Universidad de las Ciencias Informáticas

FACULTAD 6

Sistema Integrado de Gestión Estadística (SIGE).

Título: Aplicación informática para la recolección de los datos primarios de los centros informantes del Sistema Integrado de Gestión Estadística.

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

Autor:

Koisam Rodriguez Villamil

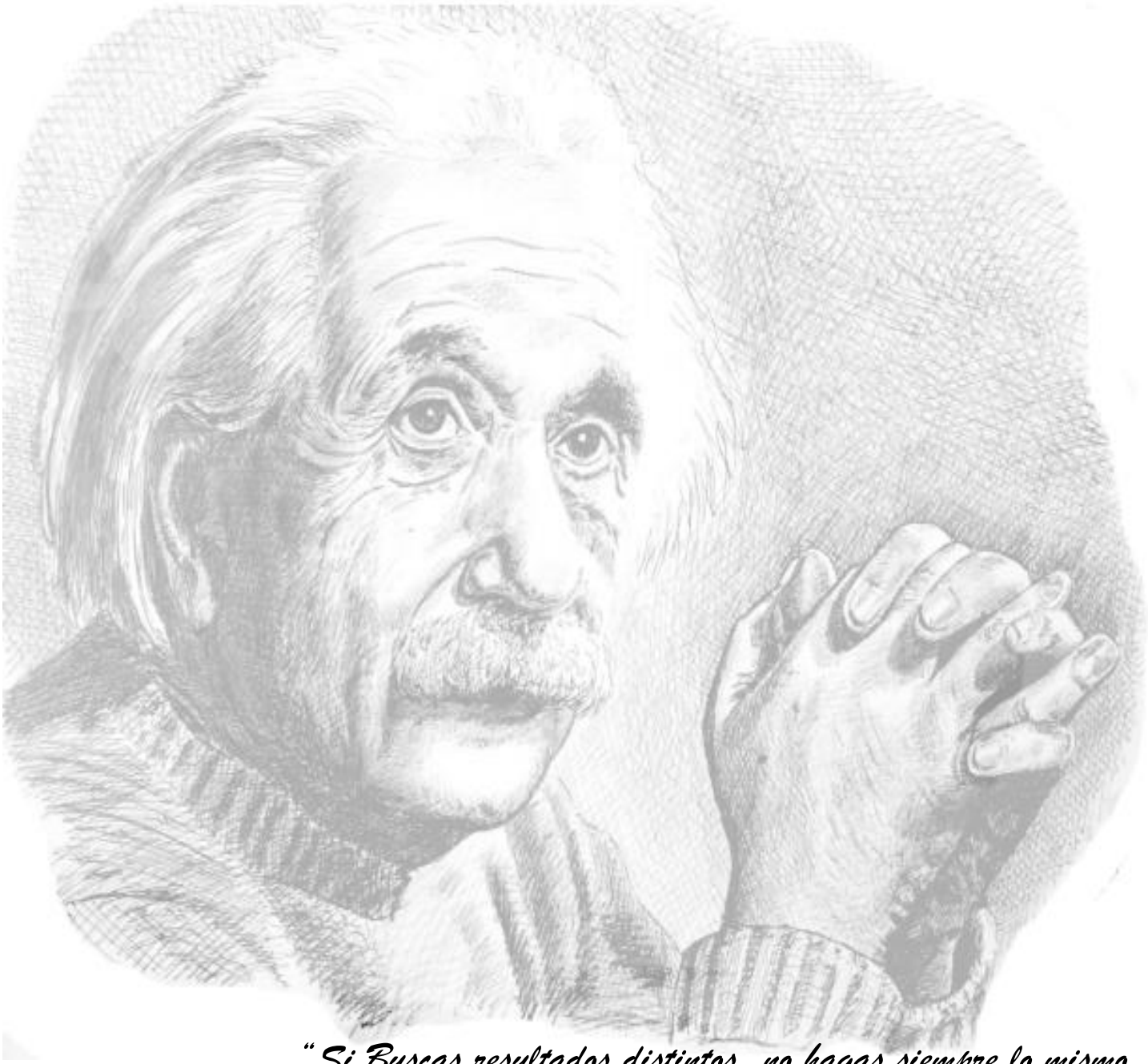
Tutores:

MsC. Reynaldo Alvarez Luna

Ing. Adrián Rosales Cruz

Ing. Yalina López Brull

Junio de 2014



"Si Buscas resultados distintos, no hagas siempre lo mismo."

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo 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 _____.

Koisam Rodríguez Villamil	**Ing. Adrián Rosales Cruz**
Firma del Autor.	Firma del Tutor.
MsC. Reynaldo Álvarez Luna	**Ing. Yalina López Brull**
Firma del Tutor.	Firma del Tutor.

DATOS DE CONTACTO

Nombre: Koisam

Apellidos: Rodríguez Villamil

Correo: kvillamil@estudiantes.uci.cu

Nombre: Reynaldo

Apellidos: Álvarez Luna

Ingeniero en Ciencias Informáticas, UCI 2008

Máster en Informática Aplicada, UCI 2011

Correo: rluna@uci.cu

Nombre: Yalina

Apellidos: López Brull

Título universitario: Ingeniero en Ciencias Informáticas

Año de graduado: 2013

Correo: yolopez@uci.cu

Nombre: Adrián

Apellidos: Rosales Cruz

Título universitario: Ingeniero en Ciencias Informáticas

Año de graduado: 2012

Correo: adrosales@uci.cu

AGRADECIMIENTO

En primer lugar me gustaría agradecerles especialmente a mi familia y a todas esas personas que de una forma u otra me ayudaron a ponerle fin a esta etapa de estudiante.

A mi padre: Que ha sido el tutor mío de años que siempre ha estado guiándome en mi vida para hoy forjarme como ingeniero. Después de tantos fines de semana de estudio, con su frasecita mañanera prepara papel y lápiz que voy a dictar sentencia y a partir de ahí 8 horas seguidas de estudio intenso, pero gracias a eso le agradezco con toda mi vida y este título de Ingeniero en Ciencias Informáticas es especialmente para él y para mi madre por supuesto.

A mi madre: Que ha sido la luz que me ha alumbrado todos mis caminos y gracias a su preocupación y su amor incondicional puedo estar aquí hoy diciendo estas palabras, que no me alcanzarían ni 80 páginas para agradecerle todo lo que ha hecho por mí.

A mi familia: A mi tía favorita Aurora que a pesar de sus locuras siempre se ha preocupado por mí y estado al tanto en todos los momentos de mi vida, también agradecerles a mi primo Jeovany, a mi tía Sarahi, a mi prima Claudia que de una forma u otra han aportado su granito de arena.

A mis seres queridos: A mi tío Noel, a mi abuela Aurora, a mi abuela Marta, a mi abuelo Alberto que hoy por hoy no están presente en este mundo pero sé que desde el cielo me están observando y cuidando de mí.

A la gente del grupo: A esos amigos y compañeros que me han ayudado cada vez que he lo necesitado, gracias por explicarme esa duda cuando la tenía que eran más de una por cierto.

A mis amigos: A un grupo especial de personas que siempre los recordaré, por tantos momentos que pasamos y nos divertimos: gracias a Orlando, Adler, Maykol, Carlos, Jomba por ser algo más que conocidos.

A mis hermanos: Uno de los agradecimientos más importantes para mí ya que son más que amigos, son hermanos, son parte de mi familia, que pueden estar seguro que siempre tendrán a una persona en quien confiar aquí. Yoel tu que me apoyaste cuando más estuve con la soga al cuello, siempre alentándome y dándome fuerzas, gracias mi hermano, Alain tu que desde primer año compartiste conmigo en todo momento, fiestas, casa, cuarto, cama, gracias mi hermano por haberte conocido, Adrián el emito, a lo mejor no he compartido tanto contigo pero por tu carácter, comprensión y desinterés te has convertido en

un hermano para mí y sabes que tienes una casa en la Habana para cuando quieras ir y por último agradecerle a un verdadero amigo, hermano y no sé qué más a Luis Alejandro gracias colega porque gracias a ti hoy por hoy he podido regalarle este título a mis padres, y aunque ya te vas sabes que aquí dejas a un hermano, te quiero *brother*.

A mis tutores: Agradecerle a mi tutora y súper amiga Yali que no ha dejado de estar atrás de mí en ningún momento para que pueda acabar con esta tragedia, gracias mi amiga. A mi tutor Adrián que el bien sabe lo agradecido que estoy con él, que más que ser un tutor fue un amigo, un consejero, un hermano, gracias colega por orientarme en todo momento y a mi tutor Reynaldo que a pesar de tener pocos encuentros por mi culpa claro, se portó muy elegante y le tengo mucho aprecio y admiración.

A mi oponente querida: Gracias Yanet porque de no ser por ti tampoco estuviese aquí, por esta lucha que te di estos últimos días que aunque me descuartizaste el documento dos días antes de entregarlo te quiero igual.

A los profesores: Agradecerle a mi tribunal de tesis que ellos dijeron la última palabra, a aquellos profesores que a pesar de ser como soy han sabido tolerarme y ayudarme a pesar de todo y darme consejos que aunque no lo parecía si los retenía, gracias al profesor Jorge Luis que es digno de respeto y admiración de mi parte y por confiar en mí hasta el último momento.

A Yuliet Pérez Ananias: Palabras no tengo para agradecerte todo lo que has hecho por mí en estos años y sobre todo este último año de mi carrera. Marcaste una línea en mi vida, gracias por comprenderme, aguantarme, ayudarme a salir de grandes apuros aquí en la UCI estando siempre junto a mí, a guardar tantos secretos que solo tú y yo conocemos y apoyarme en todo momento, además de haber sido una increíble pareja, esa estupenda mujer, amiga, confidente, tutora, muchas gracias por forjarme como ingeniero y ojala algún día haya perdón en tu corazón hacia mí.

A Wendy: Que te haya dejado de ultima no quiere decir que seas menos importante, gracias por tu compañía y por alegrarme cada momento de mi vida.

DEDICATORIA

A mis padres: Por todo el amor que me han brindado y tu entrega incondicional hacia mí, y todos esos gustazos que me han propiciado…

A mi familia: Por haberme apoyado en toda esta carrera y decirme siempre que si se puede…

A todas las personas: Que a lo largo de mi carrera me han apoyado y ayudado a ser lo que soy hoy por hoy…

RESUMEN

La Universidad de las Ciencias Informáticas cuenta con varios centros de desarrollo de software, entre ellos el Centro de Tecnologías y Gestión de Datos. En dicho centro se desarrolló el Sistema Integrado de Gestión Estadística (SIGE) donde su principal cliente es la Oficina Nacional de Estadística e Información, la cual presenta una estructura organizativa a lo largo del país dividido por diferentes niveles de dirección. SIGE se encuentra desplegado actualmente a todos los niveles del país, dígase municipales, provinciales y nacional, el cual se sustenta de los datos primarios obtenidos de los Centros Informantes en formato Excel. Sin embargo dicho proceso se torna engorroso debido a que el sistema solo permite la captura y actualización de la información a través de diferentes campos que componen una fila del Excel, los cuales en ocasiones contienen un gran cúmulo de información, trayendo como consecuencias que la inserción de los datos sea propensa a errores.

El presente trabajo de diploma tiene como objetivo desarrollar una aplicación informática que permita la gestión de datos primarios a partir de cargas iniciales. Para guiar el desarrollo de la investigación se usó como guía la metodología OpenUp, se determinaron los patrones que organizarán la arquitectura y diseño de la aplicación informática y se realizaron pruebas a nivel de unidad, de sistema y de desarrollador para comprobar el correcto funcionamiento de la aplicación informática.

Como resultado se obtuvo un sistema que le permitirá a SIGE informatizar el proceso de captura y actualización de los datos primarios obtenidos de los Centros Informantes, logrando de esta forma que el proceso y validez de los datos insertados respectivamente se realice de manera absoluta.

Palabras Clave: sistema integrado de gestión estadística, datos primarios.

ABSTRACT

University of Informatics Sciences has several software development centers, including the Center for Technology and Data Management. At the center of the Integrated Management System Statistics (EMIS) where his main client is the National Office of Statistics and Information, which presents an organizational structure throughout the country divided by different levels of management was developed. EMIS is currently deployed at all levels of the country, say municipal, provincial and national, which is based on primary data obtained from informants Centers in Excel format. However this process becomes troublesome because the system only allows the capture and updating of information across different fields that make up a row in Excel, which sometimes contain a wealth of information, bringing as consequences insertion data is error prone. This diploma work is to develop a software application that allows the management of primary data from initial loads. To guide the development of research was used to guide the OpenUp methodology, the architectural pattern designs and patterns to be used for the correct implementation of the module and tests were performed at the unit level and system to verify proper operation was determined computer application. The result is a system that will allow EMIS computerize the process of capturing and updating of primary data from reporting establishments thus achieving the process and validity of the data inserted respectively make absolutely was obtained.

Keywords: *statistics integrated management system, primary data.*

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) han propiciado el avance de cada una de las ramas de la sociedad, la economía, la ciencia y la técnica causando impacto en los procesos de gestión de la información que se llevan a cabo actualmente (1).

Cuba no está exenta del uso y beneficios de las TIC, por lo que informatizar el país y desarrollar la industria de *software* para contribuir al desarrollo económico y social del mismo, han sido los pilares fundamentales en las últimas décadas. Una muestra lo constituye la creación de la Universidad de las Ciencias Informáticas (UCI), que plantea entre sus objetivos la informatización del país para elevar la calidad de vida, así como desarrollar la industria de *software* para contribuir al progreso económico y social del país.

La UCI posee convenios de colaboración con diferentes empresas u organizaciones siendo una de ellas la Oficina Nacional de Estadísticas e Información (ONEI), que tiene como misión: integrar, organizar y dirigir el trabajo estadístico estatal del país de manera eficiente. Mediante su Sistema Estadístico Nacional (SEN) se gestionan los procesos asociados a esta, permitiendo *"…brindar la información requerida para el control del plan de la economía, el análisis del comportamiento de la economía y las demás necesidades del país*" (2).

Para facilitar su trabajo y obtener datos de forma más precisa, la ONEI ha priorizado la introducción de técnicas de almacenamiento y procesamiento de datos en formato digital y con esto el desarrollo de una aplicación informática de soporte al marco metodológico del trabajo estadístico (3). La ONEI ha recurrido a los servicios que ofrece la UCI a través del Centro de Tecnologías y Gestión de Datos (DATEC), el cual tiene como objetivo *"…crear bienes y servicios informáticos relacionados con la gestión de datos, área del conocimiento que agrupa tanto a los sistemas de información, como a los denominados sistemas de inteligencia empresarial o de negocios, cuyo propósito fundamental es apoyar el proceso de toma de decisiones"*.

Perteneciente al centro DATEC se encuentra el Departamento de Integración de Soluciones, que desarrolla como uno de sus productos el Sistema Integrado de Gestión Estadística (SIGE). SIGE se encuentra actualmente desplegado en la ONEI y tiene como objetivo principal la informatización de procesos de captura de información estadística.

Entre los módulos que presenta SIGE se encuentra el módulo de Gestión de Configuración, el cual tiene como objetivo principal la captura y la actualización de la información de los datos primarios. Se define como datos primarios en un conjunto de características que responden a un concepto del negocio que establece la ONEI, tales como: Centros Informantes, Clasificadores, Registros, Indicadores y Aspectos.

El proceso, de captura y actualización de datos primarios, se realiza principalmente durante las cargas iniciales de la aplicación, que es la etapa donde la entidad entrega documentos digitales en formato Excel. Una vez recopilados todos los documentos digitales, un usuario de la aplicación deberá insertar la información mediante el sistema SIGE, a través de la opción “Adicionar”, que puede ser accedida desde las interfaces correspondientes a cada uno de los datos primarios. Sin embargo esta funcionalidad solo permite la inserción al sistema de los campos correspondientes a cada fila del documento Excel y en ocasiones estos documentos contienen grandes volúmenes de información. Trayendo como consecuencia que la realización de este proceso sea compleja y lenta, propiciando largas jornadas de trabajo e inserción de información propensa a errores; afectando así la ejecución del proceso y validez de los datos insertados respectivamente.

Por lo antes se identifica como **problema de la investigación** *el flujo actual de recolección de datos primarios demora y complejiza la actualización de la información en el Sistema Integrado de Gestión Estadística.*

El **objeto de estudio** se centra en *la actualización de datos en los Sistemas de Información Estadística*, especificándose como **campo de acción** *la actualización de datos en SIGE.*

Para dar solución al problema de la investigación se define como **objetivo general** *desarrollar una aplicación informática para la recolección de los datos primarios de los Centros Informantes que mejore la actualización de datos en SIGE.*

Los **objetivos específicos** definidos en la investigación son:

- Elaborar el marco teórico conceptual relacionado con los aspectos teóricos que sustentan la investigación.
- Identificar las funcionalidades de la aplicación informática.
- Realizar el diseño de la aplicación informática a partir de las funcionalidades identificadas.
- Implementar la aplicación informática.
- Realizar pruebas para verificar el correcto funcionamiento de la aplicación informática.

Para dar solución a la problemática, se plantearon algunas interrogantes con el fin de encaminar la investigación a resultados satisfactorios:

- ¿Cómo actualizar la gestión de los datos primarios en SIGE mediante el desarrollo de una aplicación informática que permita la inserción de los mismos de forma absoluta?
- ¿Cómo verificar el correcto funcionamiento de la actualización de los datos primarios en la aplicación informática desarrollada?

