Universidad de las Ciencias Informáticas

Facultad 3

Título: Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Armando Robert Lobo

Tutor: Ing. Yasmany Molina Díaz

Ciudad de la Habana, Junio 2009

DEDICATORIA

Dedico especialmente este trabajo a mi abuelo Luciano Roberto Lobo Pérez que dejó de estar presente entre nosotros hace muchos años, pero que se ha convertido para mí en un ejemplo de revolucionario y de hombre de principios digno a imitar.

A mi abuela Aida A. Ramírez Socas por el amor y los años de su vida que dedicó en educarme.

A mi mamá Mariela Lobo Ramírez y a mi papá Armando Robert Martínez porque sin su amor, constante apoyo y confianza no hubiese llegado jamás hasta aquí.

A Yunia del Toro López por haberme recibido como un hijo, y soportarme durante estos 10 años, ¡y los que faltan…!

A mis tíos Roberto e Ileana por su confianza, apoyo y motivación.

A mis hermanos, Daniel, Roxana, Zayme y Emanuel que sirva mi esfuerzo de motivación.

AGRADECIMIENTOS

A mis profesores todos, aún a los que ya no puedo recordar, especialmente a los profes Alina, Dagmaris y Daniel de Betancourt a quienes debo la pasión por las Ciencias Informáticas que devino en mi vocación profesional.

A Yasmany Molina Díaz por las carreras que ha tenido que dar conmigo, por la confianza que ha depositado en mí, y por sus oportunas indicaciones.

A mi numerosa familia que espera de mi lo mejor, y a quienes no puedo defraudar.

A mis amigos, por su sinceridad, por aguantar mis intervenciones en las interminables reuniones y por estar allí siempre que necesité el apoyo de alguno.

Al pueblo que día a día aún sin saber aporta cada centavo (con esfuerzo, sacrificio y sudor) que se invierte en la educación de sus hijos, para con ellos que son la Patria misma será mi compromiso.

A esta universidad que me ha facilitado una educación y preparación de verdadera excelencia como profesional y como revolucionario.

A la Revolución Socialista por buscar el sueño de igualdad y de un mundo mejor posible.

A Fidel y a la dirección de nuestro Gobierno y Partido por su esfuerzo en construir esta Revolución Socialista.

RESÚMEN

No es posible tomar decisiones si no se posee información, aún la mínima al menos. La estadística adquiere relevancia en tanto hace posible transformar grandes cantidades de datos en información útil a los procesos administrativos como soporte a la toma de decisiones. El presente trabajo de diploma describe la arquitectura del Sistema Integrado de Gestión Estadística Nuragas 2.0 desarrollado de conjunto por el Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD) adscrito a la Infraestructura Productiva de la Universidad de las Ciencias Informáticas y la Oficina Nacional de Estadística de Cuba (ONE). El sistema en su conjunto permite definir qué información captar a través de formularios estadísticos, capturar, validar, almacenar, procesar y consultar dicha información. Se ha pensado como una aplicación web enriquecida, realizada con tecnologías libres, destacándose el uso de AJAX, Linux, Apache, PostgreSQL y PHP como tecnologías subyacentes. Se presenta una arquitectura multiplataforma, robusta, flexible y extensible candidata a convertirse en arquitectura tipo al desarrollo de soluciones empresariales. En tanto el sistema propuesto por su concepción sobrepasa el marco operativo de la ONE y se extiende al de una solución para el apoyo a la toma de decisiones en una plataforma de inteligencia de negocios.

"¿Qué será nuestro país dentro de 40, 50, 100 años? Todos nosotros nos preocupamos por ese país y quisiéramos tener el privilegio de que todo lo que hagamos sea para ese futuro, y que esta Cuba de hoy, que nosotros decimos que es el prototipo de la que soñó Martí al morir, siga siendo (…) si es posible una Cuba mejor que la de hoy y más revolucionaria que hoy. Y eso va a depender de ***nosotros*** *(…)* ***de todos*** *los revolucionarios"*

Fidel Castro Ruz
24 de febrero de 1998

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La estadística nacional es organizada, propuesta y ejecutada por la Oficina Nacional de Estadísticas (ONE), entidad adjunta al Ministerio de Economía y Planificación de la República de Cuba, dado el carácter esencialmente central de la administración del Estado, toda empresa, organismo, organización, o institución identificado como centro informante y poseedor de información de interés para la toma de decisiones de los órganos de la administración del Estado está en la obligación de reportar información estadística bajo convenio con la ONE. Para cumplir con su misión la ONE ha priorizado a partir del 2006 la introducción de las más novedosas técnicas de almacenamiento y procesamiento de datos. En este contexto en octubre de ese mismo año se inició de conjunto con la Universidad de las Ciencias Informáticas (UCI) el desarrollo de una aplicación que soporte el marco metodológico del trabajo estadístico, nace así el Proyecto de Informatización de la ONE que desarrolla el Sistema Integrado para la Gestión Estadística (SIGE) 1.0.

La implantación de este sistema para la ONE supone un ahorro considerable en recursos (sobre todo insumos ofimáticos) un aumento de la eficiencia operativa y un mejoramiento de las condiciones laborales gracias a la automatización de tareas tediosas. La implantación de este sistema es un objetivo estratégico de la ONE para el 2009 y se inserta en el marco de los cambios a fin de elevar la eficiencia de la gestión por parte de la administración del estado a todos los niveles.

SIGE ha suscitado interés por parte de otras entidades como son la Unión de Empresas Militares de las Fuerzas Armadas Revolucionaria, del Ministerio de Salud Pública, y el Ministerio de la Industria Alimenticia. Se encuentra en el catálogo de productos de la UCI y fue registrado como propiedad intelectual de sus desarrolladores a principios del 2009, adicionalmente con el surgimiento del ERP cubano (un sistema para la planeación de recursos empresariales estratégico para la informatización del país) surge la oportunidad de incorporarlo dentro de esta importante solución.

En noviembre de 2008 el equipo del Proyecto de Informatización de la ONE (desarrollador del SIGE) es trasladado para Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD) quien replantea el desarrollo del mismo como parte de sus productos con el condicionamiento de que se realice sobre la web y en tecnologías fundamentalmente libres aprovechando la experiencia y madurez

alcanzada por dicho centro en esta plataforma, integrando al SIGE al Paquete de Apoyo a la Toma de Decisiones en Sistemas Inteligentes (PATDSI). Así comienza un nuevo proyecto a fin de migrar y extender el alcance del sistema anterior.

¿Cómo lograr que los desarrolladores de la nueva versión de este sistema sobre la web y en software libre tengan una idea clara y centrada de cómo debe estructurarse el sistema?

La Arquitectura de Software como disciplina tiene por objetivo definir la organización fundamental de un sistema encarnada en sus componentes las relaciones entre estos y el entorno, y los principios que orientan su diseño y evolución (según plantea el estándar IEEE 1471-2000 al definir la arquitectura de software). El principal producto obtenido en esta disciplina es la especificación formal de la arquitectura a través de vistas que capturan las diferentes perspectivas del sistema que posteriormente servirán de base a cada uno de los implicados para de manera coherente conducir el desarrollo posterior del producto. Es importante tener en consideración desde los momentos iniciales en que se define la arquitectura los atributos de calidad (escalabilidad, extensibilidad, robustez, flexibilidad, etc.) que se desean potenciar en esta en función de estrategias de desarrollos futuros.

SIGE en su versión 1.0 basa su arquitectura sobre .NET utilizando como gestor de bases de datos SQL Server 2000 y el Análisis Manager también de Microsoft, para el análisis avanzado de datos, SIGE es una aplicación de escritorio que entre los requisitos de instalación demanda Windows 98 o superior como sistema operativo para el cliente y Windows Server 2003 o superior como sistema operativo para el servidor. Como se puede observar se trata de tecnología puramente propietaria algo que va en desacuerdo con las líneas directrices para la informatización de la sociedad cubana establecidas por el Ministerio de la Informática y las Comunicaciones. A ello se suma la demanda de recursos necesarios para soportar esta tecnología que no siempre está en correspondencia con las posibilidades de hardware del cliente, las dificultades que como aplicación de escritorio implica su instalación y configuración, y la necesidad de incorporarlo al PATDSI como aporte importante a las posibilidades futuras de este. Por ello se espera por tanto obtener para SIGE 2.0 Nuragas una arquitectura que considere en primer lugar el desarrollo sobre la web en software libre para su integración al PATDSI del CENTALAD, que explote racionalmente los recursos de hardware conforme a las posibilidades de la ONE, que potencie atributos de calidad como robustez, flexibilidad y extensibilidad. En tanto que facilite a los miembros del equipo de desarrollo la información necesaria para desempeñar sus respectivos

roles en la realización del sistema en lo que respecta a las decisiones arquitectónicamente significativas sobre las cuales basarán su trabajo.

En función de lo antes visto se ha identificado el siguiente **Problema Científico**:

¿Cómo diseñar una arquitectura de software que sea multiplataforma, robusta, flexible y extensible para el Sistema Integrado de Gestión Estadística 2.0 Nuragas en su realización para software libre y sobre la web?

Objeto de Estudio: Proceso de desarrollo del Sistema Integrado de Gestión Estadística 2.0 Nuragas.

Campo de Acción: Diseño y descripción arquitectónica para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.

Se persigue con ello el **Objetivo General** de: Diseñar una Arquitectura de Software para el Sistema Integrado de Gestión Estadística Nuragas 2.0 multiplataforma, robusta, flexible y extensible para su desarrollo en software libre y para la web.

Idea a Defender: Si se seleccionan adecuadamente los elementos arquitectónicos, se establecen las relaciones entre estos y el ambiente, y se identifican los principios que orienten el diseño y evolución posterior del sistema se obtendrá un diseño de arquitectura multiplataforma, robusta, flexible y extensible para el Sistema Integrado de Gestión Estadísticas 2.0 Nuragas en su desarrollo en software libre y sobre la web.

Para su consecución se han planteado las siguientes **Tareas de la Investigación:**

1. Estudio del estado del arte de los sistemas estadísticos, de la arquitectura de sistemas web, las tecnologías, herramientas y los diferentes recursos reutilizables.

2. Definición de las herramientas a utilizar en el entorno de desarrollo.

3. Diseño de la propuesta arquitectónica multiplataforma, robusta, flexible, y extensible que cumpla con los requerimientos de los clientes y las necesidades de los usuarios.

4. Validación del diseño de arquitectura propuesto.

En la cual se han de utilizar los siguientes **Métodos:**

Teóricos:

- Análisis y Síntesis: para el procesamiento de la información y arribar a las conclusiones de la investigación, así como para precisar las características del modelo arquitectónico propuesto.
- Histórico – Lógico: Para determinar las tendencias actuales de desarrollo de los modelos y enfoques arquitectónicos.
- Inducción y deducción: A partir del estudio de distintos estilos y modelos de arquitecturas así como del estudio de los diferentes frameworks para el desarrollo de arquitecturas para aplicaciones web arribar a proposiciones específicas.
- Método Sistémico: Para determinar los componentes y definir las relaciones entre estos.

Empíricos:

- Observación: Para la percepción selectiva y sistemática de las restricciones y propiedades del sistema, y para el control de la evolución de la arquitectura inicial.

Resultados Esperados

Se espera obtener un diseño de Arquitectura para el Sistema Integrado de Gestión Estadística Nuragas 2.0 que sea multiplataforma, robusto, escalable, flexible y extensible que cumpla con los requerimientos de los clientes y usuarios para su desarrollo en software libre y para la web. Dicha arquitectura a diferencia de la anterior permitirá hacer un uso más racional del hardware disponible en las oficinas de estadística a la vez que permitirá la integración al PATDSI una suite de Inteligencia de Negocios

haciendo de este un producto puntero que tendría una fuerte demanda nacional y probablemente internacional.

Este trabajo está estructurado en 3 capítulos:

Capítulo 1: Fundamentación Teórica

En este capítulo se presenta la definición del marco teórico y del modelo teórico de la investigación, y el estudio del estado del arte de los sistemas estadísticos, de la arquitectura de sistemas web, las tecnologías, herramientas y los diferentes recursos reutilizables.

Capítulo 2: Diseño Arquitectónico

Se hace una propuesta de solución al problema planteado en el diseño teórico de la investigación, a través de la presentación y discusión de las principales secciones incluidas en el artefacto de ingeniería "Documento Descripción de la Arquitectura".

Capítulo 3: Validación de la Propuesta de Arquitectura

Se realiza la evaluación de la propuesta de diseño de la arquitectura utilizando el método ATAM basado en escenarios.

CAPÍTULO 1: Fundamentación Teórica

En este capítulo se presenta brevemente los elementos del trabajo estadístico en Cuba para luego de forma sintética analizar el estado del arte de los sistemas de estadística existentes en el mundo y en nuestro país. Se analiza la evolución de la disciplina arquitectónica, su desenvolvimiento dentro del proceso de desarrollo de software, el rol del arquitecto y los elementos de calidad relevantes en la arquitectura de software. Se presentan las tecnologías con un importante impacto arquitectónico sobre las cuales se asentará el desarrollo posterior.

Estado del Arte de los Sistemas de Gestión Estadística

La Estadística en Cuba, la Oficina Nacional de Estadísticas

El control de los datos estadísticos dentro de la infraestructura de un país constituye el eslabón principal para la toma de decisiones de los órganos de dirección en los diferentes sectores socioeconómicos. Cuba tiene una larga historia en materia de estadística. La ONE es una institución gubernamental adscripta al Ministerio de Economía y Planificación rectora de la estadística oficial en el país y, mediante su Sistema Estadístico Nacional (SEN), organiza, dirige, controla y regula la actividad de la estadística en Cuba.

Fig. 1 Estructura Institucional de la ONE

La ONE tiene una estructura institucional distribuida territorialmente en las provincias y municipios del país, (Fig. 1). Existen 14 oficinas provinciales (OTE) y 169 oficinas municipales de estadísticas (OME)[1]. Esas oficinas tienen atención administrativa y metodológica por la oficina nacional.

La información estadística en Cuba está agrupada en el Sistema Estadístico Nacional (SEN), que se

[1] La clasificación presentada como ONE, OTE, y OME empezó a quedar en desuso a partir del 2007 cuando se acuerda en un Taller Nacional con los delegados de las provincias utilizar los términos ONE nacional, ONE provincial y ONE municipal respectivamente.

divide en los subsistemas siguientes:

- Sistema de Información Estadística Nacional (SIEN): que incluye los formularios estadísticos recopilados por la Oficina Nacional de Estadísticas a través de sus dependencias y los ministerios con fines nacionales.
- Sistema de Información Estadística Territorial (SIET): que incluye los formularios estadísticos recopilados por la oficina de estadística territorial y entidades territoriales con fines territoriales aprobados por la ONE.
- Sistema de Información Estadística Complementaria (SIEC): que incluye los formularios estadísticos recopilados por todos los ministerios y entidades para sus propios fines aprobados por la ONE.

La información estadística la brindan las empresas, instituciones y organizaciones según lo previsto en el programa de captación de datos convenido con todas las entidades.

En (Robert Lobo, y otros, 2008) se identifican los principales requisitos funcionales que debe cumplir un sistema para que se adecue a las necesidades de la ONE, en síntesis se requiere que la aplicación garantice:

1. La confección de los formularios del SEN
2. Gestionar los centros informantes y los atributos de interés relativos a estos
3. Captar y validar la información a través del llenado del formulario y su posterior almacenamiento en una base de datos.
4. Transferir la información a lo largo de la jerarquía según corresponda.
5. Procesar y recuperar la información

Se trata fundamentalmente más que del análisis matemático relativo a la estadística, de un sistema de gestión masiva de datos estadísticos, dado el carácter central de la administración del estado, característica específica de Cuba que obliga a realizar un censo económico todos los meses con todas las implicaciones que esto lleva. Actualmente los paquetes estadísticos existentes no contemplan alguno que otro de los requisitos planteados, otros se acercan pero son privativos y caros al punto de

hacerse prohibitivos, por otro lado se centran en el trabajo de análisis fundamentalmente, no en el proceso de gestión de la información. De los productos nacionales el primero, MicroSet NT, ya no contempla las necesidades actuales, ni resulta lo suficiente amistoso al usuario que se esperaría y se hace difícil mantener funcionando sobre los modernos equipos de cómputo ya que data de los años 70, el segundo (SIGE v. 1) si bien cumple con los requisitos está desarrollado para tecnología propietaria, es una aplicación de escritorio que requiere ser instalada y configurada en cada PC individualmente además del despliegue del servidor. En el análisis se han tenido en cuenta los exponentes más representativos y conocidos nacional e internacionalmente.

Biomedical Computers Programs (BMDP)

Según (Statistical Solutions Ltd., 2005) BMDP es uno de los paquetes de software estadísticos más antiguos. El primer manual para BMDP se publicó en 1961. Los resultados de cada programa se pueden guardar en un archivo de BMDP y utilizarse como entrada en otros programas. No contempla la gestión de datos de acuerdo requisitos que la ONE demanda a pesar de que posee un excelente potencial para el procesamiento estadístico como disciplina matemática propiamente. BMDP es comercializado como una suite con alrededor de 42 programas individuales para SO Windows por Statistical Solutions Ltd.

EViews

EViews es un paquete estadístico para Windows, usado principalmente para análisis econométrico desarrollado por Quantitative Micro Software (QMS). El EViews combina la tecnología de hoja de cálculo con tareas tradicionales encontradas en software estadístico tradicional, empleando una interfaz de usuario gráfica. Estas características se combinan con un lenguaje de programación. El EViews puede ser empleado para análisis estadístico general, pero es especialmente útil para realizar análisis econométrico (Brockwell, y otros, 2002). La versión más actualizada del EViews es la 6.0, no contempla la elaboración de formularios para la captura masiva de información, ni las características de gestión de información requeridas por la ONE.

