruzada.

Una vez integrado el componente tabla cruzada al Generador Dinámico de Reportes v2.0, se pudo apreciar que funciona correctamente. Cuando el usuario va a adicionar un componente tabla cruzada, el sistema visualiza la interfaz de configuración asociada al mismo, permitiendo seleccionar los elementos necesarios para generar el componente en el reporte de forma tal que continúe el flujo normal de actividades. Además el usuario después de haber adicionado el componente tabla cruzada puede realizar todas las operaciones siguientes como modificar las propiedades del mismo, transformar su tamaño, cambiar su posición o simplemente eliminarlo del reporte en cuestión; así como acceder a la vista XML y vista previa del módulo Diseñador de Reportes para observar el código en formato XML y una panorámica del mismo.

Conclusiones parciales

En el desarrollo del presente capítulo se representó la estructura del componente tabla cruzada en el diagrama de componentes, señalando la organización y las dependencias que existen entre ellos. También se definieron los estándares de codificación que fueron utilizados, destacando el uso del estándar camelCase y la utilización de los comentarios para explicar el flujo del código y el propósito de las funciones implementadas. Además se definieron como niveles de pruebas el nivel de desarrollador y de integración, utilizando las pruebas funcionales a través del método de caja negra mediante la técnica de partición de equivalencia para la detección y corrección de las no conformidades existentes, así como las pruebas de integración mediante la estrategia de integración incremental descendente para la configuración del componente tabla cruzada.

CONCLUSIONES GENERALES

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

- Se analizaron los conceptos teóricos que sustentaron la investigación, para el desarrollo del componente tabla cruzada, lo que permitió la selección de la metodología, las herramientas y tecnologías que se aplicaron durante la implementación del mismo.
- A partir de las funcionalidades descritas se realizó el análisis y el modelo de diseño del componente tabla cruzada para el Generador Dinámico de Reportes v2.0.
- Se realizó la implementación del componente propuesto de acuerdo con la especificación de los requerimientos y la realización del diseño.
- Se aplicaron pruebas funcionales de tipo caja negra, descritas en los casos de pruebas realizados y las pruebas de integración, validando con estas, el correcto funcionamiento e incorporación del componente tabla cruzada al Generador Dinámico de Reportes v2.0 .

RECOMENDACIONES

Al concluir este trabajo se recomienda:

- Extender el componente tabla cruzada para que se habiliten las funcionalidades avanzadas del mismo.
- Mejorar la usabilidad de la ventana de configuración inicial del componente tabla cruzada.
- Desarrollar el componte tabla para la representación de datos lineales dado que el componente desarrollado solo soporta datos cruzados.

REFERENCIAS BIBLIOGRÁFICAS

ALEGS, L. 2008. Definición de Servidor de aplicaciones. [en línea]. [Consulta: 28 mayo 2015]. Disponible en: http://www.alegsa.com.ar/Dic/servidor de aplicaciones.php.

BERNAL ROJAS, J.A., 2013. *Intérprete SQL para el módulo Diseñador de Consultas del GDR*. 2013. S.l.: s.n.

BLANCO CRIADO, A. 2009. Informes Crosstab con iReport. [en línea]. [Consulta: 26 mayo 2015]. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=iReportCrosstab.

BRITO RODRIGUEZ, J.C., 2012. *Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0*. julio 2012. S.l.: s.n.

CARRILLO, F. 2011. Pruebas del Software: Niveles de Prueba del Software. [en línea]. [Consulta: 10 junio 2015]. Disponible en: http://ingenieriadepruebasdelsoftware.blogspot.com/2011/01/niveles-de-prueba-del-software.html.

DEFINICIÓN ABC 2008. Definición de Tabla » Concepto en Definición ABC. [en línea]. [Consulta: 26 mayo 2015]. Disponible en: http://www.definicionabc.com/tecnologia/tabla.php.

DEFINICION.DE 2008. Definición de tabla - Qué es, Significado y Concepto. [en línea]. [Consulta: 26 mayo 2015]. Disponible en: http://definicion.de/tabla/.