Para dar cumplimiento a los objetivos e interrogantes expuestos se proponen las siguientes ***tareas de la investigación:***

1. Describir el proceso de actualización de datos primarios en SIGE.
2. Definición de la metodología, herramientas y lenguaje de programación adecuado a utilizar en la implementación de la aplicación.
3. Confección del Modelo de Dominio del proceso de actualización de datos para SIGE.
4. Definición de los requisitos funcionales de la aplicación informática.
5. Realización del Modelo de Casos de Uso de la aplicación informática.
6. Descripción de los Casos de Uso de la aplicación informática.
7. Elaboración del Diagrama de Clases de Diseño de la aplicación informática.
8. Elaboración del Modelo de Datos de la aplicación informática.
9. Implementación de la aplicación informática.
10. Definición y realización de pruebas para comprobar el correcto funcionamiento del sistema.
11. Definición y realización de pruebas para verificar la correcta actualización de datos primarios en el sistema.

Métodos científicos

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. Se clasifican en teóricos y empíricos, los cuales están dialécticamente relacionados.

Métodos teóricos

En la investigación se utiliza el método teórico **analítico-sintético** para a partir del análisis del proceso de inserción y actualización de datos, en el cual intervienen los Centros Informantes, se tiene como principal objetivo lograr identificar y describir los datos primarios que intervienen en dicho proceso.

La modelación es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. Se aplica en el trabajo mediante la modelación o diseño de los diagramas u artefactos generados por la metodología aplicada para el desarrollo de la aplicación

Métodos empíricos

La entrevista individual como método empírico utilizado permite obtener información sobre la presente investigación, donde se entrevistó al cliente en varias ocasiones. Previo a las entrevistas realizadas se elaboró una guía sobre los temas a tratar. En este sentido se lograron definir los requisitos funcionales del sistema con la mejor precisión posible, así como capturar elementos restrictivos para el diseño o de preferencia del cliente, recogidos en los requisitos no funcionales.

La presente investigación está estructurada en 4 capítulos:

Capítulo 1: "Fundamentación teórica de la solución"

En este capítulo se realiza una explicación del problema y se exponen los conceptos fundamentales que fueron necesarios adquirir para una mejor comprensión de la investigación que se realiza y se describe la justificación de la metodología, lenguaje de programación y tecnologías utilizadas para el desarrollo de la aplicación informática.

Capítulo 2: "Características del sistema"

Se describe el funcionamiento del negocio a través del diagrama del modelo de dominio y elementos de la solución propuesta que se recogen en la definición de los requisitos funcionales. Estos últimos se detallan mediante la descripción de los casos de uso del sistema. En este capítulo se definen además los requisitos no funcionales de la aplicación.

Capítulo 3: "Diseño del sistema"

Ofrece una visión del modelado del sistema, acorde a los requerimientos definidos para la aplicación en el capítulo anterior y los patrones de arquitectura y diseño utilizados en dicha aplicación.

Capítulo 4: “Implementación y pruebas”

En este capítulo se describe el proceso de implementación de la solución. Además se especifican las pruebas realizadas a la aplicación, así como el análisis de los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA SOLUCIÓN

En este capítulo se realiza un análisis y caracterización del tema de investigación, profundizando en la actualización de datos primarios en SIGE. Se definen los conceptos fundamentales relacionados con la investigación, que desde el punto de vista teórico permite una mejor comprensión de la situación problemática planteada. Se selecciona la metodología de desarrollo a utilizar, así como la herramienta de modelado, el lenguaje de programación y las tecnologías empleadas.

1.1 Conceptos Básicos para el Dominio de la Investigación.

Son conceptos que propician un mejor entendimiento de la investigación que se desarrolla, esclareciendo el significado de los términos más utilizados en la misma, para una concepción concreta de lo que se expone.

1.1.1 Datos Primarios de SIGE.

Los datos primarios se definen como la información capturada por SIGE, el cual a través del Módulo de Gestión de Configuración gestiona los conceptos asociados a: registros, clasificadores, clasificaciones, centros informantes, indicadores y aspectos. Por cada concepto se genera un Excel, el cual responde a un conjunto de características equivalentes a los campos (dígase columnas) de una fila.

Centros Informantes

Los Centros Informantes o empresas constituyen el elemento primario del sistema ya que de ellos se reporta la información estadística, son las entidades que le tributan información a la ONEI (4). En el proceso de actualización que se lleva a cabo en SIGE, los Centros Informantes son manipulados como datos que están estructurados por once elementos que lo caracterizan: código, estado, código de registro, nombre descriptivo, dirección geográfica, fecha de creación, provincia y municipio a la que pertenece, detalles, correo y teléfono. Un ejemplo se muestra en la figura 1.

código	estado	cod. registro	nombre descriptivo	direccion	fecha creacion	codote	codome	detalles	correo	telefono
2681419124	1	26801	Fiscalía Municipal de Artemisa	calle 42 no. 3310 e/ 33 y 44	2013-01-30	22	2209	"	"	"

Figura 1. Descripción de la información referente a un Centro Informante.

Indicador

Los indicadores son utilizados para clasificar y definir, de forma más precisa, objetivos e impactos. Constituyen medidas verificables de cambio o resultados, diseñadas para contar con un estándar contra el cual evaluar, estimar o demostrar el progreso con respecto a las metas establecidas (4).

Los indicadores son datos que están estructurados por nueve elementos que lo caracterizan: identificador, clasificación de seguridad, tipo de indicador, código de temática, nombre descriptivo, unidad de medida, periodicidad, nota metodológica y si es sumable o no. Un ejemplo se muestra en la figura 2.

id	*clasificación*	*tipo*	*código*	*nombre*	*unidad*	*periodicidad*	*nota*	*sumable*
2,68E+12	3	1	268040101	Total de juicios celebrados	U	1	Total de juicios celebrados	TRUE

Figura 2. Descripción de la información referente a un Indicador.

Aspecto

Los Aspectos son características específicas de los Indicadores. Están estructurados por cuatro elementos que lo caracterizan constituidos por: identificador, alias, nombre descriptivo y nota metodológica. Un ejemplo se muestra en la figura 3.

identificador	alias	nombre descriptivo	nota metodologica
m268mesact	Mes	Mes Actual	Este aspecto especifica el valor de los indicadores en el mes en el que se procesa la informacion

Figura 3. Descripción de la información referente a un Aspecto.

Registros

Los Registros permiten agrupar a los Centros Informantes de acuerdo a la actividad económica que desempeñan (4). Estos están estructurados por tres elementos que lo caracterizan, constituidos por: código de registro, nombre descriptivo y alias. Un ejemplo de ello se muestra en la figura 4.

código	nombre descriptivo	alias
26801	Registro de Entidades de la Fiscalía General de la República	rfiscalia

Figura 4. Descripción de la información referente a un Registro.

Clasificadores

Los clasificadores son aquellos instrumentos que sólo asignan un código a elementos definidos, como es el caso del codificador de la División Política Administrativa (DPA) que identifica un código a las provincias y municipios de la DPA. Sin embargo, un clasificador al establecer agrupaciones y niveles jerárquicos, define el alcance de cada elemento. Los clasificadores pueden ser un clasificador de productos, de actividades económicas o de educación (5). En el contexto de la investigación los clasificadores están estructurados por siete elementos que lo caracterizan constituidos por: clasificador, alias, nombre descriptivo, dominio, código del clasificador, código de clasificación y nombre descriptivo de clasificación. Un ejemplo se muestra en la figura 5.

clasificador	alias	nombre descriptivo	dominio	código	código clasificacion	nombre descriptivo
1	dpa	Clasificador de la División Político Administrativa	4	1	2100	Pinar del Río

Figura 5. Descripción de la información referente a un Clasificador.

Clasificaciones

Las Clasificaciones son conceptos más específicos contenidos dentro de los Clasificadores. En esencia son los valores que puede tomar un Clasificador en diferentes contextos. Las caracterizan dos elementos, su código y su nombre descriptivo. Un ejemplo de Clasificaciones asociadas al Clasificador antes descrito se muestra a continuación:

código clasificación	nombre descriptivo
2100	Pinar del Río
2101	Sandino
2102	Mantua

Figura 6. Descripción de la información referente a Clasificadores.

1.1.2 Sistema Integrado de Gestión Estadística (SIGE).

SIGE es una aplicación web basada en tecnologías libres, que tiene como objetivo principal la informatización de los procesos de captura de datos estadísticos. Su concepción genérica permite adecuarse a negocios con bases metodológicas similares. Este sistema ha sido desarrollado con el fin de gestionar la información estadística del país.

SIGE se encuentra actualmente desplegado en el país a todos los niveles, dígase municipales, provinciales y nacional. Está compuesto por tres módulos principales denominados: Gestión de Configuración, Diseñador de Formularios y Entrada de Datos.

1.2 Actualización de datos en SIGE.

El proceso de actualización de datos primarios de SIGE se lleva a cabo en el módulo de Gestión de Configuración, el cual es el encargado de la captura y actualización de la información de los datos primarios. Los mismos son adicionados en el sistema por un especialista mediante un documento Excel. A continuación se describe detalladamente la aplicación de dicho proceso.

El especialista abre cada uno de los documentos Excel contenedores de datos primarios, dígase Centros Informantes, Registros, Clasificadores, Indicadores y Aspectos. Estos datos pueden variar dentro del documento en dependencia de su cantidad, que puede ser ínfima o contener un gran número de datos. Luego el especialista selecciona la opción en cuestión que permite adicionar datos primarios, los cuales son seleccionados de un documento Excel para luego ser insertados en el sistema SIGE. Un ejemplo de documento Excel utilizado por los especialistas, en este caso compuesto por varios indicadores, se muestra a continuación.

2	idcodIndicador	ciondedeindi	idcodtematica	nombredescriptivo	iddem	odic	notametodologica	essumable
3	2680401010001.00	3 1	268040101	Total de actuaciones recibidas en la Fiscalía	U	1	Total de actuaciones recibidas en la Fiscalía	TRUE
4	2680401010002.00	3 1	268040101	De ellas cantidad de actuaciones elevadas fuera del término	U	1	De ellas cantidad de actuaciones elevadas fuera del término	TRUE
5	2680401010003.00	3 1	268040101	Total de actuaciones recibidas en la Fiscalía por hechos priorizados	U	1	Total de actuaciones recibidas en la Fiscalía por hechos priorizados	TRUE
6	2680401010004.00	3 1	268040101	llas cantidad de actuaciones elevadas fuera del término por hechos priorizad	U	1	De ellas cantidad de actuaciones elevadas fuera del término por hechos priorizados	TRUE
7	2680401010005.00	3 1	268040101	Total de actuaciones con menores involucrados	U	1	Total de actuaciones con menores involucrados	TRUE
8	2680401010006.00	3 1	268040101	Total de menores transgresores	U	1	Total de menores transgresores	TRUE
9	2680401010007.00	3 1	268040101	De ellos cantidad de menores transgresores por género	U	1	De ellos cantidad de menores transgresores por género	TRUE
10	2680401010008.00	3 1	268040101	De ellos cantidad de menores transgresores de 6 a 10 años de edad	U	1	De ellos cantidad de menores transgresores de 6 a 10 años de edad	TRUE
11	2680401010009.00	3 1	268040101	De ellos cantidad de menores transgresores de 11 a 16 años de edad	U	1	De ellos cantidad de menores transgresores de 11 a 16 años de edad	TRUE
12	2680401010010.00	3 1	268040101	De ellos cantidad de menores transgresores por tipicidad delictiva	U	1	De ellos cantidad de menores transgresores por tipicidad delictiva	TRUE
13	2680401010011.00	3 1	268040101	Total de menores transgresores por hechos priorizados	U	1	Total de menores transgresores por hechos priorizados	TRUE
14	2680401010012.00	3 1	268040101	De ellos cantidad de menores transgresores por género	U	1	De ellos cantidad de menores transgresores por género	TRUE
15	2680401010013.00	3 1	268040101	De ellos cantidad de menores transgresores de 6 a 10 años de edad	U	1	De ellos cantidad de menores transgresores de 6 a 10 años de edad	TRUE
16	2680401010014.00	3 1	268040101	De ellos cantidad de menores transgresores de 11 a 16 años de edad	U	1	De ellos cantidad de menores transgresores de 11 a 16 años de edad	TRUE
17	2680401010015.00	3 1	268040101	De ellos cantidad de menores transgresores por tipicidad delictiva	U	1	De ellos cantidad de menores transgresores por tipicidad delictiva	TRUE
18	2680401010016.00	3 1	268040101	Total de menores que resultaron víctima	U	1	Total de menores que resultaron víctima	TRUE
19	2680401010017.00	3 1	268040101	De ellos cantidad de menores víctimas por género	U	1	De ellos cantidad de menores víctimas por género	TRUE
20	2680401010018.00	3 1	268040101	De ellos cantidad de menores víctimas de 1 a 5 años de edad	U	1	De ellos cantidad de menores víctimas de 1 a 5 años de edad	TRUE
21	2680401010019.00	3 1	268040101	De ellos cantidad de menores víctimas de 6 a 10 años de edad	U	1	De ellos cantidad de menores víctimas de 6 a 10 años de edad	TRUE
22	2680401010020.00	3 1	268040101	De ellos cantidad de menores víctimas de 11 a 16 años de edad	U	1	De ellos cantidad de menores víctimas de 11 a 16 años de edad	TRUE
23	2680401010021.00	3 1	268040101	De ellos cantidad de menores que resultaron víctimas por tipicidad delictiva	U	1	De ellos cantidad de menores que resultaron víctimas por tipicidad delictiva	TRUE
24	2680401010022.00	3 1	268040101	Total de menores que resultaron víctima por hechos priorizados	U	1	Total de menores que resultaron víctima por hechos priorizados	TRUE
25	2680401010023.00	3 1	268040101	De ellos cantidad de menores víctimas por género	U	1	De ellos cantidad de menores víctimas por género	TRUE

Figura 7. Documento Excel con Indicadores.

Un segundo módulo denominado Diseñador de Formularios cuyo objetivo principal es, informatizar el proceso de diseño de formularios necesarios para la captación de la información estadística en los Centros Informantes. Un formulario está compuesto por Indicadores y Aspectos, donde responden a las filas y columnas de un formulario respectivamente. Para asociar los Indicadores y Aspectos a un formulario, es necesario adicionarlos mediante un documento Excel semejante al proceso de adición que se lleva a cabo en el primer módulo. La diferencia radica en que para asociar estos datos a un formulario tienen que haberse adicionados previamente a través del primer módulo.

El tercer módulo denominado Entrada de Datos, es el proceso primario del trabajo estadístico donde se captura la información a través de la publicación de los formularios diseñados por el módulo correspondiente. Se definen diferentes estados por los que debe pasar un formulario luego de ser publicado: Digitación, Validación y Archivo. Cuando un formulario es publicado este pasa directamente al estado de Digitación. Luego el formulario es revisado por el especialista, quien al guardar los cambios originados, propicia que el formulario pase al estado de Validación. Finalmente se realiza una nueva revisión donde el formulario es validado, pasando así al estado de Archivo. Actualmente si el especialista lo considera necesario, puede mediante el sistema enviar un formulario que se encuentre en estado de Validación al estado de Digitación nuevamente.

El proceso de actualización de datos primarios de SIGE realizado específicamente en el primer y segundo módulo, al requerir grandes volúmenes de datos provoca demoras en las cargas iniciales de la información que las empresas necesitan para su caracterización y organización a nivel de país.

1.3 Tecnologías y Herramientas para el Desarrollo.

Para el desarrollo de la investigación se realizó un detallado estudio de las metodologías de desarrollo de *software* existentes y las herramientas a utilizar, con el objetivo de seleccionar aquellas que se adaptan mejor a las características de la solución propuesta.

1.3.1 Metodología de Desarrollo.

Las metodologías ofrecen un conjunto de procedimientos, técnicas, herramientas y documentación para asistir a los desarrolladores al implementar un *software*. La selección de la metodología a emplear, propone los roles que el equipo de desarrollo deberá cubrir, las actividades que debe desempeñar cada

uno de ellos, los artefactos que serán generados de cada tarea, así como el registro de cada detalle de la información que se irá generando.

OpenUp

La Metodología OpenUp es un proceso de desarrollo unificado que está basado en Rational Unified Process (RUP). Mantiene las mismas características de RUP pues está dirigido por casos de uso, centrado en la arquitectura y además es iterativo e incremental (6). El ciclo de vida de un proyecto según la metodología OpenUp se divide en cuatro fases fundamentales:

- Concepción: En esta fase se pretende determinar los objetivos y establecer el alcance del proyecto.
- Elaboración: El propósito de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para el desarrollo de la siguiente fase.
- Construcción: Esta fase tiene como propósito completar el desarrollo del sistema basado en la arquitectura definida, enfocándose en el diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo.
- Transición: En esta fase se asegura que el software esté listo para entregarse a los usuarios.

OpenUp propone cinco flujos de trabajo:

- Gestión del Proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Requisitos: Se realizan entrevistas con el cliente para comprender el problema a resolver y se definen los requerimientos.
- Arquitectura: Se realiza el diseño de los requisitos que serán implementados.
- Implementación: Se realiza la implementación del sistema basándose en el diseño realizado.
- Prueba: Busca los defectos a lo largo del ciclo de vida.

Se utilizará para el desarrollo de esta investigación la metodología OpenUp, coincidiendo el criterio de selección, con las definidas por el grupo de arquitectura del Departamento de Integración de Soluciones del Centro de Tecnologías de Gestión de Datos. La selección se debe principalmente a que el tiempo estimado para el desarrollo de la aplicación informática no excede los seis meses y se cuenta con un equipo de trabajo pequeño.

1.3.2 Lenguaje de Modelado Unificado (UML)

UML 2.0

El Lenguaje Unificado de Modelado (UML) es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas de *software*. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. No es un lenguaje de programación. Es un lenguaje para modelado orientado a objetos de propósito general evolutivo, ampliamente aplicable y estándar en la industria de la informática. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama.