Minitab

Minitab fue diseñado para ejecutar funciones estadísticas básicas y avanzadas. Combina el uso de Microsoft Excel con la capacidad de ejecución de análisis estadísticos (Minitab Inc., 2009). En 1972, instructores del programa de análisis estadísticos de la Universidad Estatal de Pennsylvania (Pennsylvania State University) desarrollaron MINITAB como una versión ligera de OMNITAB, un programa de análisis estadístico del Instituto Nacional de Estándares y Tecnología (NIST) de los Estados Unidos. Como versión completa en el 2006 cuesta $1195 USD, pero una versión para estudiantes y académicos se ofrece como complemento de algunos libros de texto. Es fundamentalmente utilizado en una metodología para la optimización de procesos empresariales. A similitud de los anteriores posee un fuerte trabajo componente matemático-estadístico pero no está concebido para gestión de información estadística propiamente.

R-Project

R es un lenguaje y entorno de programación para análisis estadístico y gráfico. Se trata de un proyecto de software libre, resultado de la implementación GNU del premiado lenguaje S. R y S-Plus -versión comercial de S- son, probablemente, los dos lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy populares en el campo de la investigación biomédica, la bioinformática y las matemáticas financieras (R-project, 2009). A esto contribuye la posibilidad de cargar diferentes librerías o paquetes con finalidades específicas de cálculo o gráfico. Pero esencialmente se trata de un lenguaje y de las herramientas de desarrollo de este por lo que no contempla las necesidades que en principio requiere la ONE.

Statistical Package for the Social Sciences (SPSS)

Statistical Package for the Social Sciences (SPSS) es un programa estadístico informático muy usado en las ciencias sociales y las empresas de investigación de mercado. Fue creado en 1968 por Norman H. Nie, C. Hadlai (Tex) Hull y Dale H. Bent. Tiene capacidad para trabajar con bases de datos de gran tamaño, y posiblemente las mejores prestaciones para el cálculo estadístico propiamente. Resulta un fuerte competidor para R (software libre) y para el resto de los programas estadísticos comerciales (SPSS Inc., 2009). Tiene versiones para Windows y para Linux, pero se trata de un software privativo.

Su equivalente libre PSPP está en desarrollo y aspira a alcanzar todas las funcionalidades prestadas por SPSS. Es un sistema modular fuertemente extensible. No cumple con los requisitos requeridos por la ONE para el procesamiento estadístico en lo que a gestión respecta.

SAS

Statistical Analysis System (SAS) es quizás la solución de inteligencia de negocios más completa que hay en el mercado. Existe como lenguaje de procesamiento estadístico desde los años 70 cuando sus primeras versiones estaban orientadas a mainframes de IBM. Es comercializado por la compañía del mismo nombre quien tiene una importante cuota de mercado en el mundo de este tipo de sistemas. Cumple con los requisitos de la ONE pero no se trata de una herramienta libre, sino comercial y por demás costosa.

Microset NT

Microset NT fue desarrollado en MS-DOS en los años 70 por informáticos de la ONE, aún se sigue explotando. Permite parte de las funcionalidades demandadas, pero no el total y dinámico trabajo con la información suministrada, condiciona la salida a un formato de tablas previamente parametrizados en el sistema. Dado que se desarrollo como aplicación para consola (con elementos visuales para el modo texto) no es todo lo amigable que se puede esperar de una aplicación actual. MicroSet NT no evolucionó como aplicación para Windows y actualmente presenta problemas para ejecutarse sobre cualquier Windows que no pertenezca a la familia 9x. Independiente de ello MicroSet capta los requisitos esenciales de un sistema estadístico para la ONE, hace un excelente trabajo en los que a usabilidad refiere ya que un usuario con experiencia llega a ser muy eficiente en su utilización y aún sin utilizar bases de datos ni un modelo cliente servidor permite trabajar en red de manera colaborativa. Las principales ideas de este son la base del SIGE.

Sistema Integrado para la Gestión Estadística (SIGE)

El Sistema Integrado para la Gestión Estadística (SIGE) 1.0 fue desarrollado de conjunto con la ONE de forma tal que contempla el marco metodológico y operativo de esta. Como sistema está integrado por

módulos altamente especializados y cohesivos que permiten la gestión de los centros informantes, la elaboración del SEN para realizar la captación de los datos estadísticos, la digitalización y validación de la información recolectada, y su posterior consulta y análisis, a tales efectos está integrado por los módulos de Registros y Clasificadores, Generador de Modelos y Encuestas, Entrada de Datos, y Reportes respectivamente (Robert Lobo, y otros, 2008).

El SIGE responde perfectamente a las necesidades de la ONE sin embargo fue realizado sobre .NET utilizando como gestor de bases de datos SQL Server 2000 y el Análisis Manager, también de Microsoft, para el análisis avanzado de datos, SIGE es una aplicación de escritorio que entre los requisitos de instalación demanda Windows 98 o superior como sistema operativo para el cliente y Windows Server 2003 o superior como sistema operativo para el servidor, algo que contradice la política de migración a Software Libre que sigue el país. Respecto al uso de la tecnología propietaria en este sistema como decisión inicial facilitó la productividad del equipo quien además ya estaba capacitado en el uso de las herramientas para el desarrollo pero en la actualidad no se puede esperar que un sistema de apoyo a la gestión del gobierno desarrollado nacionalmente se pliegue al uso software propietario cuya licencia además excluye a nuestro país (y otros) en una clausula especial agregada por el congreso de los Estados Unidos de América. SIGE no contempla tampoco la integración al ERP (proyecto que comenzó después) algo que ahora se hace necesario a fin de captar automáticamente de los registros primarios de las empresas (almacenados en la base de datos del ERP) la información estadística que de otra forma tendría que introducirse a mano. Además incorporado al ERP no solo sería una herramienta de la ONE sino que llegaría a cada una de las empresas que se favorecerían enormemente de este y simplificarían la cantidad de trabajo que hoy tiene la ONE. No posee una versión para la web y puesto que se trata de una aplicación desktop con arquitectura cliente servidor debe ser instalado y configurado en cada cliente individualmente, además del despliegue de los componentes del lado del servidor.

A continuación se ofrece una tabla comparativa de las diferentes soluciones antes presentadas:

Tabla 1 Comparación de los diversos paquetes estadísticos existentes

Solución	Cumple los requisitos	Sistema Operativo	Libre
BMDP	*Insuficiente* en lo que respecta a la entrada de datos y a la gestión de cantidades masivas de información.	Windows	No
EViews	*Insuficiente* en lo que respecta a la entrada de datos y a la gestión de cantidades masivas de información.	Windows	No

Minitab	*Insuficiente* en lo que respecta a la entrada de datos y a la gestión de cantidades masivas de información.	Windows	No
R-Project	*No,* se trata de un lenguaje de procesamiento estadístico y de las herramientas de soporte al desarrollo con el mismo.	Multiplataforma	Si
SPSS	*Insuficiente* en lo que respecta a la entrada de datos y a la gestión de cantidades masivas de información.	Windows / Linux	No
SAS	Sí, pero no es libre y resulta prohibitivo económicamente.	Windows	No
MicroSet NT	Si	MS DOS	
SIGE 1.0	Si	Windows	

De las posibilidades analizadas la mayoría no se ajusta a las necesidades de la ONE en cuanto a gestión de datos, SAS que es quien más se adecua es un software privativo desarrollado para Windows y además costoso dado que se trata de una solución integral de Business Intelligence. Las soluciones más adecuadas son por supuesto las desarrolladas nacionalmente que responden de manera particular a la ONE sin embargo MicroSet NT se comporta inestable fuera de MS DOS y por su antigüedad no puede responder a las más revolucionarias tendencias del trabajo estadístico, por su parte el SIGE 1.0 se asienta sobre una Arquitectura de tecnologías privativas empezando por el sistema operativo, no es una aplicación web y en principio no se pensó para la integración al ERP cubano.

Por ello se hace necesario el desarrollo de un nuevo sistema que tome lo mejor de las soluciones existentes (particularmente del SIGE 1.0) se asiente sobre desarrollos libres que potencien la soberanía tecnológica y que sea una aplicación web. Es por tanto imprescindible definir para este desarrollo la Arquitectura del Software que contemple las nuevas necesidades planteadas.

La Arquitectura de Software, su evolución

La mayor parte de la bibliografía consultada coincide en afirmar que es Edsger Dijkstras (Introducción a la Arquitectura de Software, 2004) quien propuso que se estableciera una estructuración correcta de los sistemas antes de lanzarse a programar, en momentos tan tempranos como 1968, reconociéndose esta afirmación como primer antecedente de una conciencia hacia la disciplina de centrar el desarrollo en función de una arquitectura. Continuado por ideas como la del refinamiento gradual (stepwise refinement) y programación a lo grande (programming in the large) de Niklaus Wirth primero y de Kron y

DeRemer después, se fueron decantando las perspectivas entre los ingenieros y los arquitectos (Camacho, y otros, 2004) (Arquitectura de software. Arquitectura orientada a servicios, 2008) (Introducción a la Arquitectura de Software, 2007).

En 1969 un año después de la concepción de la Ingeniería de Software como disciplina en otra sección de conferencias de la OTAN P. I Sharp reflexiona sobre las ideas de Dijkstras intuyendo la existencia de algo más aparte de la ingeniería, haciendo alusión al desarrollo del sistema operativo OS/360 hace especial énfasis en el diseño y la forma de los sistemas.

David Parnas sería quien más aportes haría al desarrollo del concepto en la década del 70. En diferentes trabajos desarrolla los conceptos de módulos, ocultamiento de información, estructuras de software y familia de sistemas. Demuestra que los criterios de descomposición de un sistema afecta la estructura de los programas y a su vez plantea que estas primeras decisiones tendrá un impacto significativo en el desarrollo ulterior, se convierte de esta forma en el introductor de ideas esenciales que permanecen hasta nuestros días.

La Arquitectura de Software nace como disciplina del desarrollo de software específicamente en la década del 90. En 1992 en el trabajo de Dewayne Perry y Alexander Wolf “Foundations for the study of software architecture” aunque dicho trabajo comenzó un poco antes en 1989, es donde por primera vez se habla del término “arquitectura” como analogía a la arquitectura de edificios. En este trabajo ellos definen un modelo basado en la intuición que consta de 3 componentes: elementos, forma y razón. Los elementos pueden ser de procesamiento, de datos o de conexión. La forma define las propiedades de los elementos, sus relaciones y las restricciones que operan sobre estos. La razón proporciona los motivos subyacentes para una arquitectura determinada en términos de restricciones que derivan de los requisitos del sistema. Además los autores hacen una afirmación que la historia posterior confirmaría: *“la década del 90, creemos, será la década de la Arquitectura de Software”.*

Es en la prodiga década del 90 que se empiezan a desarrollar los diferentes conceptos relacionados a la arquitectura, fundamentalmente proveniente de los trabajos del Software Engineering Institute de la Universidad de Carnegie y Mellon (CMU SEI). Es en este momento que se realiza un trabajo de unificación de la terminología, se desarrolla la tipificación de los estilos arquitectónicos, aparecen los

Lenguajes de Descripción Arquitectónica (ADL por sus siglas en inglés), también se consolida la concepción de las vistas arquitectónicas reconocidas en todos los marcos generalizadores de la arquitectura. En el 2000 Roy Fielding realiza una célebre disertación de maestría en la que presenta el modelo REST relacionado a las tecnologías de Internet y los modelos orientados a servicios y recursos, también se publica la versión definitiva de la recomendación IEEE 1471-2000 que busca homogenizar y ordenar la nomenclatura de descripción arquitectónica reconociendo a los estilos como el modelo fundamental de representación conceptual (Introducción a la Arquitectura de Software, 2004), cumpliéndose la casi profética afirmación de Perry y Wolf.

Es oportuno citar la definición de Arquitectura de Software versada en la recomendación IEEE 1471-2000 a la cual el autor del presente trabajo se acoge: "… es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución".

La Arquitectura se ha visto fuertemente influenciada por las tendencias de la industria del desarrollo de software existiendo si no un divorcio un desfasaje entre los avances de la academia y los conceptos que maneja la industria, en otras palabras como lo plantea (Introducción a la Arquitectura de Software, 2004) citando a Clements y a Northrop: "la arquitectura está siguiendo a la práctica no liderándola". Es necesario entonces mencionar importantes acontecimientos como lo son la introducción de los principios de factoría y líneas de productos, la aparición de las metodologías ágiles, la ingeniería de software basada en componentes y la aparición de los patrones. En este marco se aprecian tendencias en los estudios arquitectónicos que se van a concretar con posturas que según (Introducción a la Arquitectura de Software, 2004) llegan a tomar el carácter de escuelas, las más significativas son:

1. Arquitectura como etapa de ingeniería y diseño orientada a objetos: cuyos principales exponentes son James Rumbaugh, Ivar Jacobson, Grady Booch, Craig Larman y otros, ligado estrechamente al mundo de UML y Rational. En esta escuela, si bien se reconoce el valor primordial de la abstracción y del ocultamiento de información promovido por Parnas, estos conceptos tienen que ver más con el encapsulamiento en clases y objetos que con la visión de conjunto arquitectónica. Jamás se hace referencia a los lenguajes de descripción arquitectónica, que representan uno de los puntos fuertes reconocidos de la arquitectura de software; sucede

como si la disponibilidad de un lenguaje unificado de modelado los tornara superfluos. Para esta tendencia la arquitectura de software, concierne a decisiones sobre organización, selección de elementos estructurales, comportamiento, composición y estilo arquitectónico susceptibles de ser descriptas a través de las cinco vistas clásicas del modelo 4+1. El presente trabajo se ve fuertemente influenciado por esta corriente no solo por ser la que docentemente se promueve en la Universidad sino además por su marcada aplicación industrial, a pesar de ello existen características de las otras escuelas que también se han considerado relevantes.

2. Arquitectura estructural, basada en un modelo estático de estilos, ADL y vistas: reconocida como la corriente fundacional y clásica de la disciplina. Los representantes de esta corriente son todos académicos, mayormente de la Universidad Carnegie Mellon en Pittsburgh: Mary Shaw, Paul Clements, David Garlan, Robert Allen, Gregory Abowd, John Ockerbloom, aunque es la que ha hecho el esfuerzo más importante por el reconocimiento de la arquitectura como disciplina, sus categorías y herramientas son todavía mal conocidas en la práctica industrial. En principio se pueden reconocer tres modalidades en cuanto a la formalización; los más informales utilizan descripciones verbales o diagramas de cajas, los de talante intermedio se sirven de ADL y los más exigentes usan lenguajes formales de especificación como CHAM y Z en la vertiente más formalista, representada por el grupo de Mark Moriconi en el SRI de Menlo Park. Mientras algunos participantes excluyen el modelo de datos de las incumbencias de la arquitectura (Shaw, Garlan), otros insisten en su relevancia (Medvidovic, Taylor). Todo estructuralismo es estático; hasta fines del siglo XX, no está muy claro el tema de la posición del modelado arquitectónico en el ciclo de vida, por tanto la escuela estructuralista no nos da una visión de las actividades y su lugar en el proceso de desarrollo, a pesar de que no es la tendencia que se sigue en este trabajo si se tienen en cuenta las principales ideas asociadas a esta corriente.

3. Estructuralismo arquitectónico radical: se trata de un desprendimiento de la corriente estructuralista, mayoritariamente europea, que asume una actitud más enfrentada con el UML. Se visualizan dos tendencias, una que excluye la relevancia del modelado orientado a objetos y otra que procura definir nuevos meta modelos y estereotipos de UML como correctivos de la situación.

4. Arquitectura basada en patrones: reconoce la importancia de un modelo devenido del diseño orientado a objetos. Encuentra su texto referencia es la serie POSA de Buschman y otros, secundariamente el texto de la Banda de los Cuatro En esta manifestación prevalece cierta tolerancia hacia modelos de proceso tácticos, no tan macroscópicos, y eventualmente se expresa cierta simpatía por las ideas de Martin Fowler y las premisas de la programación extrema. El diseño consiste en identificar y articular patrones preexistentes, que se definen en forma parecida a los estilos de arquitectura (Shaw, 1996).

5. Arquitectura procesual: es una corriente reciente y trata de resolver la dificultad de la arquitectura estructuralista en lo que en relación al ciclo de vida y al proceso de desarrollo del software refiere. Intenta establecer modelos de ciclo de vida y técnicas de elicitación de requerimientos, brainstorming, diseño, análisis, selección de alternativas, validación, comparación, estimación de calidad y justificación económica específicas para la arquitectura de software.

6. Arquitectura basada en escenarios: es la corriente más nueva y es un movimiento predominantemente europeo, con centro en Holanda. Recupera el nexo de la AS con los requerimientos y la funcionalidad del sistema, ocasionalmente borroso en la arquitectura estructural clásica. Los teóricos y practicantes de esta modalidad de arquitectura se inscriben dentro del canon delineado por la arquitectura procesual, respecto de la cual el movimiento constituye una especialización. En esta corriente suele utilizarse diagramas de casos de uso UML como herramienta informal u ocasional, dado que los casos de uso son uno de los escenarios posibles. Esta modalidad ha resultado clave a la hora de definir los métodos de evaluación de arquitecturas entre ellos ATAM, CBAM, QASAR y otros. Se ha tenido en cuenta precisamente en el capítulo 3 de este trabajo para a través de un método validar la propuesta de arquitectura.

En resumen teniendo en cuenta la evolución histórica de la disciplina y las diferentes corrientes que han devenido escuelas de la arquitectura en el presente trabajo se adoptará una alineación con la primera de las corrientes con consideraciones relativas en arquitectura basada en patrones y arquitectura basada en escenarios puesto que estas tres corrientes son las que más cerca se encuentran del lado de la industria y que en síntesis nos aportan una mayor riqueza conceptual en función de la praxis objetiva

a la disciplina al desarrollo de software. Ahora si bien se ha analizado los elementos que dan lugar a la disciplina es necesario también tener en cuenta el factor humano interpretado en el rol del arquitecto, el cual se trata a continuación.

El perfil del Arquitecto de Software

El rendimiento del rol de arquitecto de sistema requiere conocimiento de las diversas disciplinas que contribuyen a la ingeniería de software y que poseer potentes habilidades de análisis, razonamiento e inferencia, disciplina de autoestudio y superación, así como la habilidad de síntesis (Rational Unified Process, 2003). Se ajusta a esta descripción el pensamiento que preside la definición del rol en la ayuda de RUP:

"El arquitecto ideal debe ser una persona de letras, matemático, familiarizado con la historia, diligente estudioso de la filosofía, conocedor de la música, no ignorante de la medicina, entendido en las respuestas de las leyes, versado en la astronomía y en los cálculos astronómicos."