ENNIS BOULY, Y. 2013. *Componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g* [en línea]. S.l.: s.n. [Consulta: 10 junio 2015]. Disponible en: zotero://attachment/110/.

ETHEK, 2010. *Codigo fuente* [en línea]. 2010. S.l.: s.n. Disponible en: http://www.ethek.com/definicion-del-codigo-fuente/.

EXCELTOTAL 2010. Para qué sirven las tablas dinámicas - Excel Total. [en línea]. [Consulta: 9 junio 2015]. Disponible en: https://exceltotal.com/para-que-sirven-las-tablas-dinamicas/.

GARCÍA QUINTELA, S., 2013. *Extensión del NetBeans IDE para el diseño de Interfaces Gráficas de Usuario con Ext JS*. 2013. S.l.: s.n.

GARCÍA SUÁREZ DEL VILLAR, C., 2012. *Herramienta para la migración de la base de datos del Generador Dinámico de Reportes (GDR) V1.7 a V2.0*. 2012. S.l.: s.n.

GAZQUEZ MARTINEZ, O., 2011. *Diseño e Implementación del módulo Diseñador de Consultas del Generador Dinámico de Reportes*. 2011. S.l.: s.n.

HERNANDEZ CARVAJAL, J.J., 2012. *Desarrollo del módulo de Administrador de Reportes del Generador Dinámico de Reportes*. 2012. S.l.: s.n.

REFERENCIAS BIBLIOGRÁFICAS

HERNANDEZ HERNANDEZ, Y., 2009. *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos*. 2009. S.l.: s.n.

IBM 2007. Rational Software: Architecture and Test. ,

INC, S., 2005. *Tabulación Cruzada*. 2005. S.l.: s.n.

JSON.ORG 2010. JSON. [en línea]. [Consulta: 26 mayo 2015]. Disponible en: http://www.json.org/.

LARMAN, C., 2006. *UML y Patrones*. 2006. S.l.: s.n.

LOBO, A.R., 2009. *Arquitectura de Software para el Sistema Integrado de Gestión Estadística*. 2009. S.l.: s.n.

MARTÍNEZ, A.G. 2011. Tablas Cruzadas En Spss: Tablas Cruzadas O Contingencia En Spss. [en línea]. [Consulta: 9 junio 2015]. Disponible en: http://www.spssfree.com/curso-de-spss/tablas-de-spss/tablas-cruzadas-con-variables-categoricas.html.

MEDINILLA SANTO, Y. de la C. aridad y ÁLVAREZ ROJAS, M. 2013. *Herramienta para la migración de la base de datos del Generador Dinámico de Reportes (GDR) V1.8 a V2.0*. S.l.: s.n.

OMG UNIFIED MODELING LANGUAGE SPECIFICATION, 2010. CiteSeerX — OMG Unified Modeling Language Specification. [en línea]. [Consulta: 26 mayo 2015]. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.143.6213.

O'REILLY 2010. XML.com. [en línea]. [Consulta: 26 mayo 2015]. Disponible en: http://www.xml.com/.

ORIHUELA SÁNCHEZ, S., 2013. *Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes*. junio 2013. S.l.: s.n.

PRESSMAN, R.S. 2010. *Software Engineering : A Practitioner's Approach*. S.l.: s.n. ISBN 978-0-07-337 597 -7.

RODRIGUEZ DURÁN, A., 2011. *IMPACTO DEL GENERADOR DINÁMICO DE REPORTES EN LOS SISTEMAS DE GESTIÓN DE INFORMACIÓN*. 2011. S.l.: s.n.

VISUAL PARADIGM, 2011. *Visual Paradigm for UML* [en línea]. 2011. S.l.: s.n. Disponible en: http://www.visual- paradigm.com/product/vpuml/.

Bibliografías

BERNAL ROJAS, J.A., 2013.*Intérprete SQL para el módulo Diseñador de Consultas del GDR*. 2013. S.l.: s.n.