Aunque UML puede usarse con varios procesos de desarrollo, fue diseñado para usarse con un proceso iterativo, incremental, guiado por casos de uso y centrado en la arquitectura, el tipo de proceso que se considera más apropiado para el desarrollo de sistemas complejos modernos.

Algunas de las propiedades de UML como lenguaje de modelado estándar son:

- Concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actuales y futuras.
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.
- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.
- Comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones.

El modelo gráfico de UML tiene un vocabulario compuesto por: Elementos, Relaciones y Diagramas. Los elementos son abstracciones que constituyen los bloques básicos de construcción. Las relaciones

encadenan los elementos. Mientras que los diagramas son la representación gráfica de un conjunto de elementos, que visualizan un sistema desde diferentes perspectivas.

1.3.3 Herramientas CASE

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE) brindan la posibilidad de informatizar actividades manuales y de mejorar su visión general de la ingeniería. Estas se conforman por un conjunto de programas, métodos, utilidades y técnicas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores; facilitando la informatización del ciclo de vida del progreso de un *software*.

Visual Paradigm 8.0

Visual Paradigm 8.0 es una herramienta que ayuda al modelado y documentación del *software*, y a generar código fuente a partir del modelo de implementación. Soporta el ciclo de vida completo del desarrollo del *software* (análisis y diseño orientado a objeto, construcción, prueba y despliegue) a través de sus correspondientes diagramas (7).

Las principales características de esta herramienta son:

- Genera los informes para la documentación.
- Presenta un editor de detalles de casos de uso incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Distribuye automáticamente los diagramas y reorganiza las figuras y conectores de los diagramas del Lenguaje Unificado de Modelado (UML).
- Cuenta con disponibilidad en múltiples plataformas.
- Su diseño está centrado en casos de uso y enfocado al negocio.
- Permite que el modelo y el código permanezcan sincronizados durante todo el ciclo de desarrollo.
- Es fácil de instalar y actualizar.
- Sus ediciones suelen ser compatibles.

Se escoge Visual Paradigm para el desarrollo de la aplicación informática pues es una herramienta multiplataforma, permite aumentar la calidad del software a través de la mejora de la productividad en el desarrollo y mantenimiento del mismo, así como su reutilización, portabilidad y estandarización de la documentación.

1.3.4 Sistema Gestor de Base de Datos (SGBD)

Los SGBD son un tipo de *software* muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta (8).

Actualmente existe una amplia gama de SGBD con características propias, no obstante, deben tener en cuenta los siguientes aspectos de manera general: abstracción de la información, independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo, recuperación, tiempo de respuesta y control de concurrencia.

PostgreSQL 9.1

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, utiliza un modelo cliente/servidor y multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Está desarrollado como *software* libre y se dispone de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Soporta operadores funcionales, métodos de acceso y tipos de datos definidos por el usuario.

Además la velocidad de respuesta que ofrece con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes, ventaja que resulta favorable (9).

Para guiar el desarrollo del módulo estadístico, se decidió emplear el SGBD PostgreSQL 9.1, ya que tiene la posibilidad de ejecutarse en varios sistemas operativos y permite utilizar numerosos tipos de datos, así como definir los propios. Se caracteriza por tener una buena escalabilidad, ya que es capaz de ajustarse al número de ordenadores y a la cantidad de memoria disponible, además cuenta con un robusto sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas. La principal razón por la que se selecciona este gestor está determinada, porque para el desarrollo de la aplicación informática, se utilizará la base de datos correspondiente a SIGE.

1.3.5 Lenguajes de Programación

El lenguaje es un sistema de comunicación que posee una determinada estructura, contenido y uso, por otra parte la programación es el procedimiento de escritura del código fuente de un *software*. La programación le indica al programa informático qué acción tiene que llevar a cabo y cuál es el modo de concretarla.

De esta manera puede decirse que el lenguaje de programación es aquella estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora. Además si se desea desarrollar una aplicación web se deben tener en cuenta tanto los lenguajes del lado del servidor como los del lado del cliente (10).

1.3.6 Lenguaje del lado del Servidor

Los lenguajes del lado del servidor son reconocidos, ejecutados e interpretados por el propio servidor, no dependen del navegador a utilizar, esto implica que no son sensibles a posibles cambios en este sentido. Otra de sus características es que el código de los *scripts* se almacena en el servidor, este se encarga de ejecutarlo y traducirlo a un Lenguaje de Marcas de Hipertexto (HTML, por sus siglas en inglés) de forma tal que no sea visible para el cliente; lo que constituye una gran ventaja en el trabajo de los programadores. Para el desarrollo de la investigación se decidió utilizar PHP 5.3.4 como lenguaje del lado del servidor.

PHP 5.3.4

PHP es un lenguaje interpretado del lado del servidor que tiene mucha utilidad por su rapidez y facilidad de realizar determinadas acciones, su código es basado en páginas HTML. Su objetivo principal ha sido mejorar los mecanismos de Programación Orientada a Objetos (POO) para solucionar las carencias de las versiones anteriores.

PHP 5 se destaca en numerosas características como son:

- Es rápido en la integración con bases de datos como *MySQL, MS, SQL, Oracle, Informix, PostgreSQL* y el servidor *Apache*.
- Tiene una de las comunidades más grandes en Internet: con lo que no es complicado encontrar ayuda, documentación, artículos, noticias y recursos (11).

1.3.7 Lenguaje del lado del Cliente

Los lenguajes del lado del cliente son independientes del servidor, esto posibilita que las páginas puedan guardarse en cualquier sitio. En este sentido es fundamental abordar sobre *ECMAScript*[1], un estándar que define un lenguaje de tipos dinámicos y que ha servido de base para otros lenguajes del lado del cliente. Para el desarrollo de la investigación se decidió utilizar como lenguaje del lado del cliente JavaScript.

JavaScript

JavaScript es un lenguaje del lado del cliente que no requiere de compilación, es decir, es interpretado. Ha tenido influencias de múltiples lenguajes (Perl, Phyton) y fue diseñado con una sintaxis similar a la de Java. La principal diferencia radica en que Java es un lenguaje orientado a objetos y JavaScript está basado en prototipos. Es compatible con la mayoría de los navegadores como Netscape, Internet Explorer, Mozilla Firefox, entre otros. Finalmente se debe señalar que este lenguaje de *scripting* es seguro y fiable (11).

1.3.8 Marcos de Trabajo (Framework)

Un *framework* o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos, que puede servir de base para la organización y desarrollo de *software*. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto (12).

Las principales ventajas de la utilización de un *framework* son:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un *framework* constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- La reutilización de componentes de *software*. Los *frameworks* son los paradigmas de la reutilización.

[1] ECMAScript es un lenguaje *script* estandarizado por Ecma International en la especificación ECMA-262 y la ISO/IEC 16262. Es ampliamente utilizado para scripting del lado del cliente en la web, en forma de varias implementaciones conocidas como JavaScript, JScript y ActionScript.

- El uso y la programación de componentes que siguen una política de diseño uniforme. Un *framework* orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

Para el desarrollo de este software se decidió utilizar los *frameworks* Symfony 1.4 y ExtJS 3.4.0.

Symfony 1.4

Symfony es un *framework* completo diseñado para optimizar, debido a sus características, el desarrollo de las aplicaciones web. Este *framework* separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web utilizando el Modelo-Vista-Controlador (MVC). Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Symfony está desarrollado completamente con PHP 5. Es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server (13).

Para el desarrollo de esta investigación, se selecciona Symfony 1.4 por la ayuda que le presta a los desarrolladores en cuanto al ahorro de tiempo, además de que con este *framework* se optimiza el desarrollo de la aplicación web. Su estructura conlleva a que se escriba menos código, ya que uno de sus puntos principales es la reusabilidad.

ExtJS 3.4.0

ExtJS es una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de los navegadores, para crear páginas web y aplicaciones dinámicas con gran facilidad. Tanto para un desarrollador individual o un equipo de desarrollo, el modelo de ExtJS mantiene su código bien estructurado por lo que incluso las aplicaciones más grandes se pueden mantener fácilmente. Tiene incluidos la mayoría de los controles de los formularios web incluyendo tablas para mostrar datos y elementos semejantes a la programación escritorio (*desktop, en inglés*) como formularios, paneles, barras de herramientas y menús.

ExtJS cuenta con disímiles ventajas:

- Tiene un alto rendimiento, interfaz de usuario personalizables.
- Un modelo de componentes muy bien diseñado y extensible.
- Compatibilidad con numerosos navegadores.
- Código reutilizable.
- Independiente o adaptable a *frameworks* diferentes.

1.3.9 Entorno de Desarrollo Integrado (IDE)

Un IDE es una aplicación compuesta por un conjunto de herramientas útiles para un desarrollador, que brinda ayudas visuales en la sintaxis, plantillas, complementos (*plugins*, en inglés) y sencillas opciones para probar y hacer un *debug*[2]. Puede ser exclusivo para un lenguaje de programación o utilizarse para varios. Generalmente incluye un editor de código, un compilador, un depurador y en algunos casos un constructor de interfaz gráfica (14).

NetBeans 7.3

NetBeans es un IDE de código abierto diseñado para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso del lenguaje de programación Java. Esta poderosa herramienta está disponible para diferentes plataformas, como es el caso de Windows XP/Vista, Linux, Mac OS. Además es utilizada por los programadores para escribir, compilar, depurar y ejecutar programas escritos en Java aunque puede ser utilizado en cualquier otro lenguaje de programación. Posee un número importante de módulos para extenderlo, lo que posibilita ampliar y facilitar su utilización (15).

Es un producto libre y gratuito sin restricciones de uso que posee una amplia gama de características:

- Creación de proyectos PHP.
- Integración con *Symfony* y *Zend Framework*.
- Editor de Código Fuente.
- Depuración de PHP.
- Integración con Sistemas de Control de Versiones.
- Disponible en varios idiomas, incluyendo el español.
- Puede usar las licencias de código abierto para realizar mejoras.

Conclusiones Parciales

Al efectuar un análisis detallado de los conceptos se pudo arribar a un mejor entendimiento de los procesos que se informatizarán durante la investigación garantizando un mejor dominio del negocio.
Se describieron las principales características relacionadas con la metodología, herramientas y lenguaje de programación que se emplearán en la propuesta de solución, argumentando la selección en cada caso, enfatizando sobre el uso del *framework* que se utilizará. Luego de un detallado estudio de las tendencias y

[2] Un ***debug*** es el proceso de identificar y corregir errores de programación.

tecnologías actuales que faciliten el desarrollo de la investigación, se decide hacer uso de PHP 5 y JavaScript como lenguajes de programación. Se utilizarán los *Frameworks* Symfony 1.4 y ExtJS 3.4.0, del lado del servidor y del lado del cliente respectivamente. Como metodología para guiar el desarrollo se selecciona OpenUp, como IDE de desarrollo NetBeans 7.3 y como SGBD se empleará PostgreSQL 9.1. Estas herramientas permitirán facilitar el trabajo del desarrollador durante el ciclo de desarrollo del *software,* brindando rapidez, flexibilidad y facilidad de uso con respecto a las características de la aplicación informática que se pretende desarrollar.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En este capítulo se describen los principales conceptos del problema, a partir de la definición del modelo de dominio. Se presenta la propuesta informática a desarrollar, definida a partir de las necesidades actuales, identificadas mediante las entrevistas con el cliente. A partir de ellos se definen además, los requisitos funcionales representados mediante diagramas de casos de uso y los requisitos no funcionales representados a nivel textual.

2.1 Modelo de Dominio

En el modelo de dominio se ilustran los principales conceptos con los que trabaja el sistema a desarrollar, permitiendo obtener una mejor comprensión del entorno y una representación visual de las clases conceptuales más significativas dentro del contexto o dominio de interés. (16).

En la siguiente figura se define el modelo de dominio para el sistema a desarrollar en la presente investigación:

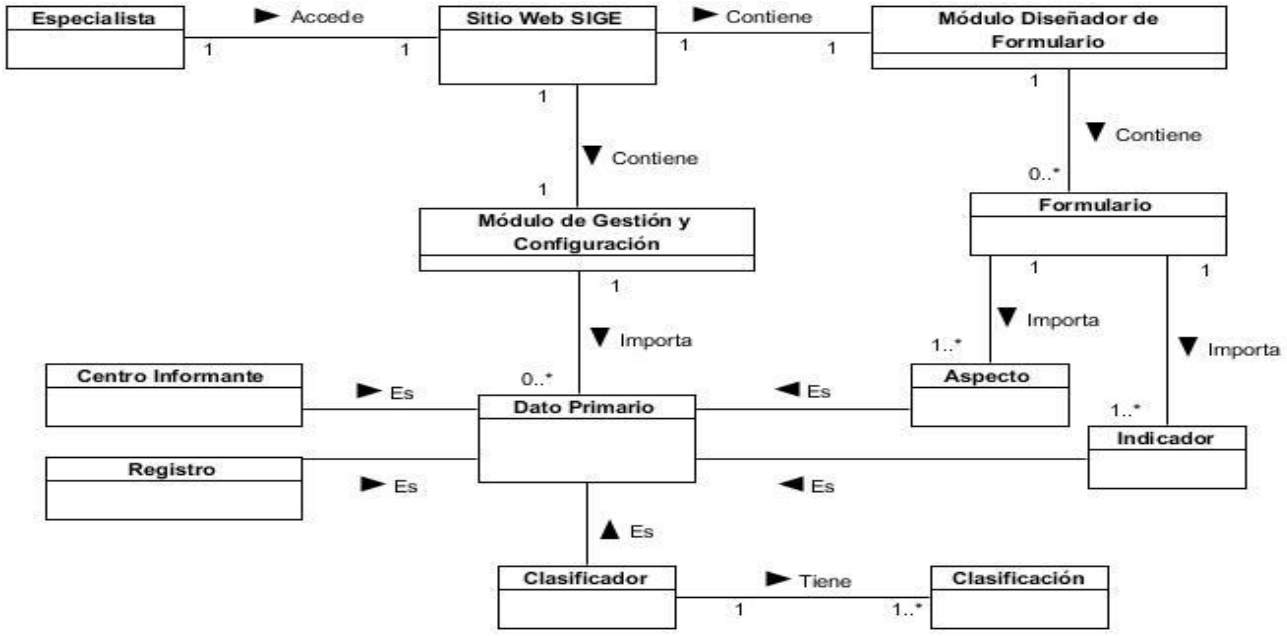


Figura 8. Modelo del Dominio.

2.1.1 Definición de las Clases del Modelo del Dominio.

Especialista

Es la persona con la autorización requerida para interactuar con SIGE. Se encarga de la captura y actualización de datos primarios apoyándose en un documento entregado por los Centros Informantes.

Sitio Web SIGE

Sistema informático que se encarga de la captura y procesamiento de información estadística.

Módulo de Gestión y Configuración

Es una de las herramientas que compone la solución informática SIGE. Es el encargado de la captura y actualización de los datos primarios en la misma.

Dato Primario

Conjunto de características que responden a un concepto del negocio que establece la ONEI, tales como: Centros Informantes, Clasificadores, Registros, Indicadores y Aspectos.

Centro Informante

Los Centros Informantes o empresas constituyen el elemento primario del sistema, de los cuales se obtiene la información estadística.

Registro

Los Registros son aquellos que permiten agrupar a los Centros Informantes o empresas de acuerdo a la actividad económica que desempeñan.

Clasificador

Los Clasificadores son conceptos generales que permiten agrupar características más específicas para describir a los Centros Informantes. Constituyen el eslabón principal a la hora de procesar la información ya que a partir de ellos, se acota la muestra sobre la cual se realiza el estudio estadístico.

Clasificación

Concepto más específico contenido dentro de los Clasificadores. En esencia las Clasificaciones son los valores que puede tomar un Clasificador en diferentes contextos.

Indicador

Los Indicadores son instrumentos para clasificar y definir, de forma más precisa, objetivos e impactos. Forman parte de los formularios a nivel de fila.

Aspecto

Los Aspectos describen o caracterizan a los Indicadores. Forman parte de los formularios a nivel de columnas.

Módulo Diseñador de Formulario

Es una de las herramientas que compone la solución informática SIGE. En este marco dicha solución garantiza el proceso de definición estructural de la captación de la información estadística.

Formulario

Los formularios se definen por filas y columnas, siendo los Indicadores los que conforman las filas y los Aspectos en las columnas. Ellos contienen información estadística procesada en SIGE.

2.2 Propuesta del Sistema

La propuesta de solución consiste en el desarrollo de la aplicación informática compuesto por tres módulos, en este caso los propios de SIGE: Gestión de Configuración, Diseñador de Formularios y Entrada de Datos.

El módulo Gestión de Configuración será el encargado de la inserción y actualización de los datos primarios de SIGE. Esto consiste en la selección de los documentos Excel mediante la opción “Importar”; para seguidamente agregar de una vez, todos los ejemplares relacionados al dato primario del documento Excel seleccionado. En caso de que existan los datos primarios a importar, no se interrumpirá la inserción del resto de los mismos contenidos en el Excel.

El módulo Diseñador de Formularios asociará Indicadores y Aspectos a formularios. Ello consiste al igual que en el primer módulo, en la selección de documentos Excel mediante la opción “Importar”; para seguidamente agregar de una vez, todo el contenido del documento seleccionado relacionado con Indicadores o Aspectos. Como premisa en este módulo se tiene que para realizar las asociaciones de los Indicadores y Aspectos a un formulario determinado, estos tienen que haberse incorporados al sistema mediante el primer módulo previamente.

Por su parte el módulo Entrada de Datos permitirá cambiar a todo formulario luego de ser publicado de un estado a otro. De esta manera un formulario en estado de Digitación puede ser cambiado a estado de Validación o Archivo, pudiéndose realizar también el mismo proceso de forma inversa.