—Vitruvius, alrededor del 25 a.C.

Se hace referencia aquí a un arquitecto de edificios pero sintetiza el hecho de que un arquitecto es en principio es alguien con elevados niveles de preparación. El arquitecto de sistema no se ocupa simplemente de la tecnología de la solución, sino de muchas otras cuestiones como, por ejemplo, operación del sistema, rendimiento, viabilidad económica, capacidad de fabricación y soporte logístico, así como los factores políticos, técnicos, sociales, financieros y medioambientales (entre otros) que son clave para estas cuestiones. Se debe poseer la experiencia y madurez necesarios para permitir un análisis objetivo e intercambiar estudios que se van a realizar a fin de seleccionar la mejor solución entre muchas, con la capacidad de discernir correctamente cuándo la información es incompleta o ambigua, y la capacidad de reconocer que lo "mejor" suele estar dictado por consideraciones políticas y económicas además de consideraciones de ingeniería.

Al trabajar con sistemas que están compuestos de personas, hardware y software, el arquitecto de sistema tiene que ser muy consciente de las limitaciones y restricciones físicas existentes en cualquier solución por los componentes humanos y de hardware. Además de la experiencia en la ingeniería de

sistemas (y el conocimiento de disciplinas aliadas como, por ejemplo, la investigación de operaciones y la economía de ingeniería), el arquitecto de sistema debe tener unos sólidos conocimientos de ingeniería de software (a causa de su ubicuidad en los sistemas modernos), así como:

- Experiencia en el dominio de problemas y un profundo conocimiento de los requisitos. Esta experiencia puede ampliarse mediante un equipo de arquitectura de sistemas.

- Cualidades de liderazgo, para dirigir el esfuerzo técnico entre los diferentes equipos, tomar decisiones críticas bajo presión y adherirse a estas decisiones. Para ser efectivos, el arquitecto de sistema y el gestor de proyectos deben trabajar conjuntamente, con el arquitecto de sistema dirigiendo las cuestiones técnicas y el gestor de proyectos dirigiendo las cuestiones administrativas. El arquitecto de sistema debe tener autoridad para tomar decisiones técnicas.

- Habilidades de comunicación para ganar confianza, persuadir, motivar y orientar. El arquitecto de sistema no tiene autoridad en virtud de su cargo, sino a causa de su habilidad demostrada y sus resultados. Para ser efectivo, el arquitecto de sistema debe ganarse el respeto del equipo de proyecto, el gestor de proyectos, el cliente y la comunidad de usuarios, así como del equipo de gestión.

El arquitecto de sistema es la fuerza técnica dirigente detrás del proyecto, no un consultor o un soñador. Orientación a objetivos y pro actividad centrándose exclusivamente en los resultados. La carrera de un arquitecto de sistema que alcanza el éxito es una larga serie de decisiones sub-óptimas (pero no que habitualmente sean *gravemente* sub-óptimas) efectuadas en condiciones de incertidumbre y bajo presión.

En la función de Arquitecto Principal debe tener conocimiento de aéreas de la arquitectura como Seguridad, Acceso a Datos, Presentación, Negocio, Interacción con Periféricos entre otras. El rol de arquitecto de sistema abarca las habilidades del diseñador de sistemas, aunque con un enfoque estratégico, no detallado.

Estilos Arquitectónicos

Los estilos arquitectónicos surgen en analogía al término utilizado en la arquitectura de edificios para tipificar las características distintivas de las formas básicas posibles de estructurar un software. Las primeras definiciones explícitas de estilo fueron propuestas por Dewayne Perry de AT&T Bell Laboratories de New Jersey y Alexander Wolf de la Universidad de Colorado, desde el mismo momento en que se funda la arquitectura como disciplina. Ellos definieron un estilo arquitectónico como "una abstracción de tipos de elementos y aspectos formales a partir de diversas arquitecturas específicas. Un estilo arquitectónico encapsula decisiones esenciales sobre los elementos arquitectónicos y enfatiza restricciones importantes de los elementos y sus relaciones posibles" (Wolf, 2002). Posteriormente Mary Shaw y Paul Clements conceptualizan los estilos arquitectónicos como un "conjunto de reglas de diseño que identifica clases de componentes y conectores que se pueden manejar para componer el sistema, junto con las restricciones locales o globales que determinan como se lleva a cabo la composición"

Los estilos se ordenan en seis o siete clases fundamentales y unos veinte ejemplares, como máximo. A continuación la clasificación propuesta por Shaw y Garlan (Shaw, y otros, 1996):

Estilos de Flujo de datos

- Tuberías y Filtros

Estilos centrados en datos

- Arquitecturas de Pizarra o repositorio

Estilos de Llamada y Retorno

- Modelo – Vista – Controlador (MVC)
- Arquitectura en Capas
- Arquitectura Orientada a Objetos
- Arquitectura basada en Componentes

Estilo de Código Móvil

- Arquitectura de Máquinas Virtuales

Estilos Peer–To–Peer

- Arquitectura basada en Eventos
- Arquitecturas Orientas a Servicios (SOA)

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

A continuación se presenta una descripción un poco más detallada de algunos de ellos por resultar de sumo interés para la concepción de la arquitectura del sistema en desarrollo.

Arquitectura en Capas

La arquitectura en capas constituye uno de los estilos que aparecen con mayor frecuencia. Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Este estilo instrumenta una vieja idea de organización estratigráfica que se remonta a las concepciones formuladas por Edsger Dijkstra en la década de 1960, de que los grandes sistemas podrían organizarse por niveles de complejidad asemejándose a las capas de una cebolla. Para algunos autores más estrictos para con el estilo, las capas internas están ocultas a todas las demás. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas que encapsulan tanto recursos como servicios facilitando aumentando los niveles de abstracción y facilitando la comprensión global del sistema. Entre las principales ventajas que se señalan del estilo (Introducción a la Arquitectura de Software, 2004) destacan:

- Alto nivel de abstracción: estilo soporta un diseño basado en niveles de complejidad, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Optimización y refinamiento: en principio el estilo admite la sustitución directa de una capa intermedia sin afectar el funcionamiento del sistema siempre y cuando se respeten las dependencias entre niveles. Los cambios en una capa dan el menor impacto global al sistema.

Entre las desventajas sin embargo se manifiesta que algunos problemas no admiten un mapeo a una estructura jerárquica, incluso aún cuando resulte lógico consideraciones asociadas al rendimiento pueden requerir las capas superiores requieran acceder directamente a servicios de las inferiores.

También se argumenta que si bien facilita el control y encapsulamiento de aplicaciones complejas resulta excesivamente pesado para aplicaciones simples.

Al respecto de las desventajas de dicho estilo el autor considera que no resulta relevante puesto que el sistema que se prevé es una clásica aplicación empresarial en la cual la arquitectura multicapa ha resultado más que probada con excelentes resultados en tanto las concesiones de rendimiento son superadas por lo ganado en organización y abstracción.

Modelo – Vista – Controlador

El patrón Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

- Modelo: responsable de administrar el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- Vista: responsable de la visualización de la información.
- Controlador: responsable de interpretar los eventos producidos por el usuario, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Esta separación permite construir y probar el modelo independientemente de la representación visual. Particularmente en aplicaciones web la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está claramente definida. Entre las ventajas del estilo señaladas están las siguientes:

- Soporte de vistas múltiples: dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente.

- Adaptación al cambio: los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Entre las desventajas, se han señalado:

- Complejidad: dado que introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. Profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar.

- Costo de actualizaciones frecuentes: desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas. Si el modelo experimenta cambios frecuentes podría desbordar las vistas con una lluvia de requerimientos de actualización.

Muchos de los marcos para el desarrollo de aplicaciones web incorporan en su diseño los elementos antes mencionados que caracterizan a este estilo, favorecido como se había mencionado por la topología misma de las aplicaciones web. Symfony y Zend framework son dos ejemplos concretos, en particular se opta por Symfony entre otras razones que se explican más adelante por proponer un mayor grado de organización, estructuración y por la existencia de un “know-how” previo en este que no puede ser desatendido.

Arquitectura de Máquinas Virtuales

La arquitectura de máquina virtual asume la existencia de un intérprete que media entre la aplicación y el hardware original sobre el cual se ejecuta la aplicación codificada en un determinado lenguaje, ello potencia la portabilidad del código en lo que respecta a la arquitectura de máquina real sobre el que este se ejecuta, resulta por tanto suficiente disponer del intérprete para la plataforma y queda garantizada la portabilidad de la aplicación. Este estilo surge en la década del 50, cuando apenas nacía la ingeniería de software, a razón de una propuesta de una máquina virtual que ejecutara un lenguaje universal en byte-codes, de manera que las aplicaciones podían escribirse en las capas más altas y

ejecutarse donde fuere sin tener que recompilarse, siempre que hubiera una máquina virtual entre el programa por un lado y el sistema operativo y la máquina real por el otro (Proposal for a universal computer-oriented language, 1958). Numerosos lenguajes y ambientes de scripting utilizan máquinas virtuales: Perl, Java Script, Python, PHP, Pascal y otros.

Arquitectura basada en Componentes

La Arquitectura basada en componentes identifica como elemento fundamental a los componentes de software y se centra en la integración de varios de ellos para conformar un sistema, se apoya fundamentalmente de la ingeniería de software basada en componentes. Los componentes que constituyen una arquitectura pueden ejercer influencia sobre la calidad del sistema final. .

Ingeniería de Software basada en Componentes

La Ingeniería de Software basada en Componentes (Component Based Software Engineering, CBSE) está enmarcada en la Ingeniería de Software y existe un interés cada vez mayor en ella, inclusive una identificación con la correspondiente escuela de arquitectura. El interés está dado primeramente por la concepción de las factorías de software (Trujillo Casañola, 2006) que plantean la posibilidad de implantar patrones de producción industrial al desarrollo de software y los preceptos de que es mucho más fácil y rápido reutilizar componentes ya desarrollados y probados, a tener que implementarlos desde cero.

Los antecedentes más significativos de las plataformas de componentes se pueden establecer en DCE y CORBA, desarrolladas por iniciativa de los consorcios OSF (Open Software Foundation) y OMG (Object Management Group), respectivamente. Aunque DCE (Distributed Computing Environment) tuvo en un principio una buena acogida por parte de la industria, algunas de las limitaciones que presenta (no es orientada a objetos, no define servicios de transacciones y solo permite invocaciones estáticas, no dinámicas) le han hecho perder gran parte de su cuota de mercado en el desarrollo de aplicaciones distribuidas. Sin embargo, la especificación de la arquitectura CORBA (Common Object Request Broker Architecture) (OMG, 1999), a pesar de definirse como una plataforma de objetos distribuidos, asienta los

principios básicos de la tecnología de componentes, habiendo consolidado en gran medida su posición. Posteriormente han aparecido diversas plataformas, entre las que debemos citar, por su importancia, COM (Component Object Model), desarrollada por Microsoft (Rogerson, 1997) y JavaBeans, desarrollada por Sun.

Uno de los campos en los que la tecnología de componentes se ha mostrado más activa es en el desarrollo de marcos de trabajo (frameworks) (Fayad, y otros, 1997). Estos se definen como un diseño reutilizable de todo o parte de un sistema, representado por un conjunto de componentes abstractos, y la forma en la que dichos componentes interactúan. Otra forma de expresar este concepto es que un marco de trabajo es el esqueleto de una aplicación que debe ser adaptado por el programador según sus necesidades concretas (Johnson, 1997). El desarrollo de aplicaciones a partir de un framework se lleva a cabo mediante la extensión del mismo, para lo cual el usuario debe tener información acerca de cuáles son sus puntos de entrada y cómo adaptarlos. Un framework se considera como la plantilla (template) de la aplicación o de la parte de la aplicación que se quiera desarrollar, debe ser bien seleccionado en dependencia de las restricciones del sistema y de las posibilidades que nos brinde el framework, estos a la vez no son excluibles sino que se pueden extender y combinarse para lograr una interacción entre distintas partes de un sistema. Se destacan en este contexto igualmente los Applications Block de Microsoft que llevan el desarrollo a un nivel de conceptualización superior, tratando a través de patrones implantados a nivel de componentes problemas reiterativos en el área del desarrollo de sistemas empresariales. Los componentes no son separables de la arquitectura. Así mismo, todos los modelos y plataformas de componentes imponen una serie de compromisos sobre dichos componentes, indicando la forma en que se describe su interfaz o que mecanismos básicos tienen que proporcionan. La interfaz de un componente no solo muestra su funcionalidad, de forma abstracta, sino que implica además toda una serie de restricciones arquitectónicas ligadas a la plataforma o modelo del que forma parte (Szyperski, 2000).

¿Cómo influirá una arquitectura de software basada en componentes en el desarrollo del Sistema Integrado para la Gestión Estadística 2.0 Nuragas?

La arquitectura de software basada en componentes permitiría utilizar componentes ya probados que no tuvieran errores y acortaría el tiempo de desarrollo, permitiría además la creación de nuestros propios

componentes que deberán ser documentados y reutilizables por otros subsistemas de la aplicación u otros sistemas. De aquí, que la arquitectura al estar basada en componentes proporcione la herramienta fundamental para lograr las expectativas de reusabilidad y reutilización de las partes de software favoreciendo en gran medida la extensibilidad. Concretamente se utilizó el diseño orientado a componentes y el uso de frameworks en la propuesta de arquitectura y el desarrollo de la solución.

Patrones

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

En Pattern Oriented Software Architecture (POSA), se definen los patrones de arquitectura como “un esquema de organización estructural para los sistemas de software que proporcionan un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y lineamientos para organizar la relación entre ellos” (Buschmann, et al., 1996). Respecto a la discusión que concierne la diferencia entre patrones y estilos Microsoft adopta el criterio de que “los patrones se refieren más bien a prácticas de re- utilización y se encuentran más ligados al uso y al plano físico, mientras los estilos conciernen a teorías sobre la estructura de los sistemas a veces más formales que concretas, enfatizando descriptivamente las configuraciones de una arquitectura, desarrollando incluso lenguajes y notaciones capaces de expresarlas formalmente” (Reynoso, et al., 2004)

Los patrones de diseño, al contrario de los estilos arquitectónicos, son muchos y muy variados y es casi imposible revisar todos los que existen a la hora de hacer una determinada aplicación, por eso se recomienda el uso de los patrones que estén predeterminados en cada uno de los estilos que se seleccionen para la arquitectura. En el desarrollo de aplicaciones enterprises son carácterístico el uso de patrones específicos asociados a las diferentes capas (presentación, negocio, acceso a datos), a continuación se presentarán algunos de estos patrones fundamentalmente los utilizados en la aplicación

(sobre todo asociados con el trabajo del framework Symfony que como se verá adelante es por el que se ha optado):

Capa de Presentación

Observer (Observador): clasifica como patrón de comportamiento a nivel de objeto, el problema que lo motiva es la existencia de diferentes objetos desacoplados que deben mantenerse actualizados del estado de otro objeto, o de los determinados sucesos que ocurren en él, su propósito es garantizar que los interesados en el estado del objeto observado sean notificados convenientemente. Es ampliamente utilizado en el diseño de componentes de interface de usuario de la capa de presentación (Gamma, y otros, 1994).

Capa de Negocio

Front Controller (Controlador frontal): un controlador concentra todas las solicitudes de la aplicación delegando la en el manejador específico que generalmente es un comando (Fowler, y otros, 2002).

Singleton (Instancia única): es un patrón creacional a nivel de objetos, el problema que lo motiva es la necesidad de tener una instancia única de un objeto, su propósito es garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma (Gamma, y otros, 1994).

Command (Orden): tiene por propósito encapsular en un objeto la acción que satisface una petición (Gamma, y otros, 1994). En el framework Symfony las acciones son un ejemplo de este patrón.

Intercepting Filter (Filtros Interceptores): tiene por objetivo realizar procesamientos antes y después de que se maneje la acción (Fowler, y otros, 2002). Es útil para tratar problemas de seguridad, validaciones, conversiones etc., también se implementa en Symfony.

Capa de Acceso a Datos

Los patrones asociados a la capa de acceso a datos tiene por objetivo resolver el conflito de impedancia que se da entre el modelo orientado a objetos y el modelo relacional de las bases de datos además de proveer los servicios asociados a la capa.

Active Record (Registro Activo): provee un objeto homólogo a una tupla de una entidad en la base de datos, encapsula el acceso a esta y la logíca de manipulación de los datos relativa a lainserción, carga, actualización y eliminación (Fowler, y otros, 2002) (implementado por varios frameworks entre ellos Propel).

Metadata Mapping (Mapeo de metadatos): matiene información de la realización del mapeo objeto relacional (Fowler, y otros, 2002) (implementado por Propel).

Query Object (Objeto de Consulta): encapsula la representación de una consulta a la base de la base de datos (Fowler, y otros, 2002).

Tecnología del lado del Servidor

Servidor Web Apache2

El servidor Apache es desarrollado por la Apache Software Foundation, presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido entre sus principales ventajas se encuentran: es modular, posee gran cantidad de extensiones para diversas tecnologías, libre, multiplataforma, amplia documentación (Mateus, 2004). Se señala como desventaja el que no posee interface gráfica que facilite su configuración.

Servidor de Bases de Datos PostgreSQL

PostgreSQL es un sistema gestor de base de datos objeto relacional (Object Relational Database Manager System), basado en POSTGRES Versión 4.2, desarrollado en la Universidad de California, en

el Departamento de Ciencias de la Computación de Berkeley (Equipo de desarrollo de PostgreSQL, 2005). POSTGRES es pionero en muchos conceptos que solo estuvieron disponibles en algunos sistemas de bases de datos comerciales mucho más tarde. PostgreSQL es un descendiente del "código abierto" (en inglés, open source) original de Berkeley. Soporta gran parte del SQL estándar y muchas modernas funcionalidades como: consultas complejas, llaves foráneas, disparadores (en inglés, triggers), vistas, integridad transaccional y control de versionado concurrente (MVCC en sus siglas en ingles).