BERZAL, F., 2009.*Relaciones entre clases*. S.l.: s.n.

BLANCO CRIADO, A. 2009. Informes Crosstab con iReport. [En línea]. [Consulta: 4 mayo 2015]. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=iReportCrosstab.

BOULY, Y.E. 2013.*Componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g*. S.l.: s.n.

BRITO RODRIGUEZ, J.C., 2012.*Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0*. Julio 2012. S.l.: s.n.

COMPANY, 2011. JasperReports Report Groups. [En línea]. [Consulta: 2 mayo 2015]. Disponible en: http://www.tutorialspoint.com/jasper_reports/jasper_report_groups.htm.

DAVID PLAT, 2012. Create Jasper Report Table component with Group By. .

DEFINICIÓN ABC, 2008. Definición de Tabla » Concepto en Definición ABC. [En línea]. [Consulta: 2 mayo 2015]. Disponible en: http://www.definicionabc.com/tecnologia/tabla.php.

DEFINICION.DE, 2010. Definición de tabla - Qué es, Significado y Concepto. [En línea]. [Consulta: 2 mayo 2015]. Disponible en: http://definicion.de/tabla/.

DÍAZ-HEREDERO, R.A. 2013. Trabajar con tablas en JasperReport. [En línea]. [Consulta: 3 abril 2015]. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=JasperTableComponent.

ETHEK, 2010. *Código fuente* [en línea]. 2010. S.l.: s.n. Disponible en: http://www.ethek.com/definicion-del-codigo-fuente/.

FRANCO NAVARRO., J.A., 2014.*UML en acción. Modelando Aplicaciones Web* [en línea]. 2014. S.l.: s.n.

GALERA 2005. Usando o componente table do Ireport. .

GARCÍA QUINTELA, S., 2013.*Extensión del NetBeans IDE para el diseño de Interfaces Gráficas de Usuario con Ext JS*. 2013. S.l.: s.n.

GARCÍA SUÁREZ DEL VILLAR, C., 2012.*Herramienta para la migración de la base de datos del Generador Dinámico de Reportes (GDR) V1.7 a V2.0*. 2012. S.l.: s.n.

GAZQUEZ MARTINEZ, O., 2011.*Diseño e Implementación del módulo Diseñador de Consultas del*

BIBLIOGRAFÍAS

Generador Dinámico de Reportes. 2011. S.l.: s.n.

GIMSON, L., 2012.*Metodologías ágiles y desarrollo basado en conocimiento*. Junio 2012. S.l.: s.n.

HERNANDEZ CARVAJAL, J.J., 2012.*Desarrollo del módulo de Administrador de Reportes del Generador Dinámico de Reportes*. 2012. S.l.: s.n.

HERNANDEZ HERNANDEZ, Y., 2009.*Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos*. 2009. S.l.: s.n.

HGUPTA 2012. Ireport group by in table component (Open Source Projects forum at JavaRanch). [En línea]. [Consulta: 2 mayo 2015]. Disponible en: http://www.coderanch.com/t/565914/open-source/ireport-group-table-component.

IBM 2007. Rational Software: Architecture and Test.

JASPERT COMMUNITY. JasperReports 6.1.0 - Table Sample. [En línea]. [Consulta: 2 mayo 2015]. Disponible en: http://jasperreports.sourceforge.net/sample.reference/table/index.html.

JSON.ORG, 2010. JSON. [En línea]. [Consulta: 23 abril 2015]. Disponible en: http://www.json.org/.

LARMAN, C., 2006.*UML y Patrones*. 2006. S.l.: s.n.

LOBO, A.R., 2009.*Arquitectura de Software para el Sistema Integrado de Gestión Estadística*. 2009. S.l.: s.n.

MICROSOFT, 2014. Cómo utilizar el componente tabla dinámica de Office Web Components con VB. [En línea]. [Consulta: 26 mayo 2015]. Disponible en: https://support.microsoft.com/es-es/kb/235542/es.