El usuario responsable de interactuar con la aplicación será un Administrador, la cual deberá permitir la autenticación a este nuevo rol. Es importante destacar que cada empresa o Centro Informante contará con una única persona con estos privilegios.

2.3 Requisitos de software

Los requisitos de software son las condiciones que debe cumplir un sistema para satisfacer las necesidades del cliente. A continuación se muestran según su tipo, los requisitos funcionales y no funcionales de la solución propuesta.

2.2.1 Requisitos Funcionales

Los Requisitos Funcionales (RF) indican las funcionalidades y servicios que el sistema debe proporcionar, son las capacidades o condiciones que el sistema debe cumplir (17):

RF1: Autenticar Usuario

Descripción	El sistema debe permitir que el usuario se autentique.
Entrada	Datos necesarios de identificación del usuario en la aplicación (usuario y contraseña).
Salida	Usuario autenticado.

RF2: Actualizar Centros Informantes

Descripción	El sistema permite al usuario actualizar los Centros Informantes, importándolos mediante un documento Excel.
Entrada	Listado de los Centros Informantes en un documento Excel.
Salida	Listado de los Centros Informantes importados.

RF3: Actualizar Registros

Descripción	El sistema permite al usuario actualizar los Registros, importándolos mediante un documento Excel.
Entrada	Listado de los Registros en un documento Excel.

Salida	Listado de los Registros importados.

RF4: Actualizar Clasificadores

Descripción	El sistema permite al usuario actualizar los Clasificadores, importándolos mediante un documento Excel.
Entrada	Listado de los Clasificadores en un documento Excel.
Salida	Listado de los Clasificadores importados.

RF5: Actualizar Indicadores

Descripción	El sistema permite al usuario actualizar los Indicadores, importándolos mediante un documento Excel.
Entrada	Listado de los Indicadores en un documento Excel.
Salida	Listado de los Indicadores importados.

RF6: Actualizar Aspectos

Descripción	El sistema permite al usuario actualizar los Aspectos, importándolos mediante un documento Excel.
Entrada	Listado de los Aspectos en un documento Excel.
Salida	Listado de los Aspectos importados.

RF7: Asignar Indicadores a un Formulario

Descripción	El sistema permite al usuario asignar los Indicadores importados a un formulario para la captación de la información registrada.

Entrada	Datos importados en el sistema.
Salida	Formulario con los Indicadores asignados.

RF8: Asignar Aspectos a un Formulario

Descripción	El sistema permite al usuario asignar los Aspectos importados a un formulario para la captación de la información registrada.
Entrada	Datos importados en el sistema.
Salida	Formulario con los Aspectos asignados.

RF9: Cambiar del Estado de Archivo a Validación

Descripción	El sistema permite al usuario cambiar el estado del formulario de archivo a validación.
Entrada	Formulario en estado de Archivo.
Salida	Formulario en estado de Validación.

RF10: Cambiar del Estado de Archivo a Digitación

Descripción	El sistema permite al usuario cambiar el estado del formulario de archivo a digitación.
Entrada	Formulario en estado de Archivo.
Salida	Formulario en estado de Digitación.

RF11: Cambiar del Estado de Validación a Digitación

Descripción	El sistema permite al usuario cambiar el estado del formulario de validación a digitación.
Entrada	Formulario en estado de Validación.
Salida	Formulario en estado de Digitación.

2.2.2 Requisitos No Funcionales

Los requisitos no funcionales (RNF) son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

Requisitos de Interfaz

RNF1: Interfaces de *Hardware*.

El sistema está alineado con las características del *hardware* adquirido por la ONEI para su informatización, a tales efectos se distinguen los siguientes elementos con sus respectivas restricciones.

- Servidor de Base de Datos: procesador Intel Pentium (IV) o superior, con 1.7 GHz de velocidad, 2 GB de RAM, 80 GB de espacio en disco duro.
- Servidor web: procesador Intel Pentium (IV) o superior, con 1.7 GHz de velocidad, 2 GB de RAM, 40 GB de espacio en disco duro.

RNF2: Interfaces de *Software*.

- Sistema Operativo: GNU/Linux Ubuntu 12.04
- Gestor de Base de Datos: PostgresSQL 9.1
- Cliente de Base de Datos: PgAdmin III

Servidor Web:

- Sistema Operativo: GNU/Linux Ubuntu 12.04
- Aplicación Servidora: Apache 2.2.22
- Lenguaje de programación PHP y librerías, versión 5.3.4

Otras dependencias requeridas:

- php5-pgsql (paquete de extensión PHP para PostgreSQL)
- php5-xsl (paquete de extensión PHP para XSLT)
- php-gd (paquete de extensión PHP para gráficos)

Requisitos de Restricciones de Diseño e Implementación

RNF3: Lenguaje y marco de trabajo para el desarrollo del sistema del lado del servidor.
El sistema deberá ser implementado en el lenguaje de programación PHP 5.3.4. Como *framework* de desarrollo se utilizará Symfony en su versión 1.4. Este *framework* separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web utilizando el Modelo-Vista-Controlador (MVC).

RNF4: Lenguaje y marco de trabajo para el desarrollo de la aplicación del lado del cliente.
Para el desarrollo de la aplicación se utilizará el *framework* de presentación ExtJS 3.4. Es una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de los navegadores, para crear páginas web y aplicaciones dinámicas con gran facilidad usando tecnologías como AJAX.

Requisitos de Confiabilidad

RNF5: La información manejada por el sistema deberá estar protegida de acceso no autorizado y divulgación. El administrador del sistema tendrá que autenticarse.

RNF6: La aplicación solo estará fuera de servicio en caso de que se le estén realizando algunas tareas de mantenimiento o configuración.

Requisitos de Eficiencia

RNF7: Cantidad de usuarios conectados en línea.
El sistema solo debe permitir a un usuario conectado en línea.

2.4 Diagrama de Casos de Uso del Sistema

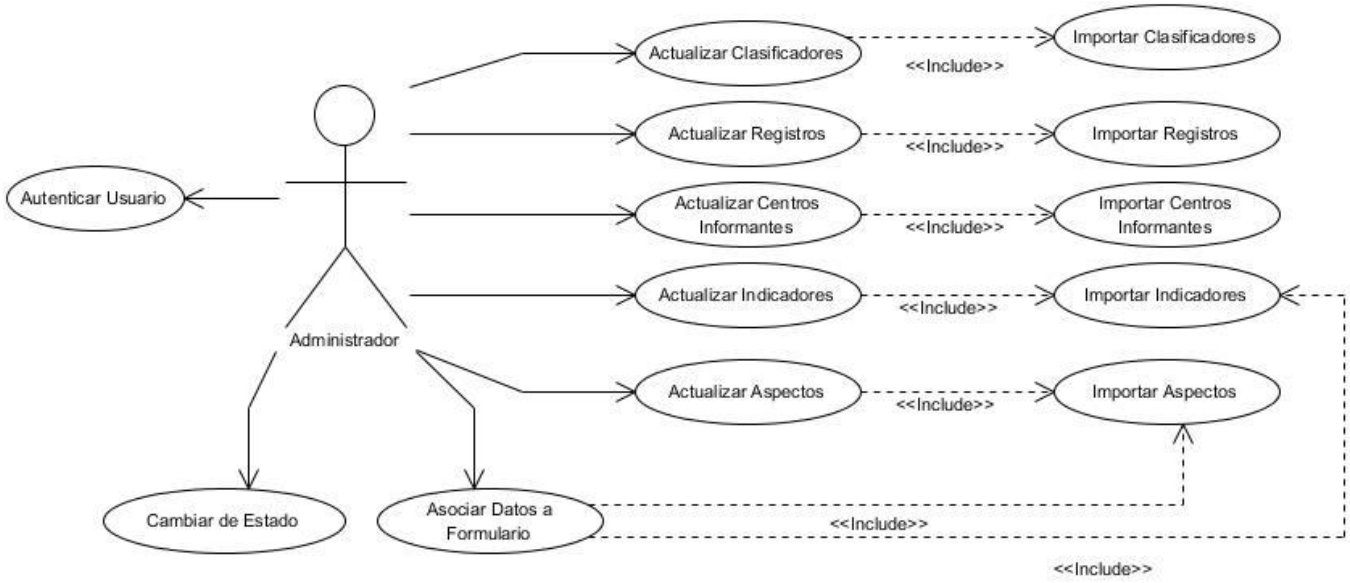


Figura 9. Diagrama de Caso de Uso del Sistema.

2.5 Descripción de Casos de Uso

Tabla 1. Descripción del CU "Actualizar Centros Informantes".

Caso de Uso:	Actualizar Centros Informantes.
Objetivo:	Permitir la actualización de Centros Informantes en el sistema, mediante la importación del mismo en un documento Excel.
Actores:	Administrador.
Resumen:	El caso de uso inicia cuando el administrador entra al módulo "Gestión de Configuración" para la actualización de los Centros Informantes. Se invoca al CU Importar Centros Informantes.
Precondiciones:	Usuario autenticado en el sistema.
Poscondiciones:	Centros informantes actualizados en el sistema.
Referencias	RF1.
Prioridad	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción "Gestión de Configuración".	
	2. Muestra un menú con los datos primarios que pueden ser actualizados.
3. El administrador selecciona la opción "Actualizar Centros Informantes".	
	4. Comprueba que exista conexión con la BD.
	5. Obtiene de la BD el listado de los Centros Informantes existentes.
	6. Muestra la interfaz "Actualizar Centros Informantes" con la información de los Centros Informantes existentes:

	• Provincia • Municipio • Registro • Código • Nombre Descriptivo • Estado Y se muestra la funcionalidad “Importar”.
7. El administrador presiona la opción “Importar”.	
	8. Se invoca al **CU Importar Centros Informantes.**
Flujo Alterno 4a No Existe Conexión con la BD	
Acción del Actor	**Respuesta del Sistema**
	4a.1. Comprueba que no existe conexión con la BD.
	4a.2. Muestra un mensaje indicando que no existe conexión con la BD.
4a.3 El usuario selecciona la opción “Aceptar”.	

Tabla 2. Descripción del CU “Importar Centros Informantes”.

Caso de Uso:	Importar Centros Informantes.
Objetivo:	Importar Centros Informantes mediante un documento Excel.
Actores:	Administrador.
Resumen:	El caso de uso se inicia cuando el administrador selecciona la opción para importar Centros Informantes. El sistema permite la importación de un fichero Excel con los datos de los Centros Informantes. El sistema verifica la validez de los datos. El caso de uso termina cuando es sistema notifica al administrador que la acción se ha realizado

	satisfactoriamente.
Precondiciones:	• Usuario autenticado en el sistema. • Documento Excel con los Centros Informantes a importar.
Poscondiciones:	Centros informantes importados en la base de datos del sistema.
Referencias	RF1, RF2.
Prioridad	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1. Muestra la ventana “Importar Centros Informantes” con los botones “Examinar”, “Aceptar” y “Cancelar”
2. Presiona el botón “Examinar”.	
	3. Muestra la ventana “Carga de archivos” para buscar la ubicación del fichero Excel con los datos a importar.
4. Selecciona la ubicación del fichero Excel y presiona la opción “Abrir”.	
	5. Cierra la ventana “Carga de archivos” y muestra el nombre del fichero seleccionado en la ventana “Importar Centros Informantes”.
6. Presiona la opción “Aceptar”.	
	7. Verifica que la información del fichero Excel importado esté correctamente estructurada.
	8. Comprueba que exista conexión con la BD.
	9. Verifica que la información del fichero Excel a importar no existan en la base de datos.
	10. Almacena la información verificada en la base de datos.
	11. Muestra el listado con la información de los Centros Informantes actualizada.

	12. Muestra al administrador el mensaje “La operación se realizó satisfactoriamente. Se adicionaron al sistema los Centros Informantes”.
Flujo Alterno 7a Fichero Excel Incorrectamente Estructurado	
Acción del Actor	**Respuesta del Sistema**
	7a.1 Muestra un mensaje de error al usuario informando: “Imposible realizar operación: El documento importado no presenta el formato correcto."
7a.2 El usuario cierra el mensaje de error.	
Flujo Alterno 8a No Existe Conexión con la BD	
Acción del Actor	**Respuesta del Sistema**
	8a.1. Comprueba que no existe conexión con la BD.
	8a.2. Muestra un mensaje indicando que no existe conexión con la BD.
8a.3 El usuario selecciona la opción “Aceptar”.	
Flujo Alterno 9a Información del fichero Excel a importar existe en la BD	
Acción del Actor	**Respuesta del Sistema**
	9a.1. Muestra un mensaje de error al usuario informando: “Todos los centros informantes existían en el sistema”.
9a.2 El usuario cierra el mensaje de error.	

2.6 Patrones de Casos de Uso

Uno de los patrones de casos de uso usados para el diseño de la aplicación informática fue Inclusión Concreta. Ello se evidencia en los CU Actualizar Clasificadores, CU Actualizar Registros, CU Actualizar Centros Informantes, CU Actualizar Indicadores y CU Actualizar Aspectos, pues los mismos contienen incluidos casos de uso con acciones concretas para importar los datos primarios asociados.

Conclusiones Parciales

Durante el desarrollo de este capítulo se identificaron a partir del modelo de dominio las clases conceptuales significativas que intervienen en el desarrollo de la aplicación informática. Se construyó el diagrama de casos de uso y se realizó una descripción detallada del CU “Actualizar Centros Informantes” por ser el de mayor impacto en la solución del problema, utilizándose como tipo de patrones de caso de uso Inclusión Concreta para describir como deberían ser estructurados y organizados los casos de uso.

CAPÍTULO 3: DISEÑO DEL SISTEMA

En este capítulo se realiza una selección de los patrones de diseño y patrones arquitectónicos para garantizar un correcto diseño y desarrollo escalable del sistema. Se describen artefactos de diseño que brindan una mayor claridad de los procesos del sistema a implementar, así como la información relevante a almacenar en la base de datos.

3.1 Arquitectura del software

3.1.1 Estilos y Patrones Arquitectónicos

Modelo-Vista-Controlador (MVC)

Este patrón está presente principalmente en las aplicaciones web. En estas se comporta de la siguiente manera: la vista es la página HTML y el código que le suministra los datos dinámicos a la página, el modelo son los datos que aporta la lógica del negocio obtenidos por el controlador y el controlador por medio de los objetos de negocio llama la lógica del negocio. La vista y el controlador forman la interfaz de usuario (18).

Este patrón separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

- **Modelo:** Encapsula los datos centrales y tiene la funcionalidad de la aplicación. Es un componente totalmente independiente de las representaciones específicas de salidas o del comportamiento de la entrada.
- **Vista:** Presentan la información del modelo al usuario de distintas maneras. Pueden existir múltiples vistas de un mismo modelo, pero cada vista tiene una relación uno a uno con un controlador.
- **Controlador:** Recibe los eventos que codifican los movimientos del mouse o entrada del teclado. Los eventos son traducidos para servir a las demandas del modelo o las vistas. El usuario interactúa con el sistema solamente a través de los controladores.

3.1.2 Patrones

Patrones

Un patrón de diseño constituye un esquema para refinar subsistemas o componentes. Es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular, logrando formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.

Para el diseño de la aplicación informática se utilizaron los patrones Patrones de Asignación de Responsabilidades (GRASP) y los patrones Gang of Four (GoF).

Patrones GRASP: Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos. Su utilización es recomendable, pues constituyen buenas prácticas de aplicación en el diseño de *software*. A continuación se describen los patrones utilizados en la concepción del sistema y como se evidencian cada uno de ellos.

Experto

Asignar responsabilidades a la clase que cuenta con la información necesaria para cumplir con esta responsabilidad. Al incluir Propel como ORM este genera las clases para la gestión de las entidades con las responsabilidades debidamente asignadas. Además se evidencia en cada una de las clases utilizadas para la solución.

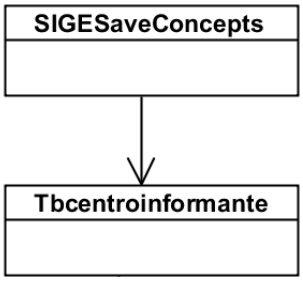


Figura 10. Patrón Experto

Creador

Encontrar un creador que se debe conectar con el objeto producido en cualquier evento. En las clases *Actions* se encuentran las acciones definidas para el sistema, en dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que las clases *Actions* son creadoras de dichas entidades.

Controlador

Las peticiones web son manipuladas por un controlador (sfActions) que es el punto de entrada único de toda la aplicación en un entorno determinado. Este patrón se evidencia en las clases *Actions*.

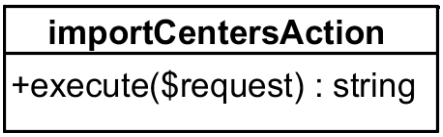


Figura 11. Patrón Controlador

Alta Cohesión

Es una medida de cuan relacionada y enfocada están las responsabilidades de una clase. Una alta cohesión caracteriza a la clase con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. En la solución se encuentran varias clases de tipo *Actions* que contienen varias funcionalidades que poseen un propósito único, posibilitando de esta manera que el *software* sea más flexible.

Bajo Acoplamiento

Es el nivel de relación que puede tener una clase con otra. En la capa Modelo se encuentran las clases que implementan la lógica del negocio y acceso a datos, estas clases tienen pocas asociaciones con otras de la Vista o el Controlador por lo que la dependencia en este caso es baja.