Cumple completamente con las características ACID (acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español) para realizar transacciones seguras, es multiplataforma, está disponible para 34 plataformas en su última versión estable. Con la integridad referencial y posee interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG; PYTHON y RUBY, además de traer soporte para la herencia y la seguridad de la capa de dispositivo de transportación de datos (SSL, Secure Sockets Layer). Además, PostgreSQL puede ser personalizado por el usuario en muchas formas, según sus necesidades, por ejemplo adicionando, nuevos tipos de datos, funciones, operadores, funciones agregadas entre otros. PostgreSQL puede ser utilizado, modificado y distribuido por cualquiera gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos.

Lenguaje de Programación PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre. Es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas. El 13 de julio de 2004, fue lanzado PHP 5, utilizando el motor Zend Engine 2.0 (o Zend Engine 2). La

versión más reciente de PHP es la 5.2.9-2 (8 de abril de 2009), que incluye todas las ventajas que provee el nuevo Zend Engine 2.

Entre las principales características que posee PHP podemos encontrar:

- Lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de extensiones.
- Posee una amplia documentación en su página oficial entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- Tiene manejo de excepciones (desde PHP5).

Tecnologías del lado del Cliente

AJAX

AJAX es un acrónimo de “Asynchronous Java Script and XML” o Java Script Asíncrono y XML. No es en sí misma una tecnología sino la conjunción de varias que dan lugar a un nuevo tipo de aplicaciones web, las aplicaciones enriquecidas para internet, por sus siglas en inglés RIA de “Rich Internet Applications”, las tecnologías que forman AJAX (Fig. 2) son:

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

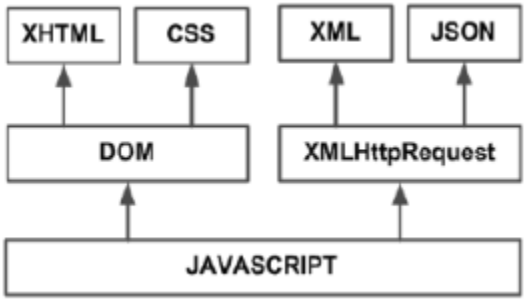


Fig. 2 Tecnologías asociadas a AJAX

En las aplicaciones web tradicionales, las acciones del usuario en la página desencadenan peticiones al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario y este procede a actualizarse completamente. Una comparación de los elementos estructurales vinculados a esta tecnología se muestra en (Fig. 3) tomada de (Eguilúz Pérez, 2008) se puede observar como en el modelo con AJAX entre las peticiones de la interface de usuario media el motor de AJAX el cual es quien envía una petición concreta al servidor web quien a su vez responde datos en XML (o JSON en algunos casos) quedando luego del lado de la aplicación la responsabilidad de actualizar únicamente la zona que corresponde.

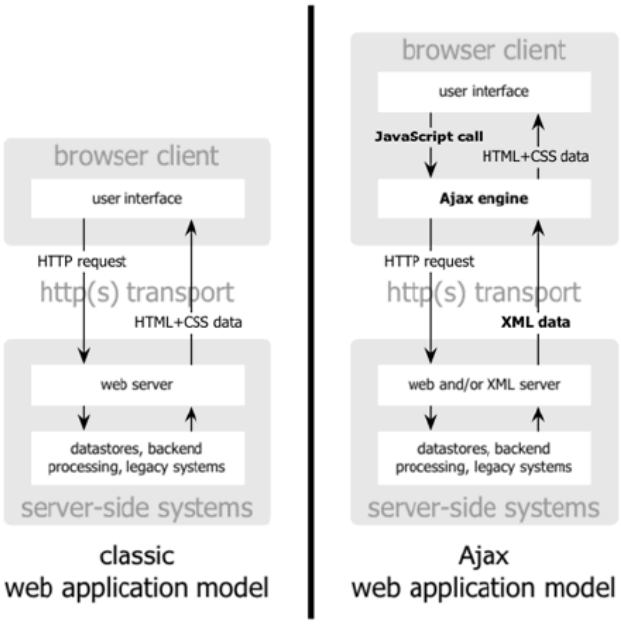


Fig. 3 Comparación entre el modelo tradicional de aplicaiones web y el modelo utilizando AJAX

La diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX está en la interacción síncrona propia de las aplicaciones web tradicionales respecto a la

comunicación asíncrona de las aplicaciones creadas con AJAX (Fig. 4). Las peticiones HTTP al servidor se sustituyen por peticiones Java Script que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor por lo que se potencia la usabilidad al mejorar la calidad de la interacción.

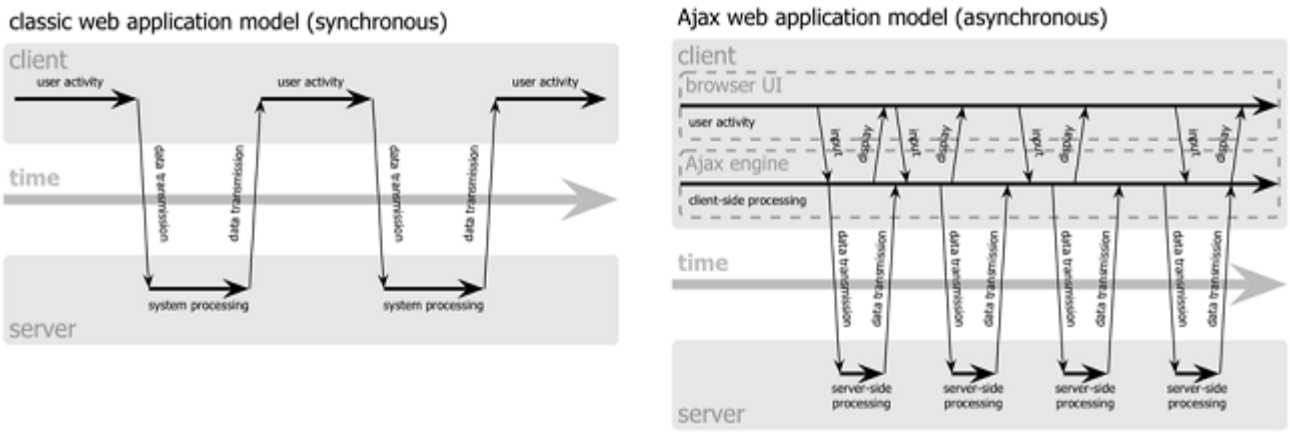


Fig. 4 Comunicación síncrona vs. asíncrona de las aplicaciones con AJAX

Desde su aparición, se han creado cientos de aplicaciones web basadas en AJAX. Además, en el caso de las aplicaciones web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio como que se pretende en este trabajo, gracias a las posibilidades del API Ext JS. Se impone entonces de manera desglosada revisar las diferentes tecnologías que conforman el espectro de las aplicaciones AJAX.

XHTML

HTML son las siglas de *Hyper Text Markup Language* o Lenguaje de Marcación de Hipertexto, un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium (W3C). XHTML es una versión avanzada de HTML y basada en XML adoptando las restricciones de este último permitiendo su validación a través de un Shema o una DTD.

La primera versión de XHTML se denomina XHTML 1.0 y se publicó el 26 de Enero de 2000 (y posteriormente se revisó el 1 de Agosto de 2002). La versión XHTML 1.1 ya ha sido publicada en forma de borrador y pretende modularizar XHTML. También ha sido publicado el borrador de XHTML 2.0, que supondrá un cambio muy importante respecto de las anteriores versiones de XHTML.

CSS

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilos creado para controlar el aspecto de los documentos electrónicos definidos con HTML y XHTML, como lenguaje su objetivo radica en especificar los detalles asociados a la presentación lográndose separar los contenidos de su presentación. Esta separación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos").

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. La creación de CSS estuvo a cargo de un grupo de trabajo conformado por el W3C que entre 9 propuestas originales consideró finalmente 2, CHSS (Cascading HTML Style Sheets) y SSP (Stream-based Style Sheet Proposal) realizadas por Håkon Wium y Bert Bos respectivamente. Entre finales de 1994 y 1995 Lie y Bos se unen para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta, resultando finalmente CSS y publicándose a finales de 1996 la primera recomendación oficial "CSS nivel 1".

El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como "CSS nivel 2". Al mismo tiempo, la siguiente recomendación de CSS, conocida como "CSS nivel 3", continúa en desarrollo desde 1998 y hasta el momento sólo se han publicado borradores. La versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que aún se está elaborando.

Java Script

Java Script fue creado por Brendan Eich de Nescape Comunications Corporation bajo el nombre de "Mocha", hizo su primera aparición en el explorador de internet Navigator 2.0 de esta misma compañía, posteriormente sería renombrado a LiveScript hasta llegar al actual Java Script. Aparece en el Internet Explorer de Microsoft a partir de su versión 3.0 con el nombre de Jscript y actualmente es un estándar, el ECMA 262 en su 3ra Edición o ECMAScript quedando normados los principales elementos de este, la 4ta edición del estándar promete funcionalidades verdaderamente revolucionarias.

Las principales características de Java Script como lenguaje son:

- **Interpretado**

- **Estructurado:** provee todos los recursos de un lenguaje estructurado como C (funciones, ciclo por conteo y por condición, condicionales, condicionales múltiples, y demás estructuras)

- **Tipado Dinámico:** como la mayoría de los lenguajes scripts los tipos de las variables están asociados al valor que contiene en un momento dado.

- **Orientado a Objetos:** implementa una variante del orientado a objetos no basada en clases propiamente sino en objetos prototipos a través de los cuales se pueden crear otros objetos idénticos y extender sus funcionalidades.

En el principio Java Script se tomó como un lenguaje torpe para desarrollos pesados, fundamentalmente porque cada navegador implementa diferentes características del estándar, por todo esto se relegó su utilidad a simples efectos visuales y a la validación de formularios. Tal realidad parece que está cambiando rápidamente a tal punto que ya se habla de un paradigma en el uso de Java Script conocido como Java Script No Obstructivo y aunque no se dispone de una definición concreta se plantean principios a seguir a la hora de desarrollar aplicaciones serias en lo que a portabilidad refiere. La aparición de tecnologías como AJAX vienen a complementar junto al acceso al DOM que desde hace tiempo provee el lenguaje, las posibilidades de desarrollo de aplicaciones enriquecidas para la

web, de esta forma se da el salto de un modelo sincrónico a disponer ahora, con la utilización de AJAX, de aplicaciones asíncronas, que por sus características llevan la interacción con la web a otra dimensión.

Un ejemplo concreto de esto es la proliferación de un conjunto de librerías y APIs que extienden los recursos del lenguaje e incorporan otros, facilitando en gran medida el desarrollo de aplicaciones cada vez más complejas, tales son los casos de Ext JS, Dojo, Backbase, jQuery, Mootools, Prototype, Yahoo! UI Library y Script.aculo.US. Particularmente Ext JS como se verá más adelante resulta el más maduro de estos frameworks ya que va a sintetizar las mejores experiencia de sus predecesores entre los cuales está Prototype, jQuery, Script.aculo.us y YUI.

XML

XML son las siglas de Extensible Markup Language o Lenguaje de Marcación Extensible desarrollado actualmente por el W3C. Es utilizado para el intercambio de datos entre diferentes plataformas, pero de forma general engloba un conjunto de tecnologías desarrolladas sobre este para el tratamiento de datos, siendo su principal característica la de describir la información que contiene.

El desarrollo de XML comenzó en 1996 en un grupo de trabajo creado por el W3C estando la primera versión lista en 1998, desciende de SGML Standard Generalized Markup Language o Lenguaje General de Marcas (ISO 8879:1986 SGML) quien a su vez fue desarrollado en la década del 60 por Charles Goldfarb, Edward Mosher and Raymond Lorie de IBM.

Tiene innumerables usos desde la transmisión de noticias vía RSS hasta la transmisión de información científica como es el caso de CML Chemical Markup Lenguage. En el marco de la descripción de la arquitectura adquiere importancia como formato para la exportación de información a través de mecanismos de serialización.

JSON

Notación de Objetos de Java Script (Java Script Object Notation por sus siglas en inglés) es una notación muy simple para definir objetos en Java Script, representa una alternativa al intercambio de información respecto a XML al resultar más ligero, a ello se suma que es más fácil su interpretación al poder ser decodificado directamente en un objeto.

Marcos para el desarrollo de Aplicaciones Web

Un marco o framework, es una estructura conceptual básica usada para resolver un problema complejo. Los marcos de desarrollo de aplicaciones pueden estar basados en un conjunto de conceptos, modelos, metodologías, componentes, herramientas de administración y de diseño, entre otras; de manera que facilite la construcción de las aplicaciones, manteniendo el orden y la homogeneidad en las partes y piezas que conforman la aplicación. Fuertemente reutilizables están diseñados para ser extendidos lo que se expresa como un conjunto de clases abstractas y el modo en que sus instancias colaboran para un tipo específico de software. Un marco de desarrollo puede incluir programas de soporte, librerías de código, códigos script, u otras aplicaciones que faciliten el desarrollo. Para las aplicaciones desarrolladas en PHP existe una cantidad considerable de frameworks libres de mucha calidad y renombre que pueden ser utilizados entre los que se pueden encontrar a Zend Framework, y Symfony entre muchos otros como CakePHP, Code Igniter y PHP4EECore, sin embargo en el marco de este trabajo solo se consideran los dos primeros sobre los cuales se posee ya una experiencia importante, al respecto se tiene en cuenta que el desarrollo del ERP utiliza Zend Framework con algunas extensiones locales, en tanto el Generador de Reportes Dinámicos fue desarrollado con Symfony, ambas soluciones con las que debe integrarse nuestro sistema. Es necesario tener en consideración además que el desarrollo de aplicaciones enriquecidas para la web nos obliga a tener un grueso del software del lado del cliente, en este caso se requiere frameworks de componentes visuales y AJAX sobre la web existiendo entre los más sobresalientes Ext JS, Yahoo User Interface (YUI) entre otros muchos como JQuery, Dojo, Mootools, Prototype, Script.aculo.US con diferentes usos, en este sentido solo se va a analizar Ext JS.

Zend Framework (ZF)

Zend Framework es un marco de trabajo de código abierto orientado a objetos, implementado en PHP 5 y bajo la licencia BSD. Tiene como objetivo simplificar el desarrollo web encapsulando al mismo tiempo las mejores prácticas de programación en PHP, y provee componentes para el patrón MVC. Entre los componentes que proporciona se encuentran aquellos que se utilizan con frecuencia en aplicaciones web, incluyendo la autenticación y autorización a través de listas de control de acceso (ACL), la configuración de aplicación, los datos de la memoria caché, el filtrado y validación de los datos proporcionados por el usuario, la internacionalización, las interfaces para AJAX, así como la composición y entrega de correo electrónico.

Symfony

Symfony es un framework bajo licencia MIT que sigue el patrón arquitectónico MVC para desarrollar aplicaciones web comerciales, gratuitas y/o de software libre escritas en PHP5. Tiene como objetivo acelerar la creación y mantenimiento de aplicaciones web y sustituir repetitivas tareas de codificación; proporcionando para esto varias herramientas que siguen la mayoría de mejores prácticas y patrones de diseño para la web. Este framework es fácil de instalar y extender, lo que permite su integración con librerías desarrolladas por terceros entre ellas Zend Framework. Presenta también herramientas de configuración que lo hacen lo suficientemente flexible como para adaptarse a las necesidades de la aplicación que se desea desarrollar. Otras de las características de Symfony es que brinda soporte a la internacionalización y es independiente del SGBD algo que se garantiza a través de un ORM que por defecto trae a Propel pero que también puede utilizarse Doctrine (incluido también a partir de la versión 1.2). Se puede ejecutar tanto en plataformas Unix y Linux, como en plataformas Windows. Su código fuente incluye más de 8.000 pruebas unitarias y funcionales, y ha sido utilizado en numerosos proyectos reales como Yahoo Bookmarks y Yahoo Answers. Las aplicaciones que emplean Symfony, pueden controlar hasta el último acceso a la información, debido a que el framework brinda herramientas para ello, e incluye por defecto protección contra ataques XSS. La empresa que lo ha creado está comprometida con el framework, ya que no se sustenta directamente del mismo, sino de las aplicaciones que crea con él; por tanto dicha empresa se interesa por aspectos como el rendimiento, la buena documentación, y un soporte a largo plazo. Posee una extensa comunidad que garantiza el mejor

una excelente asistencia técnica, soporte y desarrollo de múltiples extensiones. Las versiones estables del framework se mantienen durante tres años sin cambios, pero con una continua corrección de los errores conocidos. Además, Symfony cuenta con una documentación muy amplia, que incluye miles de páginas en el sitio oficial, una guía gratuita traducida al español conformada por más de 400 páginas a lo que se le suma excelentes tutoriales que permiten en principio la asimilación de los conceptos principales en 24 horas.

Concluido el análisis anterior, se selecciona como framework para el desarrollo a Symfony.

Ext JS

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

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

La versión 2.0 fue liberada el 4 de diciembre de 2007. Se caracteriza fundamentalmente por proveer un set de componentes que permiten el desarrollo de aplicaciones web muy similares a las de escritorio. Posee una buena documentación y es posible accede a muchos ejemplos de su uso directamente incluidos en la documentación. Si bien es una librería con elementos de mucha calidad aún no posee una herramienta lo suficientemente madura para un diseño fácil de las interfaces de usuario por tanto su principal ventaja es que se debe programar a mano el diseño de estas.

Metodologías de Desarrollo de Software

El Proceso Unificado de Rational (RUP) es una metodología que define un proceso de desarrollo de software. Como metodología es una plantilla de proceso que requiere necesariamente ser configurado para proyectos específicos tarea que realiza el rol del ingeniero de procesos. Los principios que rigen este proceso establecen que es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Es una metodología orientada a objetos que utiliza a UML como lenguaje de modelado.

Según el Proceso Unificado de Desarrollo de Software la arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por plataformas software, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Se representa mediante varias vistas que se centran en aspectos concretos del sistema. (Ayuda Rational, 2003)

Todas las vistas juntas forman el llamado modelo 4+1 de la arquitectura, recibe este nombre porque lo forman las vistas lógica, de implementación, proceso y despliegue, más la de casos de uso que es la que da cohesión a todas. La metodología de desarrollo RUP plantea que el rol de arquitecto es el responsable por el desarrollo y mantenimiento de la arquitectura de software, la que incluye como hemos visto las principales decisiones técnicas y las restricciones sobre el diseño e implementación del proyecto.