OMG UNIFIED MODELING LANGUAGE SPECIFICATION, 2010. CiteSeerX — OMG Unified Modeling Language Specification. [En línea]. [Consulta: 24 abril 2015]. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.143.6213.

O'REILLY, 2010. XML.com. [en línea]. [Consulta: 26 abril 2015]. Disponible en: http://www.xml.com/.

ORIHUELA SÁNCHEZ, S., 2013. *Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes*. Junio 2013. S.l.: s.n.

PRESSMAN, R.S. 2010.*Software Engineering : A Practitioner's Approach*. S.l.: s.n. ISBN 978-0-07-337 597 -7.

RODRIGUEZ DURÁN, A., 2011.*IMPACTO DEL GENERADOR DINÁMICO DE REPORTES EN LOS SISTEMAS DE GESTIÓN DE INFORMACIÓN*. 2011. S.l.: s.n.

SADAKAR, 2013. Jasper reports - Table component without main Query in iReport. [En línea]. [Consulta: 2 mayo 2015]. Disponible en: http://stackoverflow.com/questions/19265634/table-component-with-out-main-

BIBLIOGRAFÍAS

query-in-ireport.

SOUSA, D. 2010. Usando o componente Table do iReport. [En línea]. [Consulta: 26 abril 2015]. Disponible en: http://www.danielsousa.com.br/wp/2010/06/15/usando-o-componente-table-do-ireport/.

TIBCO SOFTWARE. Getting Started with the Table Component. [En línea]. [Consulta: 2 mayo 2015 a]. Disponible en: http://community.jaspersoft.com/wiki/getting-started-table-component.

TIBCO SOFTWARE. Tables and Crosstabs in Jaspersoft Studio. [En línea]. [Consulta: 2 mayo 2015 b]. Disponible en: http://community.jaspersoft.com/wiki/tables-and-crosstabs-jaspersoft-studio.

Uso de UML en aplicaciones Web: páginas y relaciones. [En línea] . [Consulta: 2 mayo 2015]. Disponible en: http://www.milestone.com.mx/articulos/uso_de_uml_en_aplicaciones_web.htm.

VISUAL PARADIGM, 2011.*Visual Paradigm for UML* [en línea]. 2011. S.l.: s.n. Disponible en: http://www.visual- paradigm.com/product/vpuml/.

ZYLK.NET, 2013.*GENERACIÓN DE INFORMES MEDIANTE HERRAMIENTAS OPENSOURCE* [en línea]. 2013. S.l.: s.n. Disponible en: www.zylk.net.

ALEGS, L. 2008.Definición de Servidor de aplicaciones. [En línea]. [Consulta: 22 abril 2015]. Disponible en: http://www.alegsa.com.ar/Dic/servidor de aplicaciones.php.