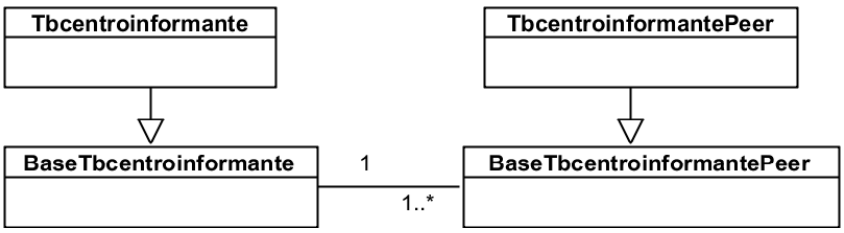


Figura 12. Patrón Bajo Acoplamiento

Patrones GoF: Los patrones GoF describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Permiten crear grupos de objetos que ayudan a realizar tareas complejas. Estos patrones pueden ser de tres tipos: de creación, estructurales y de comportamiento.

Los patrones de creación abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas.

Instancia única (Singleton): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este patrón se evidencia en que todas las clases controladoras, son instancias únicas. En la siguiente imagen se evidencia el uso de este patrón en la clase importCentersActions en la cual se crea un único objeto de la clase SIGEUpdate.

```
$connection = Propel::getConnection();
$connection->begin();
$existen = array();
$importados = array();
$sigeupdate = new SIGEUpdate();
$user = UserPeer::retrieveByPK($this->getUser()->ge
$sigeupdate->save($filename, $user, $existen,$impor
$connection->commit();
```

Figura 13. Evidencia del patrón Singleton en la clase ImportCentersAction

Acción (Command): La estrategia del patrón Acción sugiere proporcionar una interfaz genérica para los componentes en los que el controlador puede delegar responsabilidades, minimizando el acoplamiento entre estos componentes. En el *framework* Symfony se utiliza este patrón, el cual se ve evidenciado en la clase sfWebFrontController y en el método dispatch.

3.2 Diagrama de Diseño de Clases

Una clase del diseño y sus objetos, y de ese modo también los subsistemas que contienen las clases de diseño, a menudo participan en varias realizaciones de casos de uso. Para coordinar todos los requisitos de diferentes realizaciones de casos de uso, es necesario utilizar los diagramas de clases conectados a una realización de caso de uso, mostrando sus clases participantes, subsistemas y sus relaciones. De esta forma podemos guardar las pistas de los elementos participantes en una realización del caso de uso (19).

A continuación se muestran características de cada una de las clases y relaciones de dependencia y generalización de la realización del CU Actualizar Centros Informantes como parte del diagrama de clases del diseño de dicha funcionalidad. Con el objetivo de lograr una adecuada organización de los diagramas, estos se organizan de tal manera que cada elemento del diseño se ubica en la capa a la que responde según las definidas por el patrón MVC.

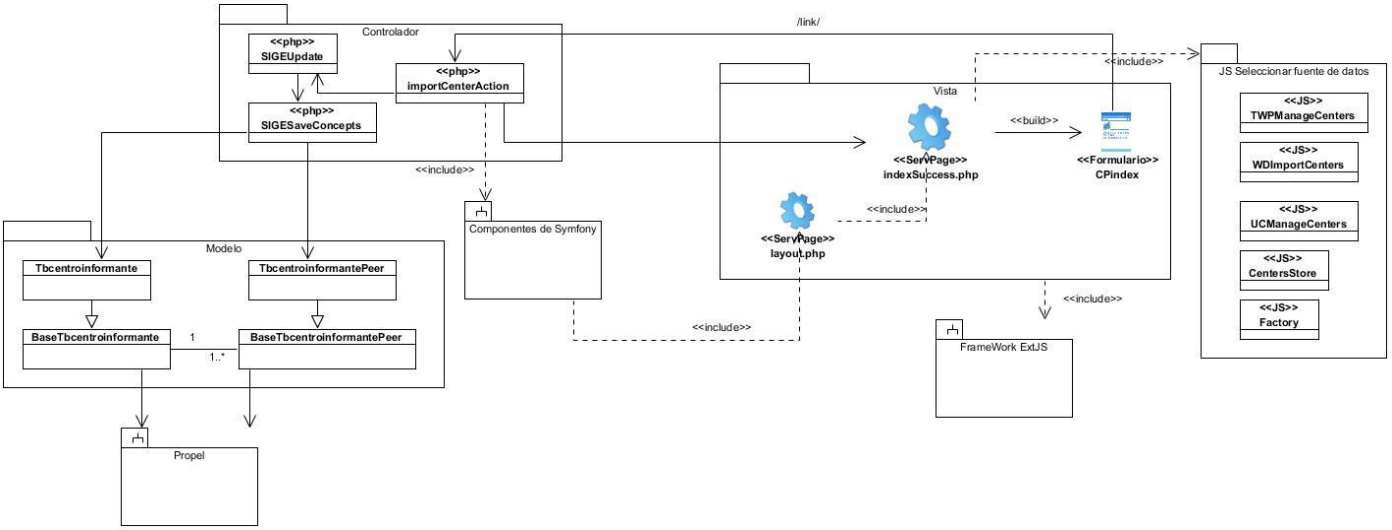


Figura 14. Diagrama de Clases del Diseño del CU Actualizar Centros Informantes.

Para un mejor entendimiento de lo modelado anteriormente se realizará una breve descripción de los elementos que intervienen en el diagrama. Primeramente, para modelar las clases del diseño de un sistema que emplea Symfony, es conveniente establecer fronteras entre qué elementos forman parte del *Framework* y cuáles del CU específico que se representa. Por tanto, todos los componentes de Symfony que intervienen en la realización del CU Actualizar Centros Informantes, fueron encapsulados en un subsistema. Es importante destacar que al ser estos componentes propios de Symfony, la manera en la que trabajan es transparente al programador. Otro elemento fundamental en los diagramas es el Mapeo de Objeto-Relacional (ORM, por sus siglas en inglés) que utiliza Symfony, el cual proporciona la persistencia de los objetos y un servicio de consulta, en este caso Propel y se representa también como un subsistema.

En el diagrama de clase de diseño se hizo evidencia en el patrón MVC, donde la Vista está compuesta por una clase indexSuccess la cual contiene todos los JS del sistema, una clase layout que es la encargada de decorar el indexSuccess, donde el layout hace un llamado a esta clase y la misma se encarga de construir la ClientPage_index, (CP) en este caso haciendo uso del JS TWPManageCenter.

El usuario hace una petición al sistema donde la CP hace un link a la clase controladora importCentersAction, la cual utiliza a la clase SIGEUpdate y esta a su vez a la clase SIGESaveConcepts que interactúa con el modelo, como se utiliza propel como ORM, este crea cuatro clases: Tbcentroinformante, BaseTbcentroinformante, TbcentroinformantePeer, BaseTbcentroinformantePeer, en la que Tbcentroinformante es la tabla principal. Luego se da respuesta a la petición del usuario a través de

la clase importCentersAction. Es importante destacar tanto el Controlador como la Vista hacen uso de los componentes de Symfony y del FrameWork ExtJS respectivamente.

3.3 Diagrama de Secuencia

Muestra las interacciones entre un conjunto de objetos (clases, actores), ordenadas según el tiempo en que tienen lugar. El diagrama se compone con los objetos que forman parte de la secuencia, estos se sitúan en la parte superior de la pantalla, normalmente a la izquierda se sitúa el que inicia la acción. De estos objetos sale una línea que indica su vida en el sistema. Esta línea simple se convierte en una línea gruesa cuando representa que el objeto tiene el foco del sistema, es decir cuando el está activo (20).

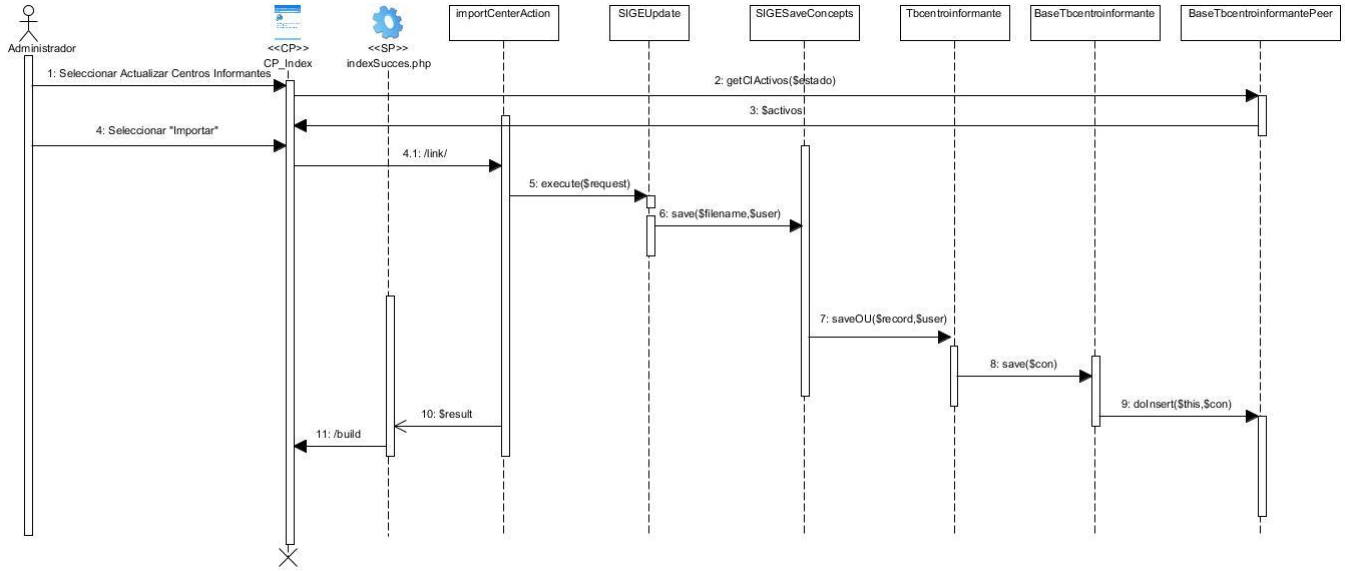


Figura 15. Diagrama de secuencia del CU Actualizar Centros Informantes.

En general los diagramas de secuencia muestran la interacción del actor con el sistema, en este caso para la realización del CU Actualizar Centros Informantes, donde el actor realiza la funcionalidad de seleccionar y el sistema se encarga de interactuar con las diferentes clases tanto de la Vista, el Modelo y el Controlador para dar respuesta a la petición requerida.

3.4 Modelo de Paquetes

Un paquete es un mecanismo utilizado para agrupar elementos de UML. Permite organizar los elementos modelados con UML, facilitando de ésta forma el manejo de los modelos de un sistema complejo. Los paquetes pueden ser simples estructuras conceptuales o pueden estar reflejados en la implementación (20).

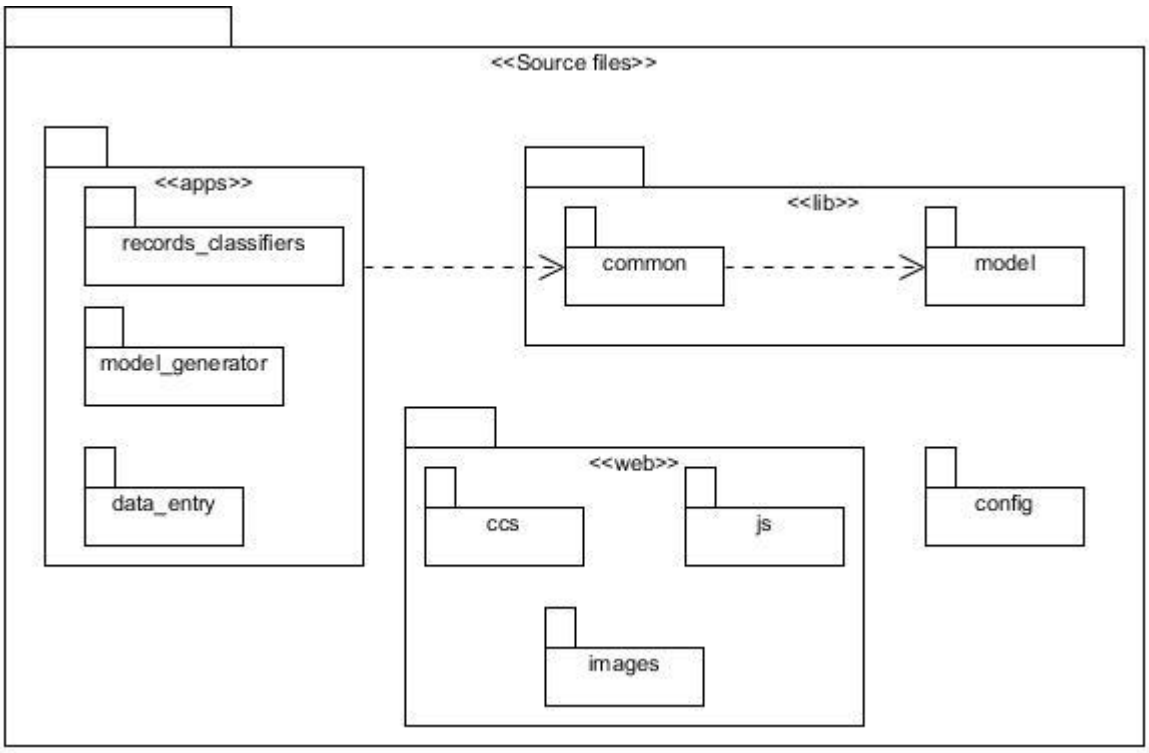


Figura 16. Diagrama de Paquetes.

El diagrama de paquetes está compuesto por una carpeta denominada <<apps>> la cual es la encargada de almacenar todos los controles del sistema. En esta carpeta se encuentra records_classifiers la cual se encarga de manejar todo el primer módulo de Gestión de Configuración (Centros Informantes, Registros, Clasificadores, Aspectos e Indicadores), model_generator la cual maneja el segundo Módulo que se encarga del diseño de los formularios que es donde se validan las acciones a la hora de asignar los datos a un formulario, la carpeta data_entry la cual maneja el tercer módulo Entrada de Datos que se encarga de la ejecución de las acciones para cambiar de estado cada formulario y patdsi3 que es la encargada de la autentificación del usuario.

En la carpeta <<lib>> se encuentran las entidades que representan las tablas en el modelo de Base de Datos y en la carpeta <<web>> se ubicarán diferentes recursos como hojas de estilos, imágenes y archivos JavaScript, cada uno en su carpeta correspondiente y para configurar los *schemas* que serán utilizados en el sistema se contará con el archivo schema.yml el cual se ubicará dentro de la carpeta config.

3.5 Modelo de Datos

El modelo de datos se utiliza para diseñar una base de datos. Muestran las entidades, sus atributos asociados, así como las relaciones entre estas entidades.

Para lograr almacenar y gestionar la información de la aplicación informática para la recolección de los datos primarios fue necesario el diseño de una base de datos, lo cual se logró a partir de la confección del modelo de datos que se muestra a continuación:

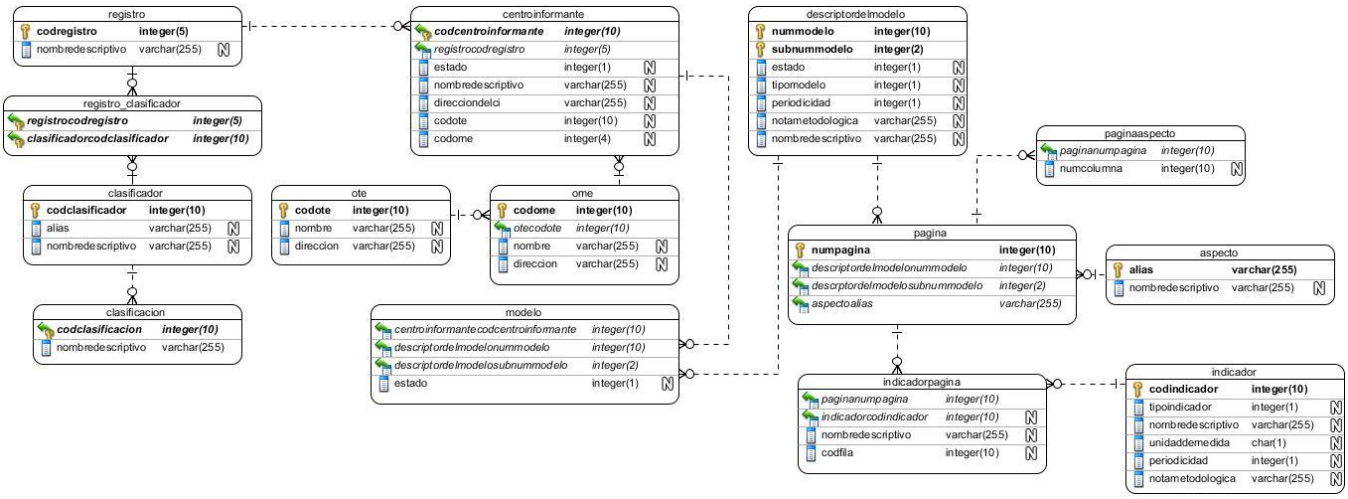


Figura 17. Diagrama entidad-relación.

A continuación se describirán las principales entidades referentes al diagrama entidad-relación.

Tabla 3. Descripción de las entidades centroinformante, indicador y aspecto.