Las principales actividades que realiza el arquitecto según RUP son:

Priorizar los Casos de Uso: estableciendo que casos de usos son críticos, secundarios, auxiliares u opcionales, lo que permite definir los módulos, subsistemas y escenarios así como la interacción entre ellos, que permita tomar decisiones hacia donde centrar los esfuerzos de implementación.

Análisis de la arquitectura: para establecer la arquitectura candidata del sistema teniendo en cuenta arquitecturas similares u otros sistemas, definir además los estilos arquitectónicos, patrones de arquitectura, los principales mecanismos de diseño arquitectónico.

Identificar mecanismos de diseño: refina el análisis de la arquitectura teniendo en cuenta las restricciones impuestas por el entorno de implementación.

Estructurar el modelo de implementación: para establecer la estructura en la que va a residir la implementación del sistema.

Reutilización de elementos de diseño existentes: analiza las interacciones en los diagramas de clases del análisis para encontrar interfaces, diseño de clases y subsistemas. Refinar la arquitectura e incorporar elementos reutilizables si es posible, identificar problemas comunes a los que se pueda crear soluciones generales (patrones, o familias de productos).

Identificar los elementos de diseño: analizar las interacciones entre las clases de análisis para identificar los elementos de modelo de diseño arquitectónico.

Describir la arquitectura en tiempo de ejecución: analizar requerimientos de concurrencia, identificar los procesos y la comunicación entre dichos procesos, identificar el ciclo de vida de dichos procesos.

Los principales artefactos a desarrollar son:

- Documento Descripción de la Arquitectura (4 Vistas + 1)
- Modelo de Despliegue
- Modelo de Implementación
- Protocolos
- Interfaces

En el desarrollo del proyecto se utiliza RUP por varias razones:

- Posee una vasta documentación y está incorporado a los programas de estudio de la universidad por lo que resulta familiar a todos los miembros del proyecto.
- Existen varias herramientas para soportar el modelado con UML tanto libres como propietarias.

- Provee un marco de buenas prácticas de gestión de proyecto
- Centra el proceso de desarrollo en la arquitectura definiendo para ella actividades y artefactos. Además es la metodología que desarrolla la tendencia arquitectónica fundamental por la que optó en el presente trabajo.

Herramientas

Eclipse

Eclipse es un entorno de desarrollo integrado (IDE) libre, fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Actualmente es mantenido por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Una de las características más importantes de Eclipse es su modularidad, pues se pueden añadir plugins según las necesidades de cada desarrollador. Una gran cantidad de estas extensiones pueden ser comúnmente encontradas en internet, tanto libres como comerciales de esta manera se puede disponer de una herramienta adaptada a las necesidades.

Zend Studio for Eclipse

Zend Studio for Eclipse fue creado por Zend Technologies Inc. con el objetivo de aprovechar las bondades de Eclipse unida a las potencialidades de su propia plataforma el Zend Studio, incorpora múltiples ventajas fundamentalmente asociadas a las posibilidades de explotación de las tecnologías propias de Zend entre ellas el Zend Framework. Se distribuye comercialmente como software privativo.

Netbeans 6.5

Netbeans es un IDE de desarrollo distribuido por SUN Microsystems con licencia dual (GPL y CDDL[2]). En su versión 6.5 se ha incorporado a los lenguajes de trabajo PHP y en su próxima versión se ha prometido incluir a Symfony como framework por defecto. Netbeans 6.5 posee múltiples ventajas entre ellas se destaca:

- Soporte a Java Script.
- Interprete de Fondo (Background Parser) capaz de identificar errores sintácticos en tiempo de edición.
- Completamiento de código.
- Marcado sintáctico que presenta en diferentes estilos de letras palabras claves, identificadores estándares, y literales en general facilitando la claridad del código.
- Auto formateo.
- Soporte a JSON.
- Marcado de ocurrencias de un identificador.
- Integración con Subversión.
- Soporte a Documentación tanto para Java Script como para PHP.

Considerando que es un IDE libre además de la riqueza de prestaciones se ha optado por NetBeans 6.5 para el desarrollo.

La Calidad en la Arquitectura de Software

Cada día una industria más competitiva demanda mejores soluciones algo que va estrechamente vinculado a la garantía de la calidad que se imprime en un sistema a través de las actividades que la aseguran. Es necesario considerar entonces como desde el diseño arquitectónico se pueden tomar las primeras decisiones que determinen positivamente la calidad del sistema, para ello se impone entender ¿Qué es la calidad del software? Según (S. Pressman, 2000) la calidad en los sistemas se entiende por "la concordancia con los requisitos funcionales y de rendimiento establecidos con los estándares de

[2] Common Development and Distribution License

desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado de forma profesional". En tanto el estándar ISO/IEC 9126 – 1998 presenta la calidad como "la totalidad de rasgos y atributos de un producto de software que le apoyan en su capacidad de satisfacer sus necesidades explícitas o implícitas", por su parte otro organismo de estandarización altamente respectado como lo es la IEEE plantea que la calidad "es el grado en el cual el software posee una combinación deseada de factores". En general entre las diversas definiciones que se dan hay cierto acuerdo en cuanto a que se deben cumplir en primera instancia los requisitos del cliente y considerar un conjunto de atributos (entiéndase rendimiento, usabilidad y otros) o factores en correspondencia con estándares.

A grandes rasgos, (Camacho, y otros, 2004) referencian una clasificación de los atributos de calidad en dos categorías:

- Observables vía ejecución: aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución, entre ellos: disponibilidad, confidencialidad, funcionalidad, desempeño, confiabilidad y seguridad.
- No observables vía ejecución: aquellos atributos que se establecen durante el desarrollo del sistema a saber: configurabilidad, integrabilidad, integridad, interoperabilidad, modificabilidad, mantenibilidad, portabilidad, reusabilidad, escalabilidad y capacidad de prueba.

En principio la arquitectura influencia de manera notable los atributos de calidad (Bosch, 2000) y dado que los inhibe o facilita (Bass et al., 1998) resulta de particular interés analizar la influencia de ciertos elementos de diseño utilizados a la hora de definir la arquitectura de un sistema, ellos son los estilos arquitectónicos, los patrones arquitectónicos y los patrones de diseño.

En (Bosch, 2000) se propone 4 técnicas de evaluación de arquitecturas de software:

- Evaluación basada en escenarios: según (Kazman, 2001) un escenario consta de tres partes (el estímulo, el contexto y la respuesta) que proveen un vehículo que permite concretar y entender los atributos de calidad. Los métodos que utilizan esta técnica cuentan con dos instrumentos de

evaluación relevantes, el *árbol de utilidades* y los *perfiles* propuestos por Kasman y Bosch respectivamente.

- Evaluación basada en simulación: consiste en la implementación de componentes de la arquitectura y la implementación a cierto nivel de abstracción del contexto del sistema donde se supone va a ejecutarse con la finalidad de evaluar el comportamiento de la arquitectura bajo diversas circunstancias. En términos de los instrumentos asociados a las técnicas de evaluación basadas en simulación, se encuentran los lenguajes de descripción arquitectónica y los modelos de colas.

- Evaluación basada en modelos matemáticos: se utiliza para evaluar atributos de calidad operacionales, permitiendo una evaluación estática del diseño arquitectónico, y se presentan como alternativa a la simulación, dado que evalúan el mismo tipo de atributos (Bosch, 2000). Sobresalen dos instrumentos específicos utilizados en esta las *Cadenas de Markov* y los *Diagramas en Bloques de Fiabilidad* (Reliability Block Diagrams).

- Evaluación basada en experiencia: existe una evaluación informal, que es realizada por los arquitectos de software durante el proceso de diseño, y la realizada por equipos externos de evaluación de arquitecturas. En cualquiera de los casos se trata de la acumulación de conocimientos del dominio que ayudan a justificar decisiones de diseño considerando dicho cúmulo de experiencias. Específicamente 3 instrumentos son aplicables en esta técnica intuición y experiencia, tradición y proyectos similares.

Todo el marco de técnicas antes mencionado ha devenido en la aparición de diferentes métodos a la hora de evaluar el diseño arquitectónico. A continuación se presentan los más importantes.

Software Architecture Analysis Method (SAAM)

El Método de Análisis de Arquitecturas de Software (Software Architecture Analysis Method, SAAM) es el primero que fue ampliamente promulgado y documentado (Kazman, 2001). Ha demostrado ser muy útil para evaluar de forma rápida distintos atributos de calidad, tales como: modificabilidad, portabilidad,

escalabilidad e integrabilidad. Se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro. Como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada en tanto las salidas de la evaluación del método resultan en una proyección sobre la arquitectura de los escenarios que representan los cambios posibles ante los que puede estar expuesto el sistema y el entendimiento de la funcionalidad del sistema, e incluso una comparación de múltiples arquitecturas con respecto al nivel de funcionalidad que cada una soporta sin modificación.

Architecture Trade-off Analysis Method (ATAM)

El Método de Análisis de Acuerdos de Arquitectura (Architecture Trade-off Analysis Method, ATAM) está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM. El método se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados, los cuales representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas, entre otros (Kazman et al., 2001).

Método de Evaluación para Arquitecturas Basadas en Componentes (MECABIC)

El objetivo principal de MECABIC es evaluar y analizar la calidad exigida por los usuarios de Arquitecturas de Software Basadas en Componentes (ASBC). El método adapta diferentes elementos de algunos métodos de evaluación arquitectónica (ATAM, BOSCH, ABD, ARID, Losavio) y establece un conjunto de pasos para determinar la calidad de sistemas basados en componentes. Incluye orientaciones para generar y discutir escenarios de evaluación iniciales, y un conjunto de preguntas a partir de las cuáles estudiar las decisiones arquitectónicas consideradas.

Conclusiones Parciales

A lo largo del capítulo se realizó el estudio del estado del arte de los sistemas estadísticos, de la arquitectura de sistemas enfocado al desarrollo web, las tecnologías, herramientas y los diferentes

recursos reutilizables se valoraron entre estos recursos los más adecuados llegándose a conclusiones respecto a cueles se utilizarán. Con lo anteriormente visto se da cumplimiento a la primera tarea planteada.

CAPÍTULO 2: Propuesta de Arquitectura

En el presente capítulo se presentan la propuesta de entorno de desarrollo y la propuesta de la arquitectura del sistema a través de las vistas propuestas por RUP a la que además se le ha agregado la vista de datos.

Propuesta de Entorno de Desarrollo

La configuración del entorno de desarrollo responde a las prerrogativas de la necesidad paulatina de incorporar tecnologías soberanas al desarrollo de software en el CENTALAD, no obstante se han considerado utilizar algunas herramientas propietarias al no existir alternativas suficientemente acabadas en el mundo del software libre. Las herramientas del entorno de desarrollo son clasificadas en horizontales o de propósito general como los sistemas operativos, utilitarios generales (antivirus, correo, control y seguimiento de tareas), las aplicaciones ofimáticas y de gestión de información; y como segunda clasificación las herramientas verticales enfocadas a cuestiones específicas del desarrollo de software, a continuación se presenta de manera desglosada la propuesta.

Herramientas horizontales

Sistema operativo

En concordancia con las políticas de soberanía tecnológica del MIC el CENTALAD perteneciente a la Infraestructura Productiva (IP) de la UCI ha definido en su política de migración los lineamientos esenciales para la implantación progresiva del software libre como alternativa viable al desarrollo y como enfoque proactivo en función de implementar lo ministerialmente establecido. Por ello se utiliza Ubuntu Linux 8.10 para el desarrollo, para el sistema en elaboración la distribución cubana de Linux el SO Nova 1.0 (que se implantará en la infraestructura de clientes ligeros) y para los servidores de base de datos, web y de aplicaciones, el sistema operativo Debian 4.0.

Seguridad (antivirus, niveles de acceso a código fuente, documentación)

La seguridad es un compromiso con un fuerte componente individual, a tales efectos cada miembro del equipo de desarrollo actuará conforme está establecido y en consecuencia con el código de ética para el uso de las tecnologías. Se presta atención a las indicaciones emitidas por Dirección de Seguridad Informática de la universidad garantizando que no se violen los principios mínimos de seguridad informática exigidos por esta.

Se tiene especial atención a los principios de programación segura específicamente los relacionados al desarrollo de aplicaciones para la web con énfasis en la validación de las entradas del usuario para evitar los ataques de inyección de código SQL y de scripts, respecto al primero se utilizan las facilidades del framework Propel para construir consultas a partir de los objetos y criterios provistos por este (véase patrón de aplicación empresarial *Object Query*) evitándose construir las consultas por concatenación.

La protección del código es otro aspecto tenido en cuenta, en este caso Java Script que queda del lado del navegador en el cliente, para ello está disponible la posibilidad ofuscar y comprimir los códigos fuentes. Se optó por la herramienta libre YUICompressor 2.4.2 para ofuscar y comprimir los archivos de Java Script. La documentación del código fuente para PHP y Java Script se realizará con PHPDoc y JSDoc respectivamente ambos integrados al IDE de desarrollo. La documentación de la ayuda y los manuales de usuarios con: Open Office para los formatos odt, pdf y html.

Control de versiones

Se utiliza Subversion 1.4.2 como servidor de control de versiones con el cliente RapidSVN 0.9.6 Este servidor está desplegado como parte de la infraestructura de desarrollo del centro y al igual que la realización de las salvas automáticas quedan a responsabilidad del Grupo de Tecnología e Infraestructura del centro en conformidad con las políticas definidas para ello.

REDMINE 0.8.3 es la herramienta utilizada para la gestión del proyecto por las posibilidades que entraña además del soporte que se le está dando directamente por un equipo del centro. Para la

recolección y gestión de métricas de manera automatizada se utiliza StatSVN 0.4.1 que se integra a Subversion directamente y que es accesible desde el REDMINE, de igual manera se realiza sobre este la gestión de no conformidades y el seguimiento de los errores detectados. Para la realización de consultas reportes personalizados a las bases de datos de las herramientas se emplea el Sistema de Gestión de Reportes Dinámicos (SGRD) 1.6 elaborado por el centro.

Relativo a la gestión documental, los documentos no versionables como tutoriales, libros, informaciones del cliente y otros se localizarán centralmente en el servidor definido para ello, en tanto los versionables se mantendrá en el repositorio. No se utiliza herramienta particular para la gestión documental.

Herramientas verticales

Herramientas de modelado

Las 2 herramientas fundamentales para el modelado son la de Base de Datos y la de Ingeniería, Embarcadero ERStudio y Enterprises Architect respectivamente, como se trata de aplicaciones para Windows en ambos casos se utilizará el emulador Wine para su explotación en el entorno de Linux. La decisión de utilizar estas herramientas responde a:

- Se posee fuerte experiencia en ellas del anterior proyecto desarrollado por el equipo.
- La mayoría de los diagramas del expediente anterior sobre los que se trabajará nuevamente están elaborados sobre estas herramientas.
- Los artefactos de ingeniería fundamentales se generan directamente utilizando las capacidades de Enterprises Architect con las adecuaciones al expediente de proyecto definido por la dirección de calidad de la UCI.

Compilación (compilador, maquina virtual, interprete)

Se utilizan 2 lenguajes interpretados, del lado del servidor PHP cuyo intérprete es el motor Zend de PHP 5.2.6, del lado del cliente el intérprete por defecto de Mozilla Firefox Gecko aunque se garantiza la portabilidad del código para cualquier interprete que cumpla con las especificaciones del estándar ECMA 262 que norma al ECMAScript.

Prueba (Pruebas Unitarias)

Se utiliza el framework LIME incorporado a Symfony para la implementación y realización de pruebas unitarias. Con el objetivo de probar los componentes del lado del servidor (fundamentalmente clases) los propios implementadores desarrollarán los scripts de prueba una vez definidas las interfaces de estas.

Entornos de desarrollo integrados

Se utiliza como IDE Netbeans 5.6. Para la visualización se utiliza Mozilla Firefox con el plugin Firebug para el entorno de desarrollo y prueba, para el entorno de explotación Mozilla Firefox únicamente. Se utiliza Symfony y los recursos que este incorpora para la realización de las pruebas funcionales desde la consola de PHP.

Lenguajes de programación

Se utilizan los siguientes lenguajes para el desarrollo de la aplicación:

- Se utilizan PHP 5.0 con las extensiones php5-pgsql, php5-xsl, y php-gdi.
- Java Script (ECMAScript 262 3ra Edición)
- XSLT para el procesamiento de XML.

Framework o Componentes

Ext JS 2.2 es un framework de Java Script para AJAX basado en Yahoo User Interface (YUI) y con interoperabilidad para Prototype, jQuery y Script.aculo.US con excelentes prestaciones visuales y de

interacción. Las características de Ext JS que argumentan su uso se discutieron en el capítulo 1. OpenJacob es un framework para dibujo con Java Script que utiliza a Mootools. Entre las ventajas que justifican su utilización esta que es libre, extensible, con un diseño robusto y optimizado, facilitando el trabajo con espacios de diseños utilizando el estilo modelo - vista - controlador. El marco UCI.webApp un componente desarrollado por el proyecto para facilitar la estructuración de la aplicación y ordenar el desarrollo que también se incorporó a la solución.

El Generador de Reportes Dinámicos como componente se trata de una aplicación completa desarrollada por el CENTALAD que se utiliza como base del desarrollo a la vez que sustituye el MGR anterior existente en SIGE 1.0. Otros componentes importantes incluidos dentro de este son PHPReport 0.4.9 y pChart 1.27 que permiten la generación de reportes y de gráficos respectivamente.

Se utilizó Symfony 1.1 a pesar de que ya está disponible la versión 1.2 del framework algunos plugins utilizados por el Generador de Reportes Dinámicos se comportan inestables con la nueva versión por los que se opta por mantener 1.1. A su vez se trabaja con el ORM Propel que por defecto se incluye, adoptando este a Creole como capa de abstracción de base de datos que permite la independencia del tipo de gestor donde reside la información, Creole soporta las principales bases de datos entre ellas SQL Server, Oracle, MySQL y PostgreSQL.