ANEXOS

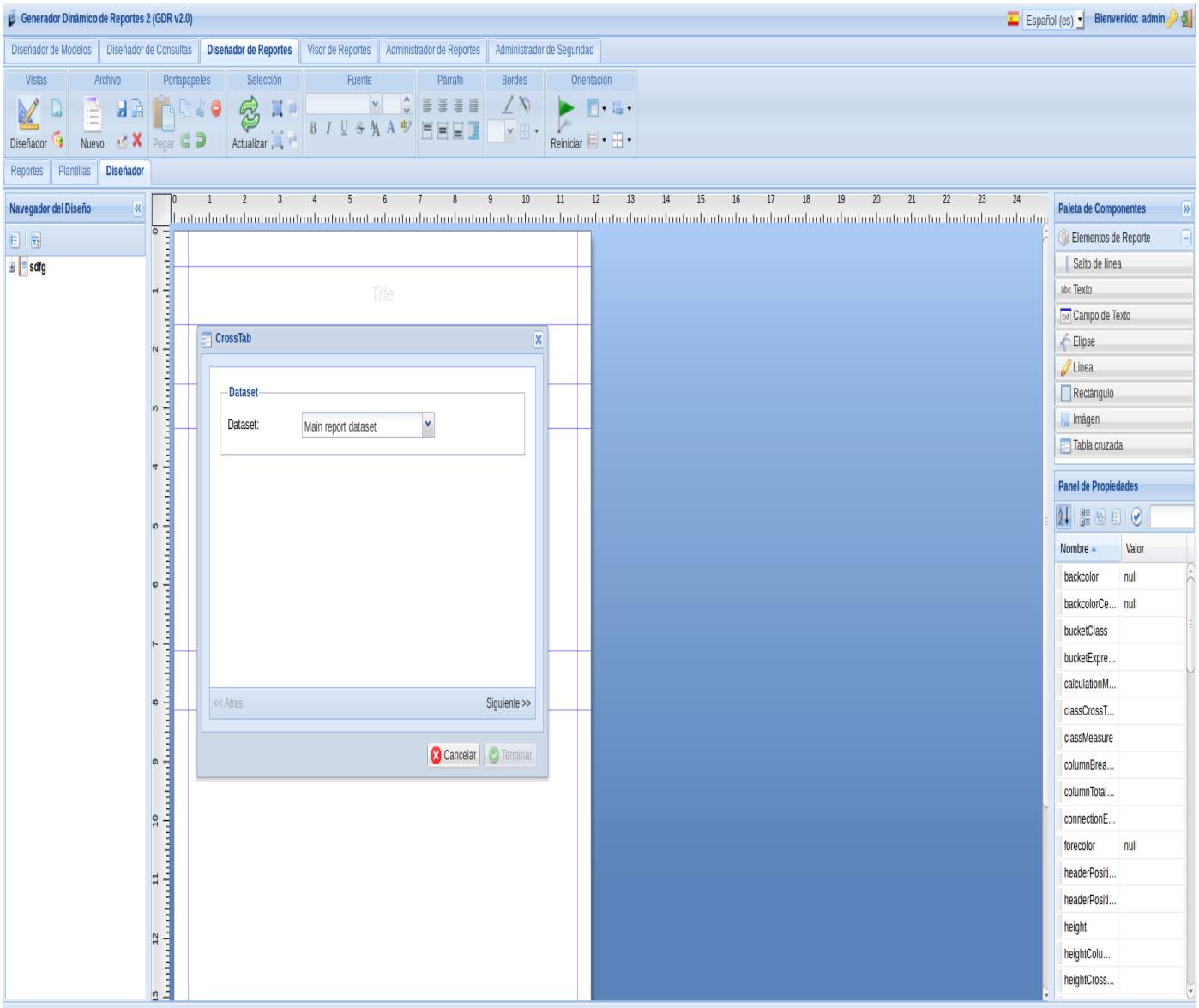


Figura 9. Componente tabla cruzada integrado al sistema GDR v2.0.

GLOSARIO DE TÉRMINOS

AJAX: Aplicación interactiva para páginas web.

API: Interfaz de Programación de Aplicaciones.

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

CASE: Ingeniería de Software Asistida por Computación.

CSV: Tipo de documento en formato abierto.

DATEC: Centro de Tecnologías de Gestión de Datos.

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

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

Frameworks: Marco de trabajo.

GPL: Licencias Open Source.

GRASP: General Responsability Assignment Software Patterns (Patrones de Software para la asignación General de Responsabilidades)

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

HTML: Lenguaje de Marcado de Hipertexto

IDE: Entorno Desarrollo Integrado.

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

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

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

JSON: Notación de Objetos de JavaScript.

MVC: Modelo – Vista – Controlador.

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

OpenUP: Proceso Unificado Abierto.

PDF: Formato de Documento Portátil.

GLOSARIO DE TÉRMINOS

PHP: Hipertexto Pre-Procesado.

RUP: Rational Unified Process.

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

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje de Modelado Unificado.

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

XML: Lenguaje de Marcas Extensible.

XP: eXtreme Programming (Programación Extrema).