Nombre: centroinformante		
Descripción: Almacena los datos de un centro informante.		
Atributo	**Tipo**	**Descripción**
codcentroinformante	integer (10)	Código numérico de un Centro

		Informante
codregistro	integer (5)	Código numérico de un Registro
estado	integer (1)	Código numérico de un estado
nombredescriptivo	varchar(255)	Contiene el nombre de un Centro Informante
direcciondelcl	varchar(255)	Contiene la dirección de un Centro Informante
codote	varchar(10)	Código numérico de la provincia
codome	varchar(4)	Código numérico del municipio
Nombre: indicador		
Descripción: Almacena los datos de un indicador.		
Atributo	**Tipo**	**Descripción**
tipoindicador	integer (1)	Código numérico de un Indicador
nombredescriptivo	varchar(255)	Contiene el nombre de un Identificador
unidaddemedida	char (1)	Almacena la unidad de medida de un Identificador
periodicidad	integer (1)	Código numérico de la periodicidad
notametodologica	varchar(255)	Almacena la información referente a un Indicador

Conclusiones Parciales

En el presente capítulo se realizaron los diferentes diagramas de clases del diseño y de secuencia, correspondientes a cada caso de uso. Se obtuvo el modelo de datos, lo que facilitó el diseño de la base de datos. Se determinó el patrón arquitectónico a seguir y para el diseño de la aplicación informática se hará uso de los patrones GRASP y GoF.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se desarrolla la fase de implementación del sistema, donde se generan artefactos tales como el diagrama de componentes y el modelo de despliegue. Se muestran también los estándares de codificación para garantizar una mejor estructura del código y se realizaran pruebas para garantizar la flexibilidad del *software*.

4.1 Modelo de Implementación

El objetivo del modelo de implementación, es traducir el modelo de diseño en diferentes componentes ejecutables de la aplicación a desarrollar. El modelo de implementación está conformado por el diagrama de componentes y el modelo de despliegue, describiendo como los elementos del modelo de diseño se implementan en términos de componentes, ficheros de código fuente y ejecutables. En el presente epígrafe se evidencian los artefactos del modelo de implementación asociados a la realización del CU Actualizar Centros Informantes.

4.1.1 Diagrama de Componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes *software*, sean estos componentes de código fuente, binarios o ejecutables. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. En cuanto a los componentes, sólo aparecen tipos de componentes, ya que las instancias específicas de cada tipo se encuentran en el diagrama de despliegue (19). Esta descripción brinda gran utilidad a la hora de implementar el sistema, facilitando la organización del trabajo haciéndolo más entendible a los desarrolladores. A continuación se muestra los distintos paquetes relacionados al patrón arquitectónico utilizado MVC, los componentes por los que están compuestos, así como los subsistemas con los que se relacionan.

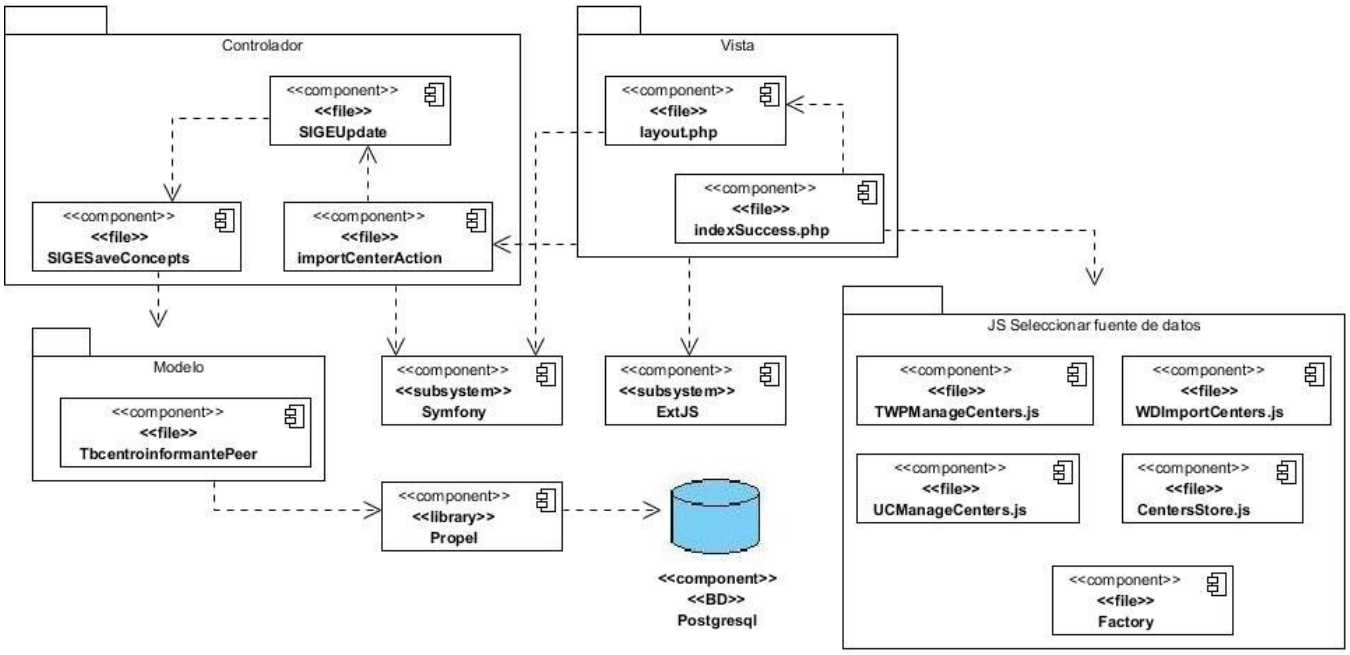


Figura 18. Diagrama de Componentes correspondiente al CU "Actualizar Centros Informantes".

4.1.2 Modelo de Despliegue

El modelo de despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de *hardware* y otras informaciones relacionadas al despliegue del sistema propuesto (21).

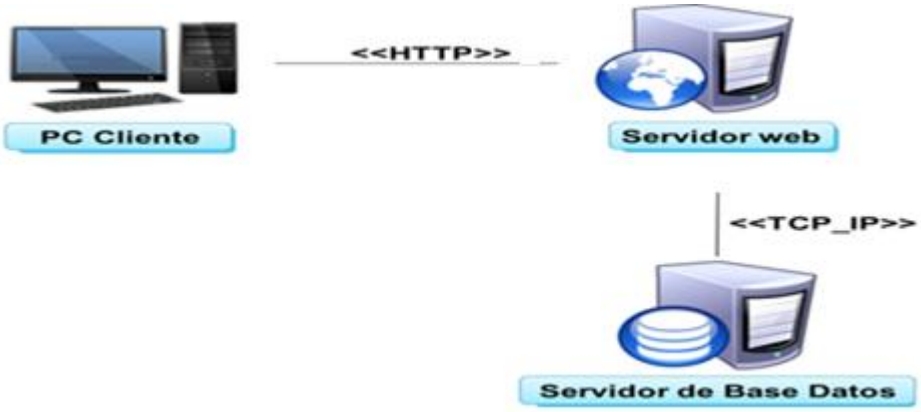


Figura 19. Diagrama de Despliegue.

PC Cliente: Se refiere a las estaciones de trabajo que el usuario utilizará para acceder a la aplicación web y transcribir sus datos.
Servidor Web: Utiliza Apache en su versión 2.2.22 para la publicación de la aplicación desarrollada; y para lograr la conexión del sistema con la PC Cliente se utiliza el protocolo HTTP. Es la herramienta principal para ejecutar la lógica de negocio en el lado del servidor.

Conexión HTTP: Es el protocolo utilizado entre los *browser* de los clientes y el servidor web. Este elemento de la arquitectura representa un tipo de comunicación no orientado a la conexión entre clientes y servidor.

Conexión TCP/IP: Es la base del Internet que sirve para enlazar computadoras. El protocolo TCP/IP es utilizado para establecer la conexión entre el servidor de aplicación y el servidor de base de datos.
Servidor de Base de Datos: Se refiere a la base de datos central la cual es soportada por SIGE. El servidor de base de datos elegido es PostgreSQL, el cual es multiplataforma.

4.2 Código Fuente

Es un texto que se ha escrito en un lenguaje de programación concreto, y que sólo puede ser leído por un experto o programador que tenga los conocimientos.

4.2.1 Estándares de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo agradable, como si un único programador hubiera escrito todo el código de una sola vez. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software (22).

Estilos de codificación empleados
Durante la implementación de la aplicación informática, fue necesario utilizar algunos estilos de codificación en busca de un estándar que aportara una mayor organización. Los estilos empleados se utilizaron con el propósito de estandarizar las nomenclaturas en la implementación del sistema y obtener un producto legible y estructurado.

Identación, llaves de apertura y cierre, y tamaño de las líneas

Se debe usar la identación sin tabulaciones, con un equivalente a 4 espacios. El uso de las llaves "{}" será en una nueva línea. La longitud de las líneas de código es aproximadamente de 75-80 caracteres, para mantener la legibilidad del código.

```
53      public static function saveRecords($recordData, &$existen, &$importados) {
54          $codregistro = $recordData[0];
55          $nombredescriptivo = $recordData[1];
56          $alias = $recordData[2];
57          $existe = RegistroPeer::ExisteRegistro($recordData);
58          $record = RegistroPeer::retrieveByPK($codregistro);
59          if (!$record) {
60              $record = new Registro();
61              array_push($importados, true);
62          } else if (!$existe) {
63              array_push($existen, true);
64          }
65          $record->setCodregistro($codregistro);
66          $record->setNombredescriptivo($nombredescriptivo);
67          $record->setAlias($alias);
68          $record->save();
69      }
```

Figura 20. Identación y Llaves.

Convención de nomenclatura

Los nombres de variables y métodos estarán escritos con el estándar "lowerCamelCase", el cual es uno de los dos tipos existentes de CamelCase. Este es un estilo de escritura que se aplica a frases o palabras compuestas donde la primera letra empieza con minúscula.

```
157     public static function saveAsciateIndicators($recordData, &$existen) {...}
```

Figura 21. Variables y métodos con el estándar lowerCamelCase.

Estructuras de control.

Para lograr una mejor legibilidad del código y facilitar su entendimiento, todas las etiquetas PHP deben ser completas (<?php ?>)... no reducidas (<? ?>).

```
1    <?php
2
3  + class SIGEUpdate {...}
84
85   ?>
```

Figura 22. Etiqueta de PHP escritas de forma completa.

4.2.2 Ejemplos de Código Fuente

De los métodos implementados, su mayoría responden a funcionalidades asociadas a actualizar los datos primarios de SIGE, que se simplifican mediante el uso que hace Symfony de Propel para realizar el mapeo objeto-relacional. Este genera la mayor parte del código de acceso a datos, proporcionando una abstracción, al punto que permite que los implementadores tengan que utilizar muy pocas consultas a la base de datos.

Por ejemplo el método importCenterAction el cual se encarga de mostrar los datos primarios de los ficheros Excel, este solo se ejecutará cuando el usuario desee importar un determinado fichero Excel.

```
class importCentersAction extends sfAction {
  public function execute($request) {
     try {
            $filename = $request->getParameter('filename');
            $path = sfConfig::get('sf_web_dir') . '/download/' . $filename;
            $this->getRequest()->moveFile('file', $path);
            chmod(sfConfig::get('sf_web_dir') . '/download/' . $filename, 0777);

            $user = UserPeer::retrieveByPK($this->getUser()->getAttribute('id'));
            if (!$user->permissionToCreate('manage_centers')) {
                return $this->renderText("{success:false, errors:{message:
                 'Acceso denegado para crear unidades de observaci&oacute;n.'}}");
            }
            $connection = Propel::getConnection();
            $connection->begin();
            $sigeupdate = new SIGEUpdate();
            $user = UserPeer::retrieveByPK($this->getUser()->getAttribute('id'));
            $sigeupdate->save($filename,$user);
            $connection->commit();
            return $this->renderText("{success:true}");
        } catch (Exception $e) {
            $connection->rollback();
            if ($e->getCode() == 1)
                return $this->renderText("{success:false, errors:{message:'" . $e->getMessage() . "'}}");
            else {
                $error = new Error("{success:false, errors:{message:
                'Imposible importar la actualizaci&oacute;n de las unidades de observaci&oacute;n.'}}", 1, $e);
                return $this->renderText($error->getMessage());
            }
        }
  }
```

Figura 23. Código Fuente de importCentersAction.

4.2.3 Interfaces Principales de la Aplicación

El diseño de la aplicación se enfocó a lograr interfaces sencillas y agradables para el cliente. Para ello se utilizó el *framework* de presentación ExtJS que permite a los usuarios familiarizarse con el uso de la aplicación rápidamente.

A continuación se muestras algunas interfaces gráficas de la aplicación informática, específicamente las asociadas a los CU Autenticar Usuario y el CU Actualizar Indicadores.

Figura 24. Interface Autenticar Usuario.

Esta es la interfaz inicial de la aplicación donde el usuario deberá registrarse con un nombre de usuario y contraseña para luego acceder al sistema.

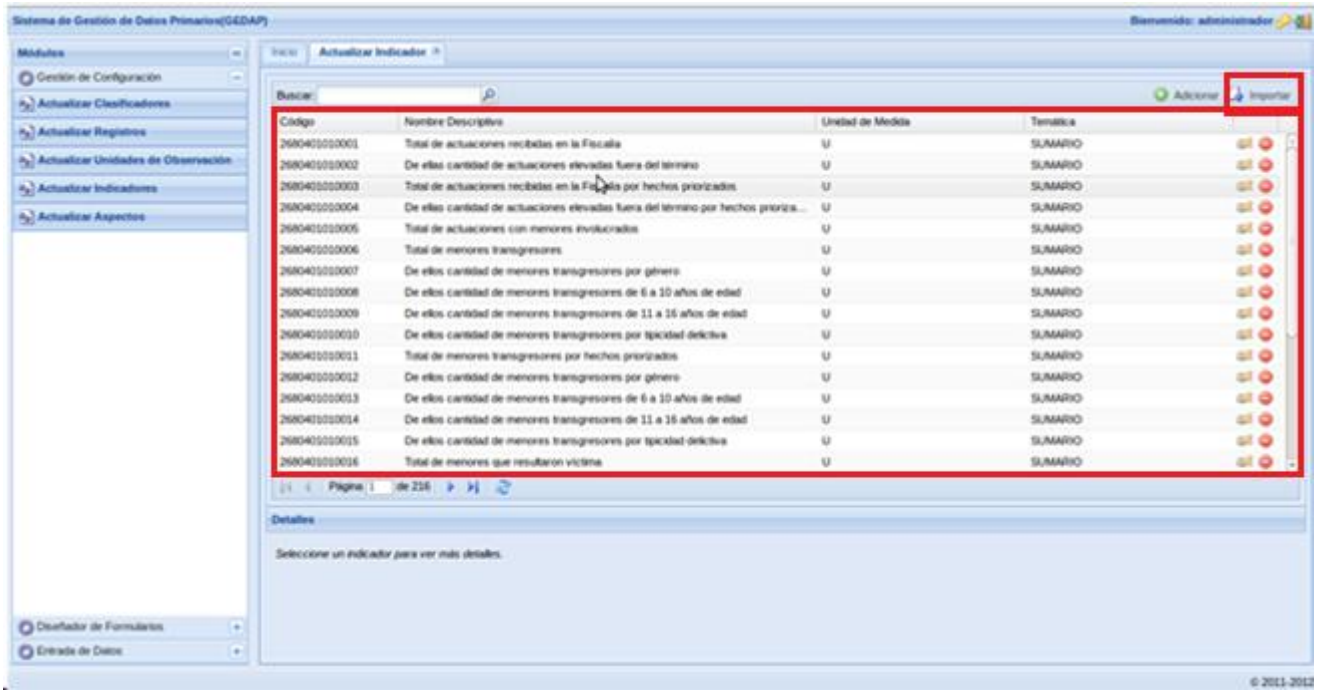


Figura 25. Interface Actualizar Indicadores.

En la figura se muestra el listado de los Indicadores importados en la aplicación informática a través del botón “Importar”.

4.3 Pruebas de *Software*

Las pruebas son un conjunto de actividades en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente (23). Estas se planean con anticipación y se realizan de manera sistemática.

Una estrategia de prueba del *software* proporciona una descripción o planificación bien detallada de los pasos a llevar a cabo como parte de la prueba que da como resultado una correcta construcción del *software*. Esta debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código fuente se han implementado correctamente, así como pruebas de alto nivel que validen las principales funciones del sistema frente a los requerimientos del cliente (23).

Tabla 4. Estrategia de prueba definida para la solución.

Nivel	Tipo de prueba	Método de prueba	Técnica de prueba
Unidad	Funcional	Caja blanca	Camino básico
Sistema	Rendimiento (Benchmark)	Caja negra	Automática
Desarrollador	Funcional	Caja negra	Partición Equivalente

4.3.1 Pruebas a Nivel de Unidad

La prueba de unidad se concentra en el esfuerzo de verificación de la unidad más pequeña de diseño del *software*: el componente o módulo del *software* (24). Estas pruebas se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente y prueban importantes caminos de control para descubrir errores dentro de esos límites.

Se aplican pruebas funcionales como tipo de prueba ya que son las encargadas de probar y validar que el *software* hace lo que se ha especificado. Estas pruebas desarrolladas a nivel de unidad fijan su atención en la validación de funciones y métodos que responden a requisitos del sistema.

La prueba de unidad está orientada al método de caja blanca, este es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba (24). Al emplear los métodos de pruebas de caja blanca, el ingeniero de software podrá derivar casos de prueba que 1) garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez, 2) ejerciten los lados verdaderos y falsos de todas las decisiones lógicas, 3) ejecuten todos lo bucles en sus límites y dentro de sus límites operacionales, y 4) ejerciten estructuras de datos internos para asegurar su validez. Una de las técnicas utilizadas para realizar pruebas de caja blanca es la técnica de la ruta o camino básico.

El método de la ruta básica permite que el diseñador de casos de prueba obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba (24).