Servidor de Aplicaciones

Como se analizó en el capítulo 1 el servidor Web Apache 2.0 con PHP 5 asume la tarea de servidor web y de aplicación al mismo tiempo. Entre las principales ventajas de Apache se encuentra que es modular, posee gran cantidad de extensiones para diversas tecnologías, libre, multiplataforma, y dispone de abundante documentación. Probablemente el servidor web más popular aún frente a la las alternativas propietarias, ya que además de mantenimiento que le da la fundación del mismo nombre se consigue soporte casi instantáneo gracias a la amplia comunidad de usuarios.

Sistema gestor de bases de batos (servidor, cliente, o modelo propio de persistencia)

Se utilizará PostgresSQL 8.2 como gestor dado que completamente con las características ACID (acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español) para realizar transacciones seguras, es multiplataforma soporta la integridad referencial y posee interfaces nativas para múltiples lenguajes entre ellos PHP, trae soporte para la herencia y la seguridad de la capa de dispositivo de transportación de datos (SSL, Secure Sockets Layer) y como cliente PgAdmin III 1.8.4 que es distribuido junto a PostgreSQL por demás un cliente de entorno gráfico de alta calidad y libre.

Propuesta de Arquitectura

Módulos que Componen la Solución

El sistema concebido responde a los procesos fundamentales identificados en el marco de gestión de la información estadística. Tales procesos comprenden la creación del formulario, la captura de la información mediante el formulario que sirve de soporte físico, la validación, el almacenamiento en la base de datos, su traspaso al almacén de datos y posteriormente su consulta y procesamiento (ya sea desde el almacén o directamente desde la base de datos). Por tal razón se ha diseñado una arquitectura modular (Fig. 5), fomentando atributos como la alta cohesión, el bajo acoplamiento, y la flexibilidad. De esta forma se garantiza un sistema adaptable a diferentes ámbitos del negocio estadístico, que no solo podría ser útil para la ONE sino además soportar negocios con una base metodológica (de trabajo estadístico) similar.

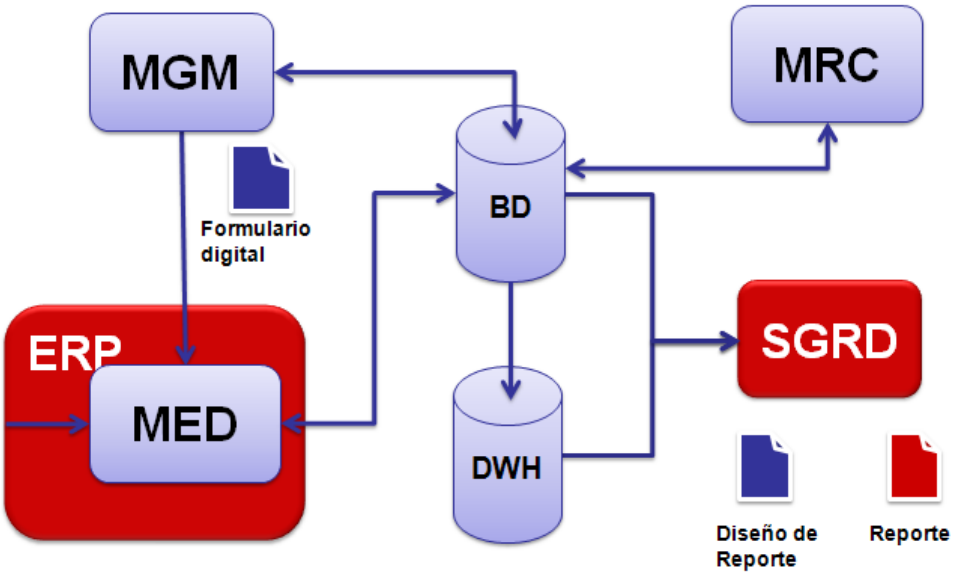


Fig. 5 Módulos de PATDSI - SIGE

Módulo Generador de Modelos (MGM)

Su función comprende la creación del modelo estadístico distinguiéndose de estos 2 tipos: los formularios de la estadística continua y los formularios de la estadística periódica. Además permite la elaboración de la estructura semántica y las validaciones de la entrada de la información. Destaca entre sus atributos el bajo acoplamiento (está vinculado como aplicación web a la base de datos pero es independiente de los otros módulos y puede existir separado de estos), la alta cohesión (su resultado es un modelo digital que reside en la base de datos o puede ser exportado a XML) manifiesta su autonomía en lo que respecta a su función, y la generalidad (al permitir la elaboración dinámica de los modelos estadísticos independientes de su complejidad).

Módulo de Entrada de Datos (MED)

Garantiza la captura, validación y posterior almacenamiento en la base de datos. Consume el modelo digital generado por MGM desde la base de datos o importándolo desde un XML previamente. De manera similar da soporte al bajo acoplamiento manifestado en su dependencia exclusiva de la base de datos, a la alta cohesión, a la generalidad al permitir la captura, validación y almacenamiento de la

información en función del modelo específicamente elaborado, e interoperabilidad con el ERP cubano si se configura el modo de recuperación de la información estadística depositada en este, característica importante al disminuir considerablemente la carga de trabajo que por concepto de digitación tienen los trabajadores de la ONE tarea por demás tediosa y deshumanizante.

Módulo de Registros y Clasificadores (MRC)

Gestiona los atributos de interés para el posterior trabajo estadístico, ello comprende los indicadores y la caracterización/clasificación de los individuos que conforman el universo o la muestra del estudio. Para el caso cubano, por ejemplo, estos individuos responden a los centros informantes (empresas, cooperativas, organismos, organizaciones de masa y otros), que se caracterizarían por su ubicación geográfica (distribución político-administrativa), forma de financiamiento, ministerio y demás atributos de interés al registro, permitiendo con ello seleccionar del universo en conjunto, una población con determinadas características. Da soporte al bajo acoplamiento, a la alta cohesión, a la generalidad, indirectamente relacionado al ERP porque este contiene los mismos atributos (nomencladores) asociados a las empresas, dado que responde a un convenio nacional.

Sistema de Gestión de Reportes Dinámicos (SGRD)

El SGRD es una solución integral a la elaboración, generación y gestión de reportes desarrollado por el Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD) de la UCI da respuesta a los requisitos de consulta y procesamiento inicialmente solicitado, además incluye un valor agregado relativo la gestión de estos reportes que comprende entre otras posibilidad la distribución automatizada de los mismos.

Objetivos y Restricciones de la Arquitectura

Requerimientos de hardware

El sistema desarrollado debe adaptarse a las características del hardware disponible y del adquirido por la ONE para su informatización, a tales efectos se distinguen los siguientes elementos con sus respectivas restricciones:

1. Servidor de Base de Datos: procesador Intel Pentium 4 1.7 GHz o AMD equivalente, 512 MB de RAM, 80 GB de espacio en disco duro.
2. Almacén de Datos: (obtener estimación de la tesis de Julio)
3. Servidor web[3]: procesador Intel Pentium 4 1.7 GHz o AMD equivalente, 512 MB de RAM, 40 GB de espacio en disco duro.
4. Estación de trabajo[4]: procesador Intel Pentium 4 1.7 GHz, o AMD similar, 256 MB RAM, 20 GB de espacio en disco duro.

Requerimientos de software

Servidor de Base de Datos

Tanto en las ONE municipales, provinciales y nacional:

1) Sistema Operativo: Debian 4.0 o Ubuntu 8.10
2) Gestor de Base de Datos: PostgresSQL 8.2
3) Cliente de Base de Datos: PgAdmin III 1.8.4

Almacén de Datos[5]

[3] Es probable que en la mayoría de los casos el servidor de base de datos y el de la aplicación web coincidan, aunque en principio no es lo que se recomienda como se verá el entorno de despliegue de la aplicación pro niveles dada la cantidad de usuarios a cada nivel no entraña problemas de rendimiento al respecto.

[4] Una variante serían los clientes ligeros adecuadamente configurados

[5] Se hace referencia al Almacén de Datos para la ONE porque será parte de la solución integral, pero no pertenece a este proyecto específicamente

Para la ONE nacional:

1. Sistema Operativo: Debian 8.2
2. Data warehouse: Pentaho - Mondrian 3.0.4 y demás herramientas de la suite Pentaho (Pentaho Data Integrator, Pentaho Business Inteligent, Pentaho Report Designer, Pentaho Work Bench y JRuby)

Servidor Web

Tanto en las ONE municipales, provinciales y nacional:

1) Sistema Operativo: Debian 4.0 o Ubuntu 8.10
2) Aplicación Servidora: Apache 2 (o superior) con PHP5 (o superior)
3) Sistema Integrado de Gestión Estadística 2.0 Nuragas (PATDSI - SIGE)
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)

Estación de Trabajo

Tanto en las ONE municipales, provinciales y nacional:

1) Sistema Operativo: Nova 1.0 o Ubuntu 8.10
2) Navegador: Mozilla Firefox 3.0.3

Requerimientos de red e Interoperabilidad con otros sistemas

El sistema debe funcionar en redes de bajas prestaciones en velocidad y ancho de banda. Respecto a la interoperabilidad debe permitir la recuperación de información desde el ERP y la importación y exportación a MicroSet NT.

Estándares de datos

1. El sistema debe garantizar la importación y exportación de información. Para ello se exportarán los modelos en archivos XML, se asumirá el esquema del modelo de datos como estándar de facto para el sistema y posteriores.

2. Puesto que durante mucho tiempo la ONE ha trabajado con MicroSet NT es este en la actualidad el estándar de la información estadística y aunque la tendencia a su desaparición con el nuevo sistema es un hecho debe garantizarse que el SIGE soporte el formato de salida y/o entrada de MicroSet NT.

Distribución geográfica del sistema

Dada la estructura organizacional de la institución el sistema se encuentra íntegramente en todos los municipios y provincias del país incluyendo Isla de la Juventud, esto se justifica fundamentalmente por prestaciones de red heterogéneas. Se tiene en cuenta varios escenarios de despliegue considerando que en un futuro, cuando las condiciones lo permitan, no sea necesario desplegar todos los componentes en cada ONE municipal. Cada ONE municipal, ONE provincial, y la ONE debe poseer íntegramente una base de datos con toda la información a su correspondiente nivel. Teniendo en cuenta la integración con el ERP Cubano, a lo largo del municipio debe ser accesible a todas las entidades el módulo de entrada de datos para recuperar la información estadística de los registros de cada uno de los subsistemas del ERP y transferirla directamente a las ONE municipales, por tanto este módulo y sus dependencias estarán donde se encuentre físicamente el ERP. Otra variante es que algunos centros informantes que no tengan o no necesiten del ERP pudieran tener el sistema con el fin de cumplir su convenio de entrega de información estadística. Teniendo en cuenta lo antes visto se tiene:

1. El sistema se encuentra íntegramente en cada una de las ONE municipales, en las ONE provinciales y en la ONE.

2. Algunos componentes del sistema estarán ubicados dentro del ERP, específicamente el de entrada de datos (y las dependencias de este) para la captación de la información estadística de la entidad.

3. El sistema podría estar en manos de terceros tal es el caso de los centros informantes que dispongan de los recursos para su implantación en respuesta a intereses particulares.

Seguridad

1. La seguridad debe ser gestionable a través del componente de seguridad desarrollado por el Centro de Compatibilización para Intereses de la Defensa. Este componente (A.A.A.) es utilizado por el ERP y garantiza la autenticación, la autorización y la auditoría.

Portabilidad, escalabilidad, y reusabilidad

1. El sistema debe ser multiplataforma (garantizado por las tecnologías que utiliza).

2. El sistema debe hacer un uso racional de los recursos de hardware, especialmente en las PC clientes.

3. Debe estar fuertemente orientado a componentes, donde se identificarán los componentes altamente reutilizables y los que son propios del negocio.

4. La documentación de la Arquitectura debe ser reusable de forma que pueda utilizarse como arquitectura tipo para sistemas similares que el CENTALAD potencia.

Restricciones del diseño

Tanto el diseño para PHP como para Java Script se debe hacer orientado a objetos a fin de potenciar todas las ventajas que este paradigma implica.

Se utilizará la tecnología que ha venido desarrollando el centro, los frameworks de presentación, lógica de negocio y acceso a datos: Ext JS, Symfony y Propel respectivamente.

Tamaño y rendimiento

Entre las múltiples posibilidades de PostgreSQL están el Control de Concurrencia Multiversión (MVCC), la creación de puntos de recuperación, los espacios de tablas, replicación asincrónica, transacciones embebidas, resguardos en línea y en caliente, planificador y optimizador de consultas y un completo sistema de registros para la tolerancia a fallos, el rendimiento y la integridad de los datos está asegurada, permitiendo que las bases de datos no tengan límite de tamaño, las tablas pueden ser de hasta 32 TB (TeraBytes) de información, se puede almacenar hasta 1.6 TB por tupla, 1 GB por campo, la cantidad de tuplas por tabla no tiene límites, el límite de columnas por tabla es de entre 250 y 1600 dependiendo del tipo de las columnas y se pueden crear tantos índices como se desee. El sistema hace uso de las principales capacidades del gestor, así en las estimaciones realizadas para almacenar la información relativa a un año se llegará en una tabla a más de un millón de tuplas, considerando el rendimiento asociado a consultas se mantienen índices que optimizan estas operaciones. El diseño del modelo de datos permite almacenar de manera óptima la información estadística prevista con el menor uso de espacio en disco posible. Finalmente se tuvo en consideración el costo de mantener la información por más de un año y al respecto se decidió que esta debe pasar al cabo de este período al almacén de datos de la ONE.

Representación Arquitectónica

Se presenta la Arquitectura fundamentalmente a través de las vistas propuestas por RUP utilizando UML 2.0:

- Vista de Casos de Usos.
- Vista Lógica.
- Vista de Implementación.
- Vista de Despliegue.

- Vista de Datos.

Adicionalmente en la sección estilos arquitectónicos y patrones de arquitectura se utilizan esquemas para representar gráficamente algunos conceptos manejados en la arquitectura del software, fundamentalmente a la hora de explicar los estilos y patrones predominantes en la arquitectura del sistema.

Se ha convenido en utilizar algunos estereotipos específicos para el diseño, particularmente a la hora de modelar para Java Script y Symfony como framework a fin de facilitar visualmente la comprensión de lo modelado.

Vista de Casos de Uso

El sistema, de acuerdo a la captura de requisitos va a estar formado por subsistemas que agrupan casos de usos, a estos subsistemas le llamaremos módulos y el criterio fundamental que se ha considerado para esta agrupación es la estrecha vinculación que existe entre los casos de usos.

Se pueden identificar los siguientes módulos:

- **Módulo de Registros y Clasificadores (MRC):** destinado a confeccionar y mantener un registro de todos los CI, mantener las clasificaciones de los mismos según la metodología de la ONE para dicha actividad.

- **Módulo Generador de Modelos y Encuestas (MGM):** destinado a la confección de los diversos y variados facsímiles o formularios de modelos del SEN, abarcará las 2 grandes variantes de modelos del SEN, los del Sistema de Estadística Continua (SEC) y los del Sistema de Encuestas Periódicas (SEP).

- **Módulo de Entrada de Datos (MED):** destinado a consumir el producto de MGM y presentar una interfaz de usuario según el modelo diseñado, permitiendo la entrada de datos en el formato

adecuado y realizar un conjunto de chequeos o validaciones sobre los datos para garantizar la corrección de los mismos.

- **Sistema de Gestión de Reportes Dinámicos(SGRD):** destinado a la generación de todo tipo de reportes a tomando como fuente de datos “cualquier base de datos”

Casos de uso arquitectónicamente significativos

Se han identificado los siguientes casos de usos como arquitectónicamente significativos atendiendo a la cantidad de clases que definen, a la complejidad de su implementación y al impacto que podrían tener no solo en la arquitectura sino en el desarrollo del sistema en general:

Desarrollar Encuesta: define las principales clases que representan los componentes de una encuesta, así como los componentes visuales que permiten el diseño de la encuesta.

Desarrollar Modelo Estadístico: define las principales clases de un modelo estadístico.

Digitar Modelo: define los componentes que se encargan de recuperar un modelo y de construir y presentar dinámicamente este como un formulario.

Transferir Información: define la interfaz para la recuperación de los datos estadísticos directamente de los módulos del ERP.

Descripción de los Casos de Usos

Módulo de Registros y Clasificadores

Todos los casos de uso de este módulo (Fig. 6) cumplen el patrón de requisitos CRUD (Create, Read, Update, Delete) en casos particulares es necesario implementar búsquedas filtradas y las funcionalidades requeridas para vincular la información.

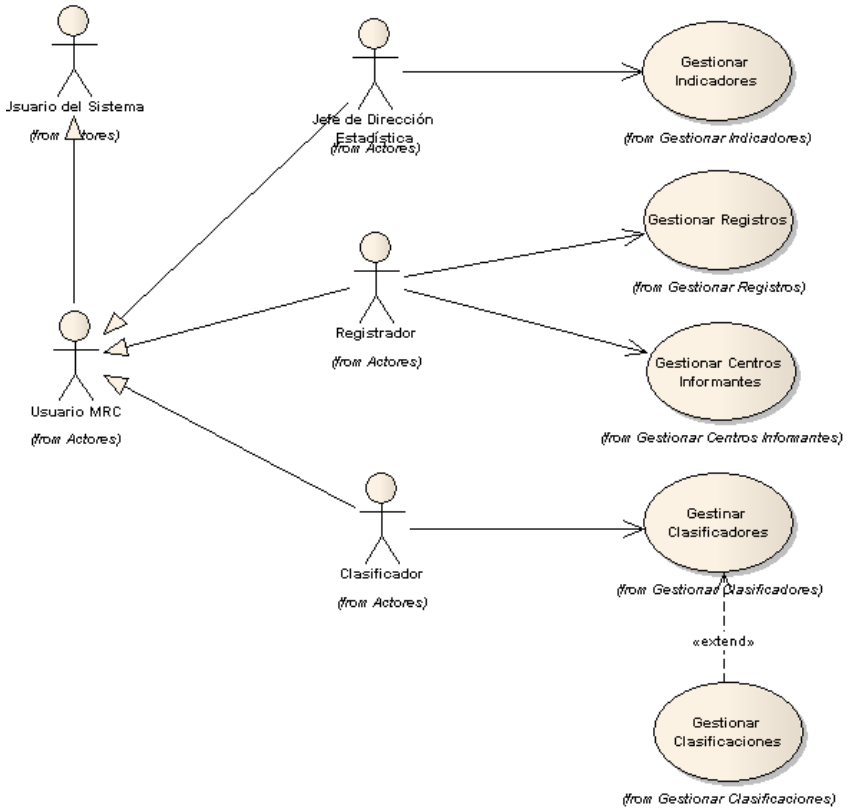


Fig. 6 Casos de Uso del Módulo de Registros y Clasificadores

CU. Gestionar Indicadores

Este caso de uso es un CRUD completo para la entidad Indicador (véase Vista de Datos), además implementa una búsqueda filtrada de los indicadores a través de sus campos más representativos.

CU. Gestionar Variantes

Este caso de uso es un CRUD completo para la entidad Variante (véase Vista de Datos)

CU. Gestionar Centros Informantes

Este caso de uso es un CRUD completo para la entidad Centro Informante (véase Vista de Datos) además permite la gestión de la asociación de las Clasificaciones correspondientes a los Centros Informantes.

CU. Gestionar Registros

Este caso de uso es un CRUD completo para la entidad Registro (véase Vista de Datos) además permite la gestión de los Clasificadores propios del Registro.

CU. Gestionar Clasificadores

Este caso de uso es un CRUD completo para la entidad Clasificador (véase Vista de Datos) además permite la gestión de las Clasificaciones que puede tener el clasificador.

CU. Gestionar Clasificaciones

Este caso de uso extiende al cu. Gestionar Clasificadores, es un CRUD completo para la entidad Clasificación.

Módulo Generador de Modelos y Encuestas

Este módulo (Fig. 7) particularmente aporta 2 de los casos de uso arquitectónicamente significativos no solo por definir la mayor cantidad de clases del sistema, sino también por las complejidades que su diseño entraña.

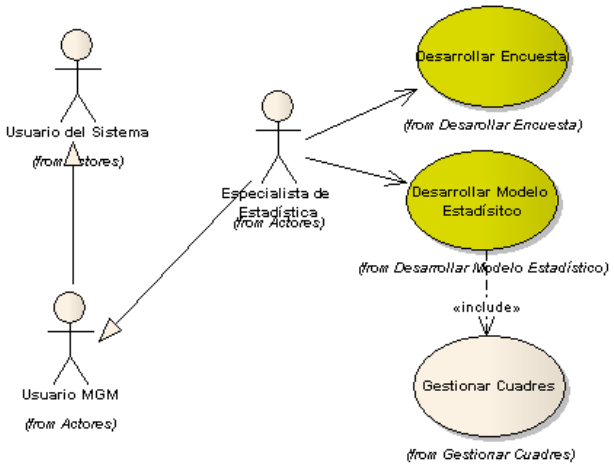


Fig. 7 Casos de Uso del Módulo Generador de Modelos y Encuestas

CU. Desarrollar Encuesta

Este caso de uso es el encargado de desarrollar los formularios visuales para la captación de la información del SIET que comprende a la encuesta periódica.

CU. Desarrollar Modelo Estadístico

Este caso de uso permite definir y desarrollar los formularios visuales para la captación de la información del SIET que comprende la estadística continua. Como parte de la definición se considera el enlace respectivo con la(s) fuente(s) de datos del ERP, así como las transformaciones que la información requiera para adecuarla al concepto estadístico.

CU. Gestionar Cuadres de Validación

Este caso de uso es responsable de la creación de las validaciones incorporadas para la captación de la información del SIET que comprende la estadística continua.

Módulo de Entrada de Datos

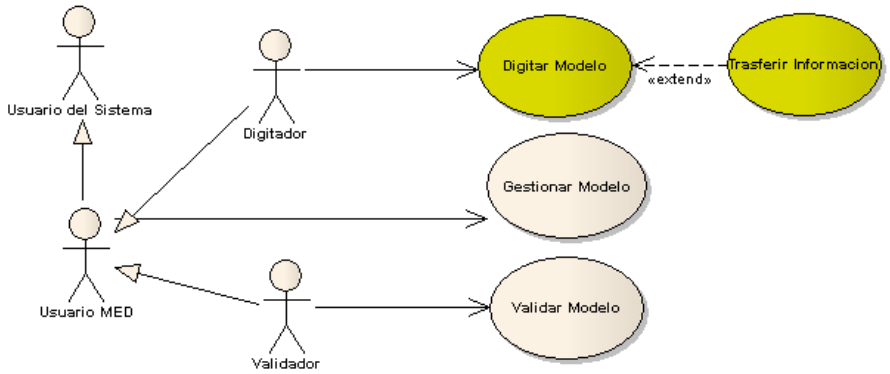


Fig. 8 Casos de Uso del Módulo de Entrada de Datos

CU. Gestionar Modelos

Este caso de uso comprende la administración de modelos tanto de estadísticas continuas como encuestas periódicas a lo largo de todo el proceso de trabajo, que comprende elaboración, validación y almacenamiento. Además incorpora las funcionalidades de importación y exportación de la información.

CU. Digitar Modelo

Este caso de uso es responsable de la entrada de datos estadísticos tanto por una vía rápida como a través del formulario visual del modelo, además presenta la ayuda metodológica asociada al modelo en digitación.

CU. Validar Modelo

Este caso de uso permite la validación de los datos estadísticos en función de los cuadres de validación agregados al modelo cuando este fue diseñado.

CU. Transferir Información

Este caso de uso es el encargado de la recuperación de información directamente de las fuentes de datos del ERP.

Vista Lógica

Estilos arquitectónicos

Atendiendo a los objetivos y restricciones vistos anteriormente se optan por los siguientes estilos que tipifican la arquitectura del sistema:

- Cliente Servidor[6]: se caracteriza por existir un nodo (o más) servidor donde reside el servicio que se expone y varios clientes que consumen dicho servicio, en el sistema (Fig. 9) este estilo responde al hecho de que se trate de una aplicación web y se concreta con un servidor de bases de datos (PostgreSQL) un servidor web (con función de servidor de aplicaciones, Apache 2) y varios clientes que acceden al sistema a través de un navegador.

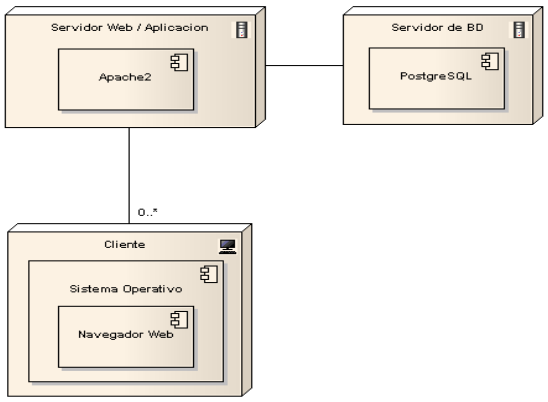


Fig. 9 Estilo Cliente - Servidor

- Multicapas: se caracteriza por organizar los componentes del sistema en capas con responsabilidades bien definidas y donde las capas del nivel más alto invocan los servicios de las del nivel inferior, en el sistema (Fig. 10) es posible identificar las siguientes capas: presentación, negocio, acceso a datos, datos e infraestructura.

[6] Cliente servidor es una especialización del estilo en capas, se presenta como un estilo más por una cuestión de entendimiento y de que resulta coherente con algunas posiciones arquitectónicas presentes en la bibliografía respectiva, es necesario destacar que no está entre los estilos presentados en el capítulo 1 del trabajo.

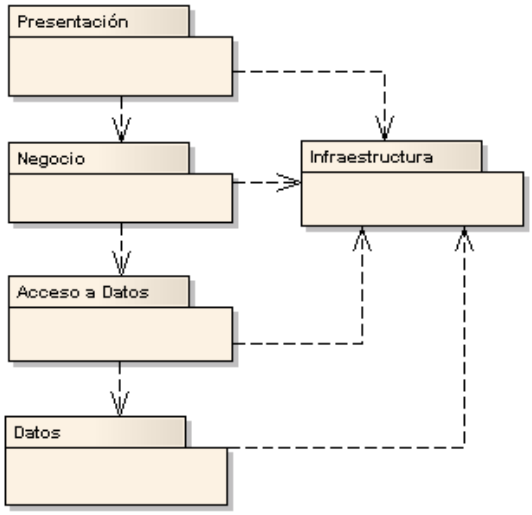


Fig. 10 Estilo Multicapas

- Máquina Virtual: es utilizado para garantizar la portabilidad de la aplicación en diferentes plataformas, consiste en el hecho que los componentes de la aplicación se ejecutan sobre un componente que abstrae la máquina real, se concreta por los intérpretes Zend Engine II de PHP y Gecko 1.8 de Java Script en Mozilla Firefox con la función de máquina virtual.

- Modelo – Vista – Controlador: separa conceptualmente la representación visual de la aplicación, las acciones que intercambian datos y el modelo de negocio y su dominio. En el sistema (Fig. 11) se concreta con la identificación de 3 elementos diferentes: la vista implementada en Java Script reside del lado del cliente en tiempo de ejecución, el Controlador, y el Modelo que junto al controlador reside del lado del servidor, la interacción entre la vista y el controlador se realiza a través de una solicitud AJAX y la respuesta dada por el controlador puede encontrarse en JSON o XML según corresponda la solicitud.

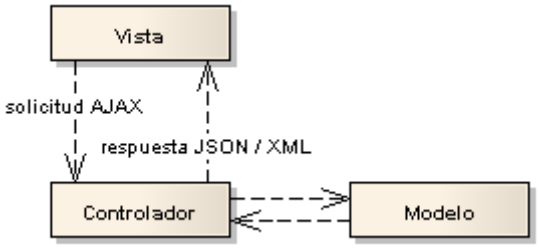


Fig. 11 Estilo Modelo - Vista - Controlador

- Basado en Componentes: es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explicitas. En el sistema (Fig. 12) se concreta con la implementación de los componentes de negocio siguiendo las especificaciones oportunas del framework Symfony, generando los componentes de acceso a datos directamente desde el esquema de la base de datos con el generador de Propel y con la implementación de componentes de presentación extendiendo los componentes ya elaborados en Ext JS.

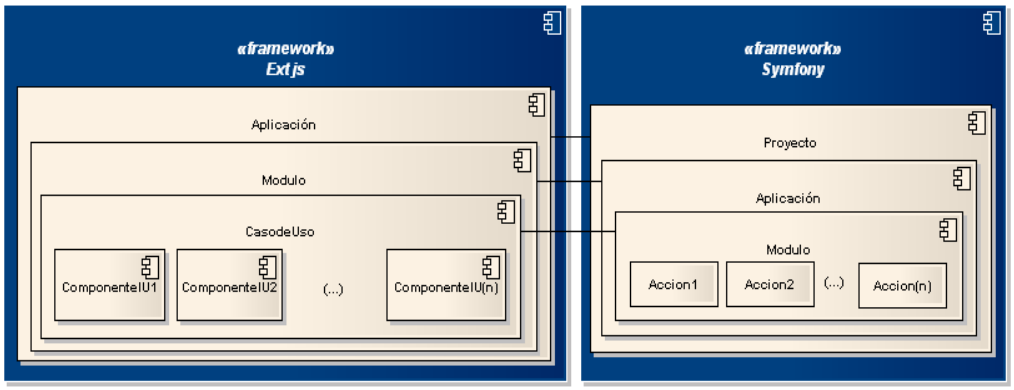


Fig. 12 Estilo Basado en Componentes

Elementos del modelo arquitectónicamente significativos

Estructura organizativa del sistema

La organización física de los elementos que conforman el sistema se realizará teniendo en cuenta los principios establecidos por Symfony que estructuran el proyecto en aplicaciones y módulos (Fig. 13). Symfony considera un proyecto como "un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos". Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones, una aplicación se ejecuta de forma independiente respecto de otras aplicaciones del mismo proyecto. Cada aplicación está formada por uno o más módulos. Los módulos almacenan las acciones, que representan cada una de las operaciones que se puede realizar en un módulo.

Teniendo en cuenta estas aseveraciones es posible homologar los conceptos que se manejan a nivel de ingeniería con las definiciones de estructura que realiza Symfony, de esta forma cuando se habla del sistema en todo su conjunto se corresponde al proyecto en Symfony, cuando se hace referencia a un módulo en ingeniería se corresponde homólogamente con una aplicación, y cuando se trate de un caso de uso concretamente corresponde a un módulo en Symfony, establecer esta correspondencia es útil a la hora implementar la estructura del proyecto.

Fig. 13 Organización propuesta por Symfony

Tabla 2 Estructura de directorios de un proyecto Symfony

Directorio	Descripción
apps	Contiene un directorio por cada aplicación del proyecto.
cache	Contiene la versión cacheada de la configuración y (si está activada) la versión cacheada de las acciones y plantillas del proyecto. El mecanismo de cache utiliza los archivos de este directorio para acelerar la respuesta a las peticiones web. Cada aplicación contiene un subdirectorio que guarda todos los archivos PHP y HTML pre procesado.
config	Almacena la configuración general del proyecto.
data	En este directorio se almacenan los archivos relacionados con los datos, como por ejemplo el esquema de una base de datos, el archivo que contiene las instrucciones SQL para crear las tablas e incluso un archivo de bases de datos de SQL.
doc	Contiene la documentación del proyecto, documentación generada por PHPDoc.
lib	Almacena las clases y librerías externas. Se suele guardar todo el código común a todas las aplicaciones del proyecto. El subdirectorio model/ guarda el modelo de objetos del proyecto.
log	Guarda todos los archivos de log generados por Symfony. También se puede utilizar para guardar los logs del servidor web, de la base de datos o de cualquier otro componente del proyecto. Symfony crea un archivo de log por cada aplicación y por cada entorno.

plugins	Almacena los plugins instalados en la aplicación.
test	Contiene las pruebas unitarias y funcionales escritas en PHP y compatibles con el framework de pruebas de Symfony.
web	La raíz del servidor web. Los únicos archivos accesibles desde Internet son los que se encuentran en este directorio. Contiene a las carpetas “js” para los archivos fuentes de Java Script, “css” para las hojas de estilos, e “images” para las imágenes.

Tabla 3 Estructura de directorios de una aplicación Symfony

Directorio	Descripción
config	Contiene los diferentes archivos de configuración creados con YAML. Aquí se almacena la mayor parte de la configuración de la aplicación, salvo los parámetros propios del framework. También es posible redefinir en este directorio los parámetros por defecto si es necesario.
i18n	Contiene todos los archivos utilizados para la internacionalización de la aplicación, sobre todo los archivos que traducen la interfaz. La internacionalización también se puede realizar con una base de datos, en cuyo caso este directorio no se utilizaría.
lib	Contiene las clases y librerías utilizadas exclusivamente por la aplicación
modules	Almacena los módulos que definen las características de la aplicación
templates	Contiene las plantillas globales de la aplicación, es decir, las que utilizan todos los módulos. Por defecto contiene un archivo llamado layout.php, que es el layout principal con el que se muestran las plantillas de los módulos.

Tabla 4 Estructura de directorios de un módulo Symfony

Directorio	Descripción
actions	Normalmente contiene un único archivo llamado actions.class.php y que corresponde a la clase que almacena todas las acciones del módulo. También es posible crear un archivo diferente para cada acción del módulo
config	Puede contener archivos de configuración adicionales con parámetros exclusivos del módulo
lib	Almacena las clases y librerías utilizadas exclusivamente por el módulo
templates	Contiene las plantillas correspondientes a las acciones del módulo. Cuando se crea un nuevo módulo, automáticamente se crea la plantilla llamada indexSuccess.php

Es importante destacar que independientemente de la riqueza de Symfony para la elaboración de la presentación estas posibilidades no se utilizan dado que la presentación se implementa utilizando Ext JS y AJAX por tal razón se hace necesario definir una organización (Fig. 14) para los componentes de presentación que oportunamente es equivalente a la definida para un proyecto Symfony.

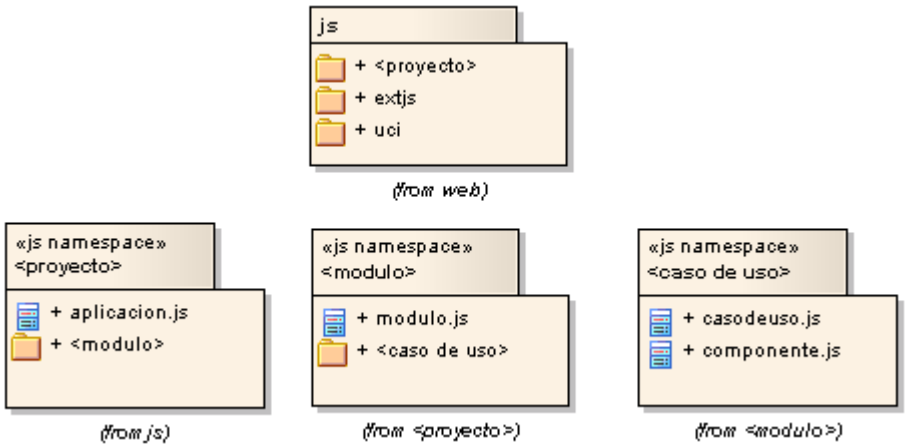


Fig. 14 Organización para los Componentes en Java Script

Tabla 5 Estructura de directorios de una aplicación Ext JS

Directorio	Descripción
<modulo>	Se incluye una carpeta para cada caso de uso así como un alias para nombrar cada una de estas carpetas.
<aplicacion>.js	Este archivo JavaScript contiene la definición de una clase Modulo para el sistema concretamente, esta clase hereda de uci.webApp.Module que entre sus responsabilidades incluye la de registrar clases de casos de usos.

Tabla 6 Estructura de directorios de un módulo Ext JS

Directorio	Descripción
<casodeuso>	Se incluirá una carpeta para cada módulo. Se escogerá el alias del modulo para nombrar cada una de estas carpetas.
<modulo>.js	Este archivo JavaScript contiene la definición de la clase Aplicación para el sistema concretamente, esta clase hereda de uci.webApp.Applicaction que entre sus responsabilidades incluye la de registrar instancias de módulos y la de ejecutar la aplicación.