Los pasos que se siguen para aplicar esta técnica son:

1) A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2) Se calcula la complejidad ciclomática del grafo.
3) Se determina un conjunto básico de caminos independientes.
4) Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Resultados de la Prueba Unitaria

Se realizaron cuatro iteraciones de pruebas unitarias y se diseñaron casos de prueba para cada uno de los caminos independientes que arrojaron las pruebas de caja blanca, identificando un total de 14 No Conformidades (NC). En la primera iteración se detectaron 8 NC, de ellas seis de lógica algorítmica y dos relacionadas con mensajes de información incorrectos a los usuarios. En la segunda iteración se detectaron 4 NC, de ellas tres de lógica algorítmica y una de mensaje de información incorrecto a los usuarios. En la tercera iteración se detectaron dos NC, ambas de lógica algorítmica. En la cuarta iteración se verificó que todas las NC encontradas con anterioridad se habían resuelto satisfactoriamente y no se detectaron nuevas NC. A continuación se muestra un ejemplo de prueba de caja blanca realizada al CU Importar Centros Informantes.

Tabla 5. Ejemplo de Prueba de Caja Blanca.

Prueba estructural de caja blanca	**CU Importar Centros Informantes**
Probador: Koisam Rodríguez Villamil.	

Código al que se aplica:
public function save($filename,$user) {
1 try {
2 $objPHPExcel = PHPExcel_IOFactory::load(sfConfig::get('sf_web_dir').'/download/'.$filename);
3 } catch (Exception $exc) {
4 if (file_exists(sfConfig::get('sf_web_dir').'/download/'.$filename)) {
5 unlink(sfConfig::get('sf_web_dir').'/download/'.$filename);
}
6 throw new Exception("Imposible realizar operación: El archivo que desea importar no presenta el formato correcto.", 1);
}
7 $flaghead = true;
8 $objWorksheet = $objPHPExcel->getActiveSheet();	

```
 9      foreach ($objWorksheet->getRowIterator() as $row) {
10          if ($flaghead == true) {
11              $this->check($row);
12              $flaghead = false;
13            } else {
13              $record = $this->getRow($row, false);
14              SIGESaveConcepts::saveOU($record[0],$user);
              }
          }
15      unlink(sfConfig::get('sf_web_dir').'/download/'.$filename);
16  }
```

Complejidad ciclomática:

1. V (G) = (A - N) +2.
2. V (G) = NP + 1.
3. V (G) = R + 1.

Dónde:
- A: aristas.
- R: regiones cerradas.
- N: nodos.
- NP: nodos predicados.

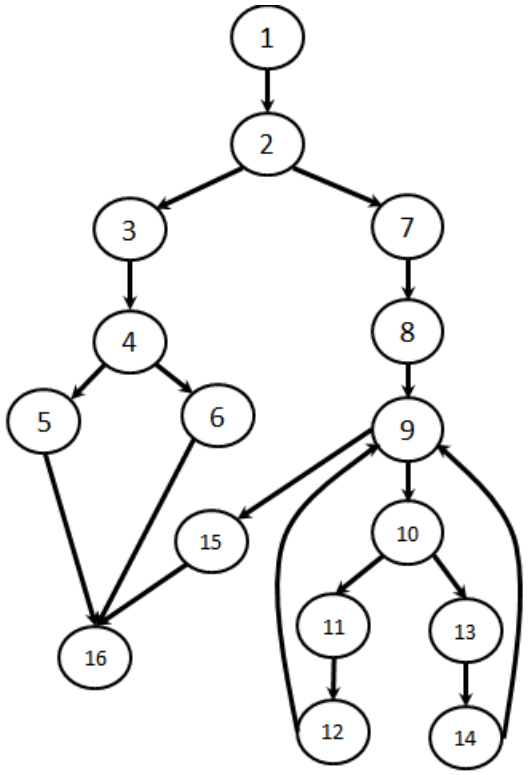
Sustituyendo los valores se obtiene que:

1. V(G)= (19-16)+2 = 5
2. V(G)= 4+1 = 5
3. V(G)= 5

Caminos independientes:

1) 1–2–3–4–5–16
2) 1–2–3–4–6–16
3) 1–2–7–8–9–15–16
4) 1–2–7–8–9–10–11–12–9–15–16
5) 1–2–7–8–9–10–13–14–9–15–16

Representación del grafo:



Caso de prueba para el camino básico 2

Descripción: Al seleccionar un fichero Excel, se trata de importar el mismo con un formato incorrecto.

Condición de ejecución: Se debe seleccionar un fichero con un formato incorrecto.	
Datos de entrada:	Fichero Excel estructurado incorrectamente.
Resultado esperado:	Imposible realizar operación. El archivo que desea importar no presenta el formato correcto.
Evaluación del caso de prueba :	Satisfactorio

En la figura 26 se muestra el número de iteraciones realizadas y la cantidad de NC detectadas en cada iteración. Estas pruebas permitieron ejecutar al menos en una ocasión cada sentencia del programa obteniendo resultados satisfactorios.

Figura 26. Gráfica de iteraciones de pruebas funcionales a nivel de unidad.

4.3.2 Pruebas a Nivel de Sistema

Las pruebas a nivel de sistema se hacen cuando el *software* está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de *software* y *hardware* han sido integrados (25).

Para darle cumplimiento al objetivo general en el cual la recolección de datos primarios demora y complejiza la actualización de la información, se establecen dos escenarios:

1: Proceso de inserción y actualización de datos primarios en SIGE.
2: Proceso de inserción y actualización de datos primarios en la solución propuesta.

Luego de analizar las diferentes pruebas de rendimiento existentes, se decide aplicar el tipo de prueba *Benchmark* o comparativa, que permite comparar el rendimiento de un elemento nuevo o desconocido con uno de carga de trabajo de referencia conocido. Un *benchmark* es el resultado de la ejecución de un programa informático que tiene como objetivo estimar el rendimiento de un elemento concreto, y poder comparar los resultados con otros similares. La elección de las condiciones bajo la cual dos sistemas distintos pueden compararse entre sí es variable. La condición para el análisis de los sistemas SIGE y la solución propuesta se centra en el tiempo de ejecución de los procesos que realizan.

Como método de prueba para realizar la comparación entre los dos sistemas se definen las pruebas de caja negra. Estas se concentran en los requisitos funcionales del *software* permitiendo al ingeniero de *software* derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa (26).

Realización de las Pruebas de Rendimiento

Teniendo en cuenta los elementos seleccionados para la comparación se tendrán en cuenta las secciones de pruebas en las que las acciones se realicen correctamente, enfocadas específicamente en el proceso de actualización de datos primarios de ambos sistemas. El objetivo fundamental que se persigue con las pruebas de rendimiento es lograr comparar los resultados obtenidos en los escenarios identificados, dando así respuesta al cumplimiento o no del problema planteado.

Rendimiento en SIGE del Proceso de Actualización de Datos Primarios

El tiempo de realización de la actualización de datos primarios de SIGE, será determinado por el tiempo que demora el usuario en la inserción de la información que caracteriza a cada dato primario más el tiempo de ejecución que demora del sistema en responder las peticiones del propio usuario.

Variables

Las variables definidas para las pruebas de rendimiento son:

TD: Tiempo que demora el usuario en la inserción de la información que caracteriza a cada dato primario.
TE: Tiempo de ejecución que demora del sistema en responder las peticiones del propio usuario.
TADP: Tiempo de realización de la actualización de datos primarios de SIGE.

Ecuación

TADP = TD + TE.

Se toma como caso de estudio el CU Actualizar Centros Informantes para calcular el tiempo de ejecución que demora el sistema en responder las peticiones del usuario, se evidencia al realizar dicha prueba a la funcionalidad Actualizar Centros Informantes. Las peticiones que hace el usuario al sistema en este CU, son las relacionadas con adicionar manualmente la información de un centro informante. En la figura 27 se muestran los resultados obtenidos haciendo uso de la herramienta JMeter 2.9.

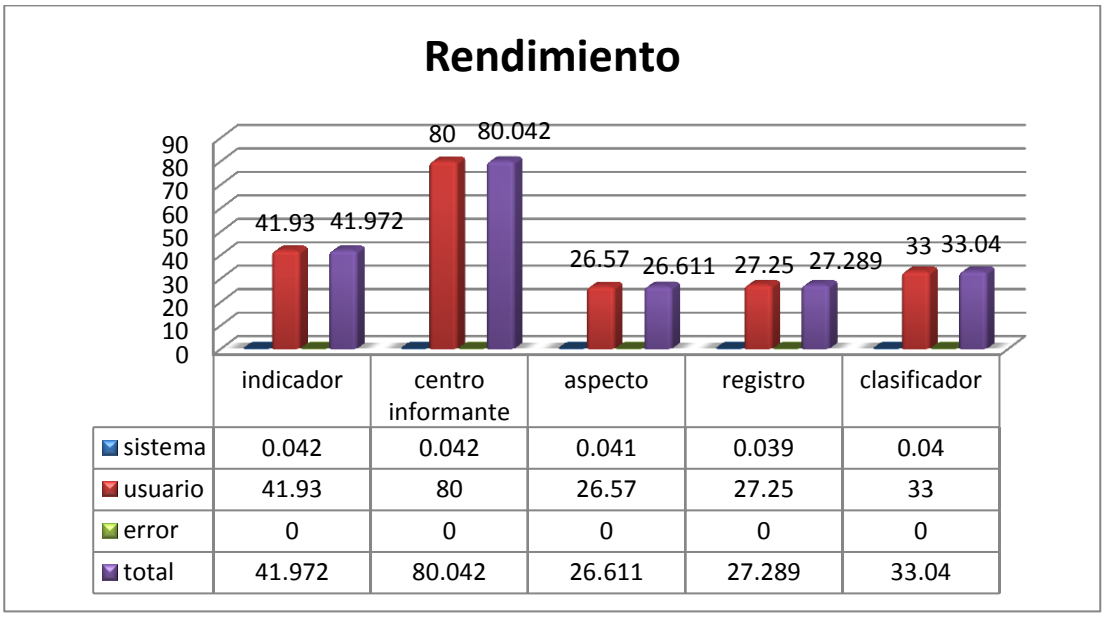
Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/records_classifiers.php/manage_records/listRecords	2	48	41	55	41	55	0,00%	10,3/sec	4,8
/records_classifiers.php/manage_centers/list0tes	1	41	41	41	41	41	0,00%	24,4/sec	11,5
/records_classifiers.php/manage_centers/list0mes	1	37	37	37	37	37	0,00%	27,0/sec	12,7
/records_classifiers.php/manage_centers/saveCenter	1	40	40	40	40	40	0,00%	25,0/sec	11,8
/records_classifiers.php/manage_centers/listCentersPager	1	40	40	40	40	40	0,00%	25,0/sec	11,8
Total	6	42	40	41	37	55	0,00%	20,7/sec	9,7

Figura 27. Resultados de pruebas de rendimiento del CU Actualizar Centros Informantes de SIGE.

Se comprobó TE fue de 0.042 segundos, con un promedio de error de 0%, logrando el sistema un rendimiento de 20.7 peticiones por segundo.

Como parte del TD, específicamente de un Centro Informante, se calculó en tiempo real un estimado de 80 segundos, alcanzándose un TADP de 80.04 segundos en adicionar un Centro Informante al sistema.

De la misma manera se realizaron las pruebas para los restantes datos primarios. Los tiempos están asociados a la adición manual de un solo ejemplar. Estos tiempos se encuentran registrados en la figura 28



	indicador	centro informante	aspecto	registro	clasificador
sistema	0.042	0.042	0.041	0.039	0.04
usuario	41.93	80	26.57	27.25	33
error	0	0	0	0	0
total	41.972	80.042	26.611	27.289	33.04

Figura 28. Resultados de pruebas de rendimiento a SIGE.

A partir del análisis anterior, se realizó un estimado del tiempo que demoran los usuarios de SIGE, en actualizar datos primarios para diferentes volúmenes de datos. Ver figura 29:

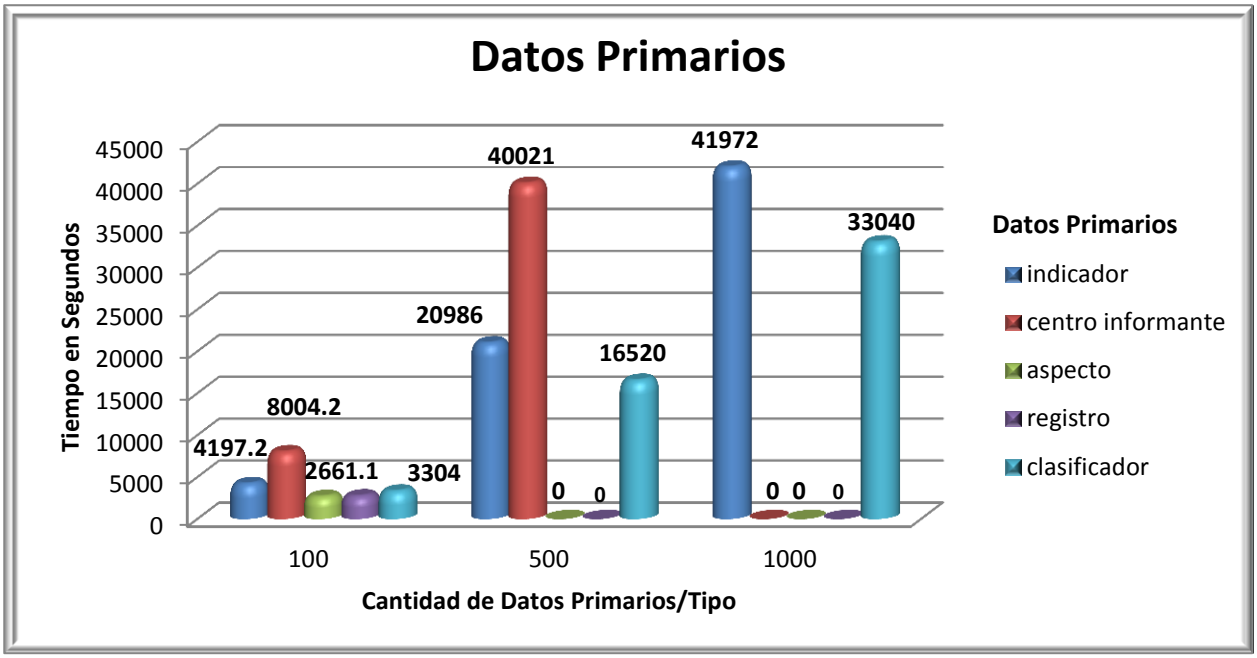


Figura 29. Resultados de pruebas de rendimiento a SIGE con volúmenes de datos variables.

Con el objetivo de realizar una interpretación más exhaustiva de los resultados mostrados en la figura anterior se analizan específicamente los Indicadores, por ser el dato primario que ofrece mayor cantidad de información.

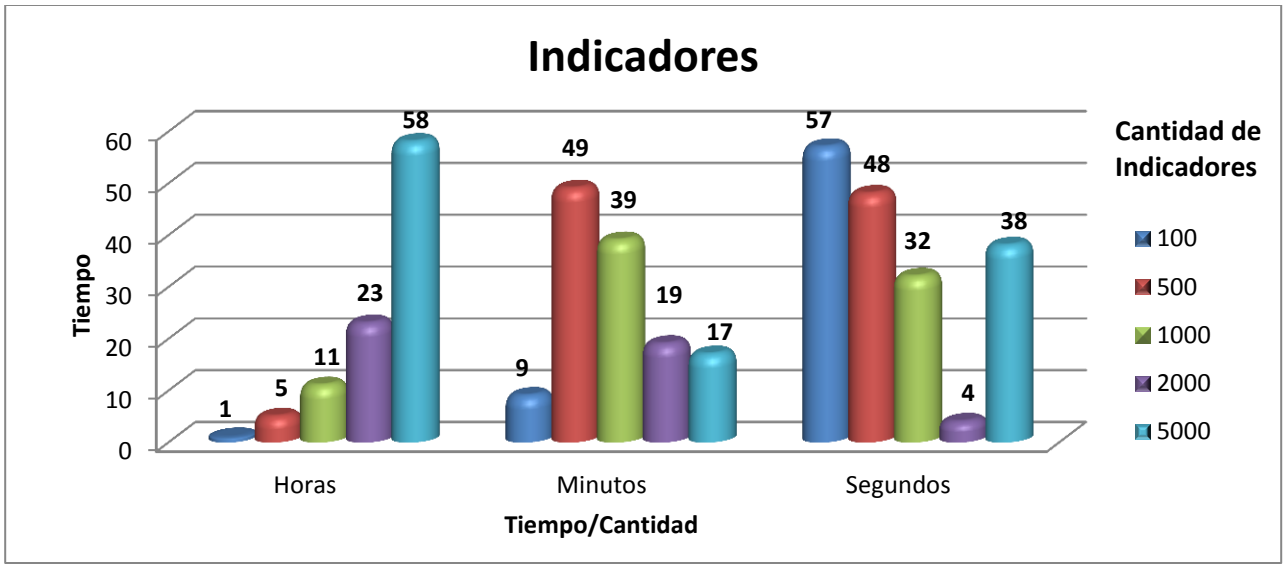


Figura 30. Resultados de pruebas de rendimiento en la adición de Indicadores.

Por ejemplo para una cantidad de 2000 adiciones manuales se obtuvo una demora en el SIGE de 83944 segundos para la actualización de estos indicadores. Ello equivale a 23 horas, 19 min y aproximadamente 4 segundos.

Rendimiento de la aplicación informática desarrollada

El tiempo de realización durante la actualización de datos primarios de la aplicación informática, será determinado por la demora del usuario en seleccionar el documento Excel a importar más el tiempo de ejecución que demora el sistema en responder las peticiones del usuario.

Variables

Las variables definidas para las pruebas de rendimiento son:

TADPA: Tiempo de realización durante la actualización de datos primarios de la aplicación informática.
TDI: Tiempo que demora el usuario en seleccionar el documento Excel a importar.
TES: Tiempo de ejecución que demora el sistema en responder las peticiones del usuario.

Ecuación

TADPA = TDI + TES.

Una vez realizadas las pruebas para verificar el correcto funcionamiento de la actualización de los datos primarios en la aplicación informática. Se obtuvieron los TADPA para una cantidad a importar de 100 datos primarios de cada tipo, mostrándose los TES en la figura 31:

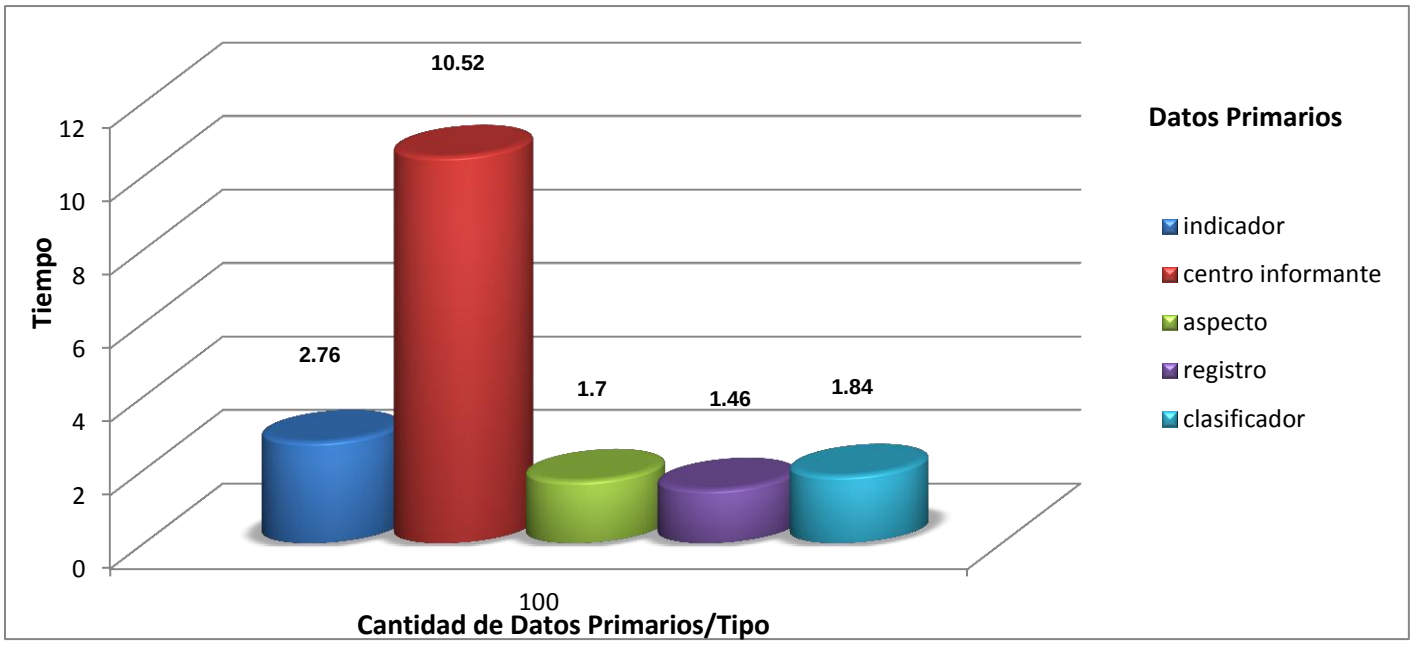


Figura 31. Resultados de pruebas de rendimiento al importar 100 datos primarios.

Con el objetivo de probar el correcto funcionamiento de los datos importados en la aplicación informática, se hicieron además pruebas para otras cantidades de datos primarios a importar, obteniéndose así los TADPA, mostrándose los TES en la siguiente figura:

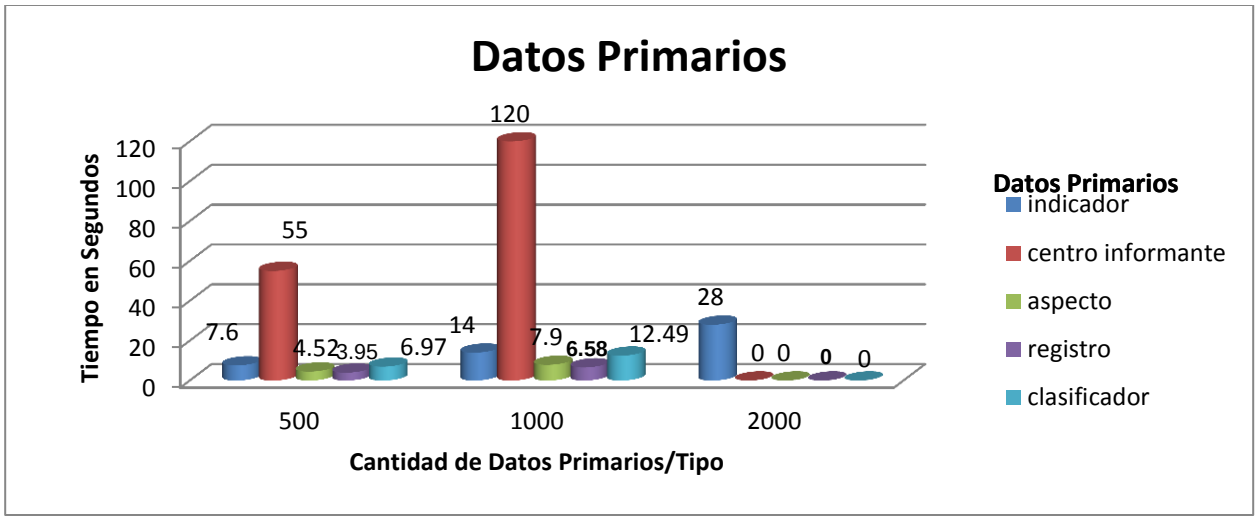


Figura 32. Resultados de pruebas de rendimiento al importar datos primarios.

Comparación del rendimiento de SIGE y la aplicación informática

Luego de haberse realizado las pruebas para comprobar el proceso de actualización de datos de SIGE y la aplicación informática desarrollada aplicando el tipo de prueba *Benchmark*, se obtuvieron resultados satisfactorios partiendo de los escenarios definidos. Estos resultados se evidencian cuando aplicando una muestra de 2000 indicadores a insertar, un usuario de SIGE demora alrededor de 83944 segundos en adicionar cuyos datos al sistema, mientras que la solución propuesta demora solamente 28 segundos en importar estos 2000 indicadores al sistema, resaltando una diferencia significativa entre el tiempo de inserción de los datos de la aplicación informática desarrollada respecto a SIGE en su versión actual.

4.3.3 Pruebas a Nivel de Desarrollador

Las pruebas de desarrollador son aquellas que son implementadas por el equipo de desarrollo y son consideradas solo para las pruebas de unidad.

Otro tipo de prueba que se le aplicará a la aplicación informática vendrá dada por las pruebas funcionales, donde pueden estar implicadas una o varias clases y la propia interfaz de usuario. El objetivo de un caso de prueba es detectar los posibles errores cuando se introducen clases de datos, además de comprender como el sistema guía el proceso y el resultado esperado. Los casos de usos tienen asociados un caso de prueba que se dividen en secciones, escenarios y en este se recogen las especificaciones de dicho caso de uso. A través de los escenarios se conforman matrices de datos, mediante las cuales se pueden

observar el uso de valores válidos e inválidos evidenciándose la técnica de partición equivalente que corresponde al método de prueba de caja negra.

Ejecución de las pruebas funcionales

Se realizaron cuatro iteraciones de pruebas funcionales y se aplicaron los 11 casos de prueba diseñados arrojando un total de 12 No Conformidades (NC). En la primera iteración se detectaron 5 NC, de ellas tres de código y dos de interfaz. En la segunda iteración se detectaron 3 NC, de ellas dos de código y una de ortografía. En la tercera iteración se detectaron 2 NC, de ellas una de interfaz y una de ortografía. En la cuarta iteración se comprobó que las NC de las iteraciones anteriores estuvieran resueltas y no se detectaron nuevas NC. Estas pruebas permitieron comprobar el correcto funcionamiento de la aplicación informática y la correcta validación de los campos.

Conclusiones Parciales

En el presente capítulo a partir de la elaboración del diagrama de despliegue se identificaron los recursos necesarios para el correcto funcionamiento y despliegue de la solución además de los diferentes diagramas de componentes vinculados a cada caso de uso. Con el uso de los estándares de codificación se logró que el código fuera verificable y comprensible. Mediante la realización de las pruebas se pudo confirmar que el sistema cumpliera con las funcionalidades requeridas. Además se realizó una comparación entre los sistemas SIGE y la aplicación informática de acuerdo a su rendimiento y velocidad de actualización de información permitiendo obtener resultados que demuestran las potencialidades de la solución desarrollada.

CONCLUSIONES GENERALES

La investigación tuvo como propósito desarrollar una aplicación informática para la recolección de los datos primarios de los Centros Informantes del Sistema Integrado de Gestión Estadística, de esta se arriban a las siguientes conclusiones:

- El estudio exhaustivo realizado durante la investigación posibilitó examinar de forma detallada el comportamiento de SIGE, analizando cómo se lleva a cabo el proceso de inserción y actualización de la información gestionada en el sistema obteniéndose un entendimiento del negocio y de la problemática planteada.
- Se seleccionaron para el desarrollo del sistema las tecnologías y herramientas que más se adecuaban a las características propias de la solución y que propiciaron que esta responda a las políticas establecidas por el centro DATEC.
- Se desarrolló una aplicación que permite informatizar el proceso de captura de información de SIGE, evitando demoras en la aplicación de dicho proceso que afectan el desempeño del sistema y de sus usuarios.
- Luego de realizar la implementación se realizaron las pruebas correspondientes que validaron la aplicación informática, donde se identificaron un conjunto de no conformidades las cuales fueron resueltas, permitiendo la aceptación final del sistema por parte del cliente.

RECOMENDACIONES

1. Extender la funcionalidad de asociar datos a un formulario, para que no solo permita la importación de indicadores y aspectos sino también reglas de validación.
2. Implementar en el sistema los requisitos de seguridad establecidos por el MININT y que incorpora ya el Sistema Integrado de Gestión Estadística v2.0.

REFERENCIAS BIBLIOGRÁFICAS

1. TIC. [En línea] http://edutec.rediris.es/Revelec2/revelec21/jmontero.htm.

2. **Rosada, Alexander Rodríguez.** ONEI. Oficina Nacional de Estadística e Información . [En línea] 2006. [Citado el: 20 de 10 de 2013.] http://www.cubagob.cu/otras_info/estadisticas.html.

3. **Sanz, Claudia Nuñez.** Sistema de Captura de Información para la Toma de Decisiones . [En línea] [Citado el: 20 de 10 de 2013.]

4. **Lidia Pérez, Nara, Nuñez Sanz, Claudia y Monné Roque, Diana.** *INFORMATIZACIÓN DE LOS PROCESOS DE GESTIÓN ESTADÍSTICA EN CUBA.*

5. ONEI. [En línea] http://www.one.cu/nomencladores.htm.

6. **Ríos Salgado, Santiago, Hinojosa Raza, Cecilia y Delgado Rodíguez, Ramiro.** APLICACIÓN DE LA METODOLOGIA OPENUP . [En línea] http://repositorio.espe.edu.ec/bitstream/21000/6316/1/AC-SISTEMAS-ESPE-047042.pdf.

7. *Visual Paradigm for UML.*

8. Base de Datos. [En línea] http://foxeando.com.ar/archivos/base_de_datos.pdf.

9. **Martinez, Rafael.** PostgreSQL-es. [En línea]] http://www.postgresql.org.es/sobre_postgresql.

10. Definición. [En línea] 2008. http://definicion.de/lenguaje-de-programacion.

11. Lenguajes del lado del servidor y del cliente. [En línea] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

12. Glosario. [En línea] http://www.glosariodigital.com/termino/framework/.

13. Symfony en pocas palabras. [En línea] http://librosweb.es/symfony_1_4/capitulo_1/symfony_en_pocas_palabras.html.

14. Entrenamiento Profesional en la Tecnología Java. ¿Que es un IDE? [En línea] 26 de 01 de 2012. http://globalmentoring.com.mx/cursos-java/java-fundamentos/que-es-un-ide.

15. NetBeans . [En línea] 2013. http://netbeans.org/index_es.html.

16. **Larman, Craig.** Modelo de Dominio. *UML y Patrones. 2da Edición.* s.l. : Prentice Hall, 2003.

17. Requisitos. [En línea] http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf.

18. **Reynoso, Carlos y Reynoso, Kicillof.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft . [En línea] 2004. http://carlosreynoso.com.ar/archivos/arquitectura/Estilos.PDF.

19. **Aguero, Jorge, y otros.** *Base de Datos y UML.* 2002.

20. Modelo de paquetes. [En línea] http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

21.SparxSystems.[Enlínea]http://www.sparxsystems.com.ar/resources/tutorial/physical_models.html.

22.Developer Network. [En línea] 2014. http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

23. PRUEBASDESOFTWARE. [En línea] 2005.

24. **Pressman, Roger Steven.** Pressman_Cap_14_Estrategia_de_Prueba_Parte_1.

25. *Flujo de trabajo Prueba.* **Software, Departamento de Ingeniería de.** 2007.

26. **Pressman, Roger Steven.** Pressman_Cap_13_Técnicas_de_Prueba.

BIBLIOGRAFÍAS

1. TIC. [En línea] http://edutec.rediris.es/Revelec2/revelec21/jmontero.htm.

2. **Rosada, Alexander Rodríguez.** ONEI. Oficina Nacional de Estadística e Información . [En línea] 2006. [Citado el: 20 de 10 de 2013.] http://www.cubagob.cu/otras_info/estadisticas.html.

3. **Sanz, Claudia Nuñez.** Sistema de Captura de Información para la Toma de Decisiones . [En línea] [Citado el: 20 de 10 de 2013.]

4. **Lidia Pérez, Nara, Nuñez Sanz, Claudia y Monné Roque, Diana.** *INFORMATIZACIÓN DE LOS PROCESOS DE GESTIÓN ESTADÍSTICA EN CUBA.*

5. ONEI. [En línea] http://www.one.cu/nomencladores.htm.

6. **Ríos Salgado, Santiago, Hinojosa Raza, Cecilia y Delgado Rodíguez, Ramiro.** APLICACIÓN DE LA METODOLOGIA OPENUP . [En línea] http://repositorio.espe.edu.ec/bitstream/21000/6316/1/AC-SISTEMAS-ESPE-047042.pdf.

7. *Visual Paradigm for UML.*

8. Base de Datos. [En línea] http://foxeando.com.ar/archivos/base_de_datos.pdf.

9. **Martinez, Rafael.** PostgreSQL-es. [En línea]] http://www.postgresql.org.es/sobre_postgresql.

10. Definición. [En línea] 2008. http://definicion.de/lenguaje-de-programacion.

11. Lenguajes del lado del servidor y del cliente. [En línea] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

12. Glosario. [En línea] http://www.glosariodigital.com/termino/framework/.

13. Symfony en pocas palabras. [En línea] http://librosweb.es/symfony_1_4/capitulo_1/symfony_en_pocas_palabras.html.

14. Entrenamiento Profesional en la Tecnología Java. ¿Que es un IDE? [En línea] 26 de 01 de 2012. http://globalmentoring.com.mx/cursos-java/java-fundamentos/que-es-un-ide.

15. NetBeans . [En línea] 2013. http://netbeans.org/index_es.html.

16. **Larman, Craig.** Modelo de Dominio. *UML y Patrones. 2da Edición.* s.l. : Prentice Hall, 2003.

17. Requisitos. [En línea] http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf.

18. **Reynoso, Carlos y Reynoso, Kicillof.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft . [En línea] 2004. http://carlosreynoso.com.ar/archivos/arquitectura/Estilos.PDF.

19. **Aguero, Jorge, y otros.** *Base de Datos y UML.* 2002.

20. Modelo de paquetes. [En línea] http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

21.SparxSystems.[Enlínea]http://www.sparxsystems.com.ar/resources/tutorial/physical_models.html.

22.Developer Network. [En línea] 2014. http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

23. PRUEBASDESOFTWARE. [En línea] 2005.

24. **Pressman, Roger Steven.** Pressman_Cap_14_Estrategia_de_Prueba_Parte_1.

25. *Flujo de trabajo Prueba.* **Software, Departamento de Ingeniería de.** 2007.

26. **Pressman, Roger Steven.** Pressman_Cap_13_Técnicas_de_Prueba.

27. IDE, Entrenamiento Profesional en la Tecnología Java. ¿Que es un IDE? [En línea] 26 de enero de 2012. [Citado el: 23 de enero de 2013.] http://globalmentoring.com.mx/cursos-java/java-fundamentos/que-es-un-ide.

28. **Sparks, Geoffrey**. El Modelo de Componentes. [En Línea] http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r73575.PDF.

29. **López, Patricia**. Lenguaje Unificado de Modelado – UML. [En Línea]. http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/materiales-de-clase-1/is1-t02-trans.pdf.

30. **Barrios, Judith**. Ingeniería de Software Orientada a Objetos [En Línea]. http://ceidis.ula.ve/cursos/pgcomp/isoo/tema9.html.

31. **Storti, Guillermo, Ríos, Gladys, Campodónico Gabriel.** Tecnología de la Información y la Comunicación. Base de datos, Modelo Entidad Relación [En Línea] 2007. http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf.

32. **Larman, Craig** Modelo de Diseño: realización de los casos de uso con los patrones GRASP. UML y Patrones. [En Línea]. http://lsi.ugr.es/~mvega/isoo/larman/cap17[1].pdf.