Tabla 7 Estructura de directorios de un caso de uso Ext JS

Directorio	Descripción
<casodeuso>.js	Este archivo JavaScript contiene la definición de la clase Caso de Uso para un módulo del sistema concretamente, esta clase hereda de uci.webApp.UseCase que entre sus responsabilidades incluye la de iniciar la secuencia de interacción entre el usuario y el sistema.

<componente>.js	Este archivo JavaScript contiene la definición de un componente que se utiliza en el caso de uso específicamente.

Elementos genéricos que componen la realización de un Caso de Uso

La modelación de la realización de los casos de uso considera la elaboración de los diferentes elementos que lo conforman (Fig. 15), a saber de la disciplina de análisis, los componentes de interface de usuario (fronteras o boundary), los componentes de negocio con los servicios y la lógica del negocio (estereotipo control), y las entidades, componentes que posteriormente serán refinados en etapa de diseño. A modo general se presentan las pautas de modelación que se han de seguir para concretar esta realización.

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

Del lado del cliente los principales elementos corresponden a las librerías “ext-base.js”, “ext-all.js”, “uci.webApp”, a los componentes específicos del negocio como paneles de trabajos, componentes propios del negocio y servicios asociados al caso de uso, y a los componentes genéricos fuertemente reutilizables. Se resalta de manera particular aquellos componentes que tengan un carácter genérico en los cuales se realizará un énfasis especial en su diseño, dado que estos componentes formarán posteriormente parte de la base tecnológica especializada que dispondrá el centro.

Del lado del servidor se encuentran las acciones (vistas con el estereotipo “symfony action”) agrupadas en el correspondiente módulo (visto con el estereotipo “symfony module”) al que pertenecen, se destaca la relación que existe de flujo de información entre el componente caso de uso del lado del cliente y el componente módulo del lado del servidor. Se puede observar que las solicitudes de AJAX se encuentran implementadas de manera concentrada en el Caso de Uso que representa nuestro enlace con las acciones que del lado del servidor respaldan estas solicitudes, devolviendo las respuestas específicamente en formato JSON al caso de uso.

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

Las relaciones fundamentales entre estos elementos son las de importación, las de dependencia, y las de flujo de datos. Cuando se presenta una importación entre componentes de Java Script se está utilizando un mecanismo de carga dinámica de la librerías de esta forma se mejora el rendimiento asociado al tiempo de carga inicial de la aplicación al cargar solamente lo estrictamente necesario en cada momento, el otro caso es la importación que de manera estática se garantiza por la vía tradicional a través de las correspondientes etiquetas HTML. La dependencia entre el panel de trabajo y el caso de uso es necesaria dado que es el panel de trabajo quien estructura y enlaza a los diferentes componentes con los servicios de esta forma se garantiza el desacoplamiento entre los componentes y la lógica del negocio particular. La relación de flujo de datos entre el caso de uso y el módulo representa la correspondencia de las invocaciones AJAX agrupadas en el caso de uso con las acciones que del lado del servidor darán respuesta a las mismas, se transfiere tanto en la solicitud como en la respuesta el objeto con todos los atributos que se envían a la acción en el primer caso y que se responden por la acción en el segundo caso.

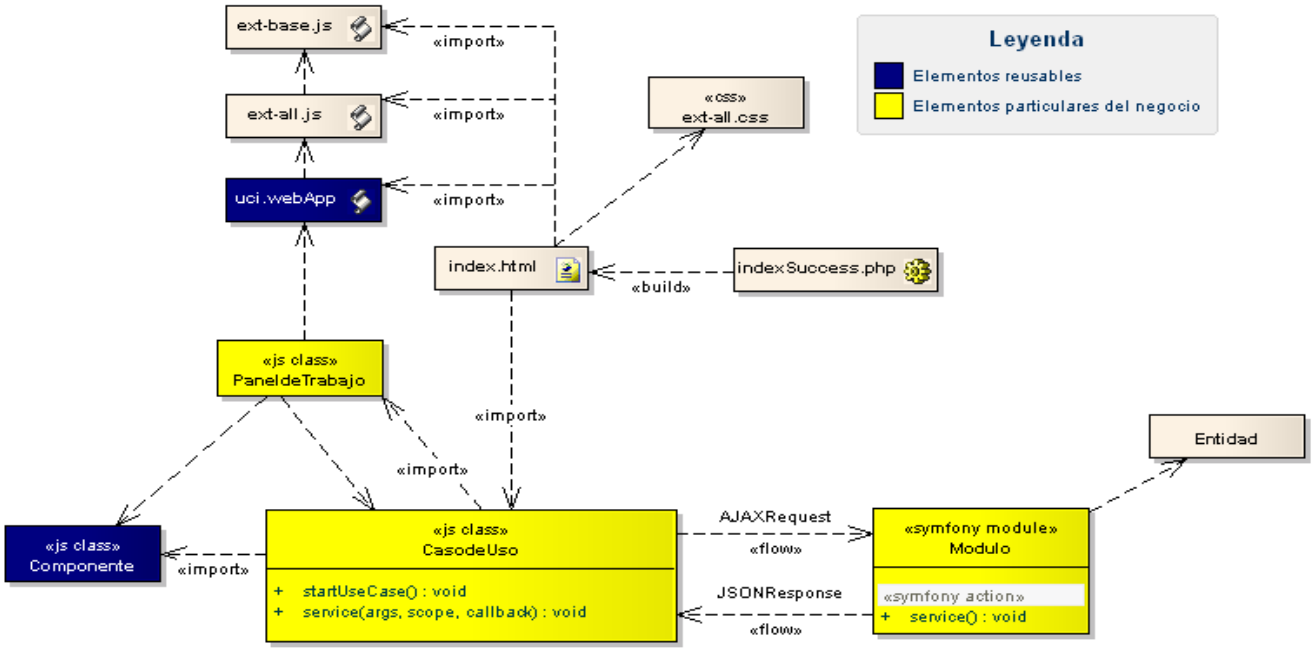


Fig. 15 Elementos Genéricos que Conforman la Realización de un Caso de Uso

Realización de los Casos de Uso (elementos arquitectónicamente significativos)

Desarrollar Encuesta

Define las principales clases que representan los componentes de una encuesta, así como los componentes visuales que permiten el diseño de la encuesta (Fig. 16). Da respuesta al requisito referido a la elaboración de los formularios para la captación de los datos estadísticos como encuestas. Podría por demás considerarse un componente genérico sobre el cual se debe hacer un énfasis especial en su diseño, dado que podría reutilizarse en un sin número de aplicaciones.

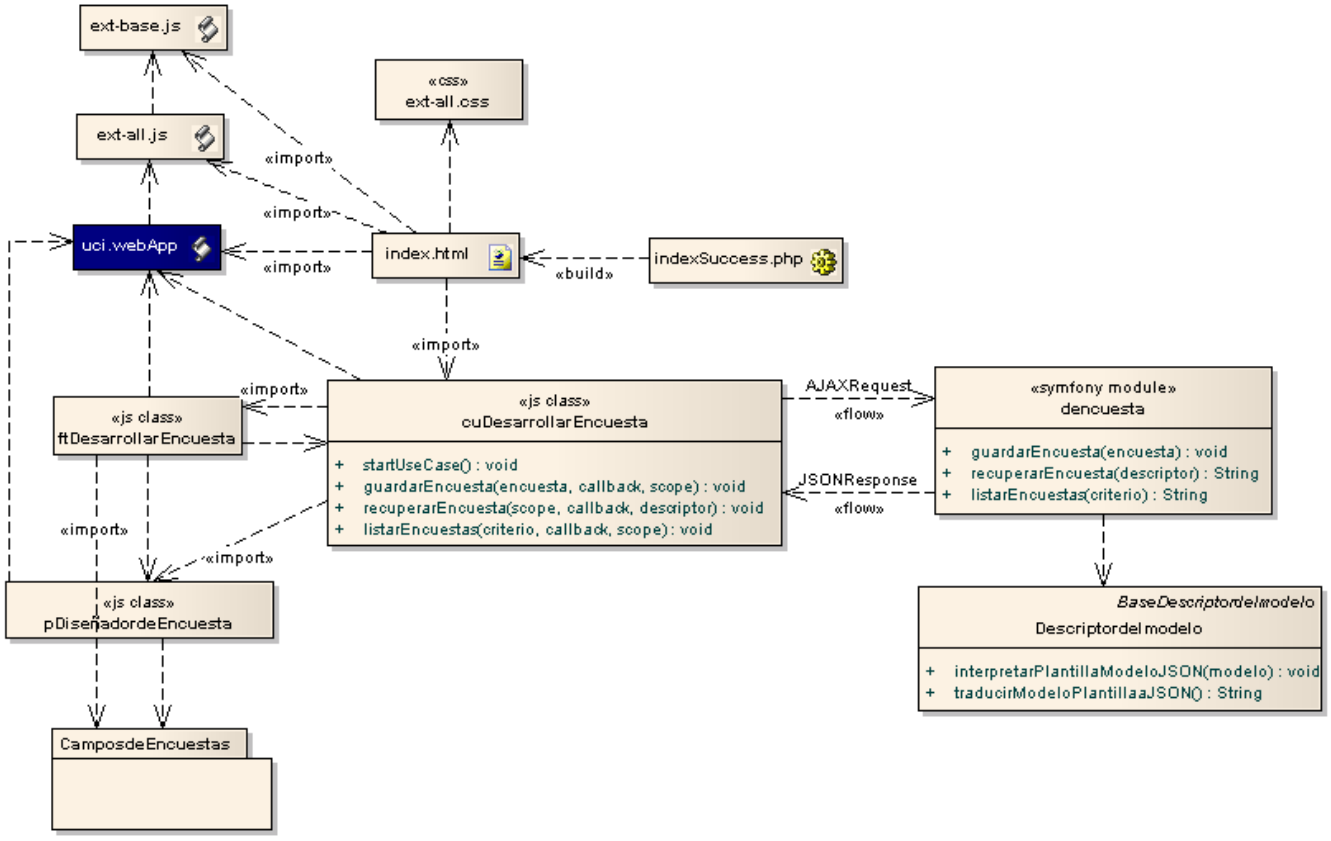


Fig. 16 Clases de la Realización del Diseño (CU Desarrollar Encuesta)

Las principales clases son:

ftDesarrollarEncuesta: es el componente que se mostrará en el área de trabajo del caso de uso. Incluye el menú con las diferentes acciones que se realizan con la encuesta y el panel diseñador de encuestas.

pDiseñadodeEncuesta: es el componente donde se diseña una encuesta. Incluye a los componentes requeridos para manejar el diseño de la encuesta a saber: **PaneldePropiedades**, **PaletadeHerramientas**, **Navegador**, y **PaneldeDibujo** todos pre-elaborados con antelación y agrupados en el marco “uci.webApp” dado el carácter genérico de estos.

cuDesarrollarEncuesta: es el componente donde se encuentra la implementación de las solicitudes AJAX. Se distinguen los servicios claves a implementar:

- *guardarEncuesta*: envía como JSON la definición de la encuesta elaborada (téngase en cuenta que se trata de un tipo específico de descriptor de modelo, para mas detalles consúltese el modelo de datos)

- recuperarEncuesta: dado el descriptor con los atributos que unívocamente definen una determinada encuesta recupera su definición como JSON y la devuelve como primer parámetro a través de la invocación de la función de retorno “callback” .

- *listarEncuesta:* dado el conjunto de criterios se retornan los descriptores con los atributos que definen unívocamente una encuesta, se devuelve como primer parámetro de la función de retorno.

- **<<symfony module>> dencuesta:** representa al módulo asociado al caso de uso, y contiene las acciones que dan respuesta a las solicitudes AJAX, a saber: *guardarEncuesta*¸ *recuperarEncuesta, listarEncuesta,* en todos los casos el retorno se realiza como texto (en formato JSON).

- **CamposdeEncuestas:** se muestran como paquete, pero refiere en realidad al conjunto de componentes (controles de usuarios) que se utilizarán en la elaboración de una encuesta.

- **Descriptordelmodelo**: esta clase se corresponde con la entidad que se persiste, téngase en cuenta que una encuesta es un subtipo de descriptor de modelo (véase el modelo de datos).

Desarrollar Modelo Estadístico

Define las principales clases de un modelo estadístico así como una gran cantidad de componentes que necesitan implementarse para cubrir las necesidades planteadas en los requisitos (Fig. 17). Da respuesta al requisito de elaborar modelos de estadística continua.

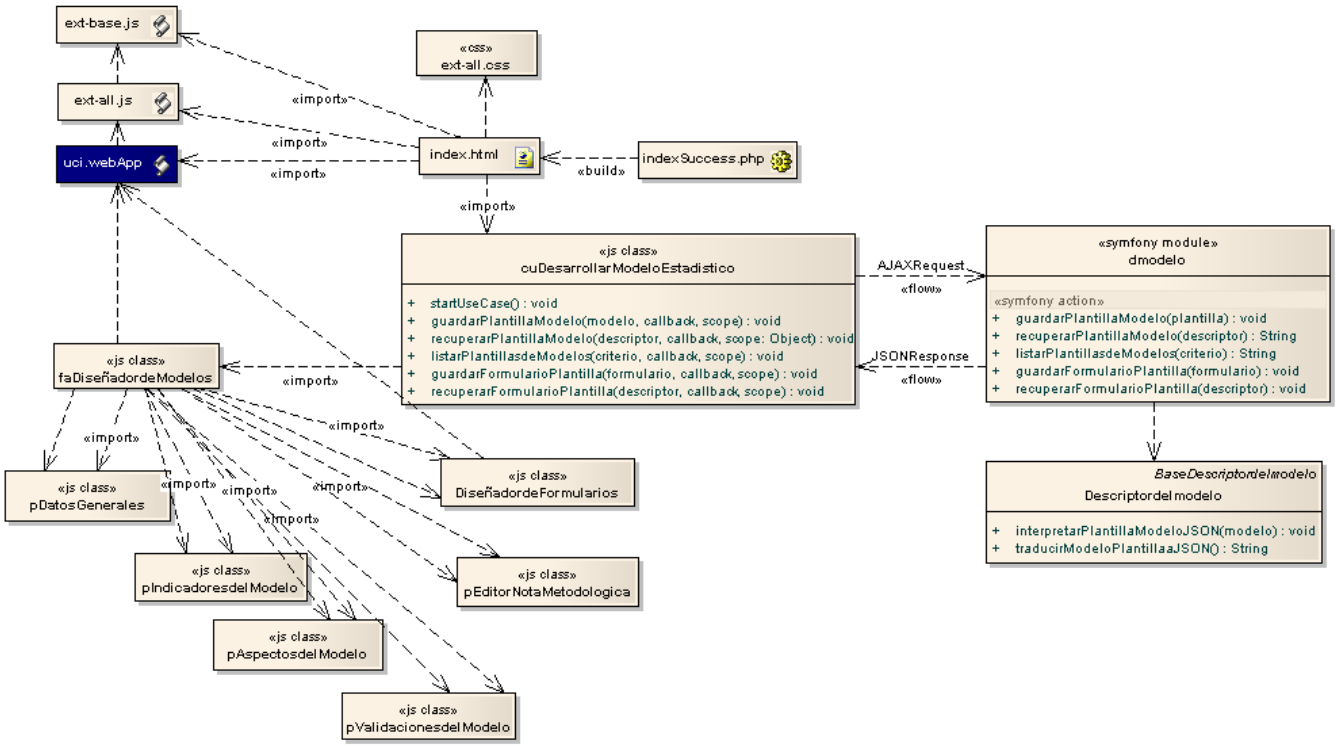


Fig. 17 Clases de la Realización del Diseño (CU Desarrollar Modelo Estadístico)

Las principales clases son:

- **faDiseñadordeModelos:** es el componente que se mostrará en el área de trabajo del caso de uso. Se trata de un panel de asistentes (wizard) que facilitará al usuario la realización del modelo a través de los pasos lógicos que este comprende.

- **pDatosGenerales:** este panel del asistente captura los datos generales que define al modelo de estadística continua.

- **pIndicadoresdelModelo:** este panel del asistente captura los indicadores (filas) asociados al modelo de estadística continua.

- **pAspectosdelModelo:** este panel del asistente captura los aspectos (columnas) asociados al modelo de estadística continua.

- **pValidacionesdelModelo:** este panel del asistente captura las validaciones asociados al modelo de estadística continua.

- **pEditorNotaMetodologica:** este panel del asistente se se utiliza para elaborar la descripción asociada a la nota metodológica del modelo de estadística continua.

- **pDiseñadordeFormularios:** este componente se utiliza para diseñar la vista de formulario de los modelos de estadística continua.

- **cuDesarrollarModeloEstadistico:** es el componente donde se encuentra la implementación de las solicitudes AJAX. Se distinguen los servicios claves a implementar:

  - *guardarPlantillaModelo*: envía como JSON la definición de la plantilla del modelo elaborada .

  - recuperar*PlantillaModelo*: dado el descriptor con los atributos que unívocamente definen una determinada plantilla de modelo recupera su definición como JSON y la devuelve como primer parámetro a través de la invocación de la función de retorno “callback”.

  - *listarPlantillasdeModelos:* dado el conjunto de criterios se retornan los descriptores con los atributos que definen unívocamente una encuesta, se devuelve como primer parámetro de la función de retorno.

  - *guardarFormularioPlantilla*: envía como JSON la definición del formulario plantilla del modelo elaborado.

  - *recuperarFormularioPlantilla:* dado el descriptor con los atributos que unívocamente definen determinado formulario plantilla recupera su definición como JSON y la devuelve como primer parámetro a través de la invocación de la función de retorno “callback”.

- **<<symfony module>> dmodelo:** representa al módulo asociado al caso de uso, y contiene las acciones que dan respuesta a las solicitudes AJAX, a saber: *guardarPlantillaModelo*¸ recuperar*PlantillaModelo, listarPlantillasdeModelos, guardarFormularioPlantilla, recuperarFormularioPlantilla,* en todos los casos el retorno se realiza como texto (en formato JSON).

- **Descriptordelmodelo**: esta clase se corresponde con la entidad que se persiste.

Digitar Modelo

Define los componentes que se encargan de recuperar un modelo y de construir y presentar dinámicamente este como un formulario (Fig. 18). Da respuesta a los requisitos de entrada, validación y almacenamiento de la información estadística.

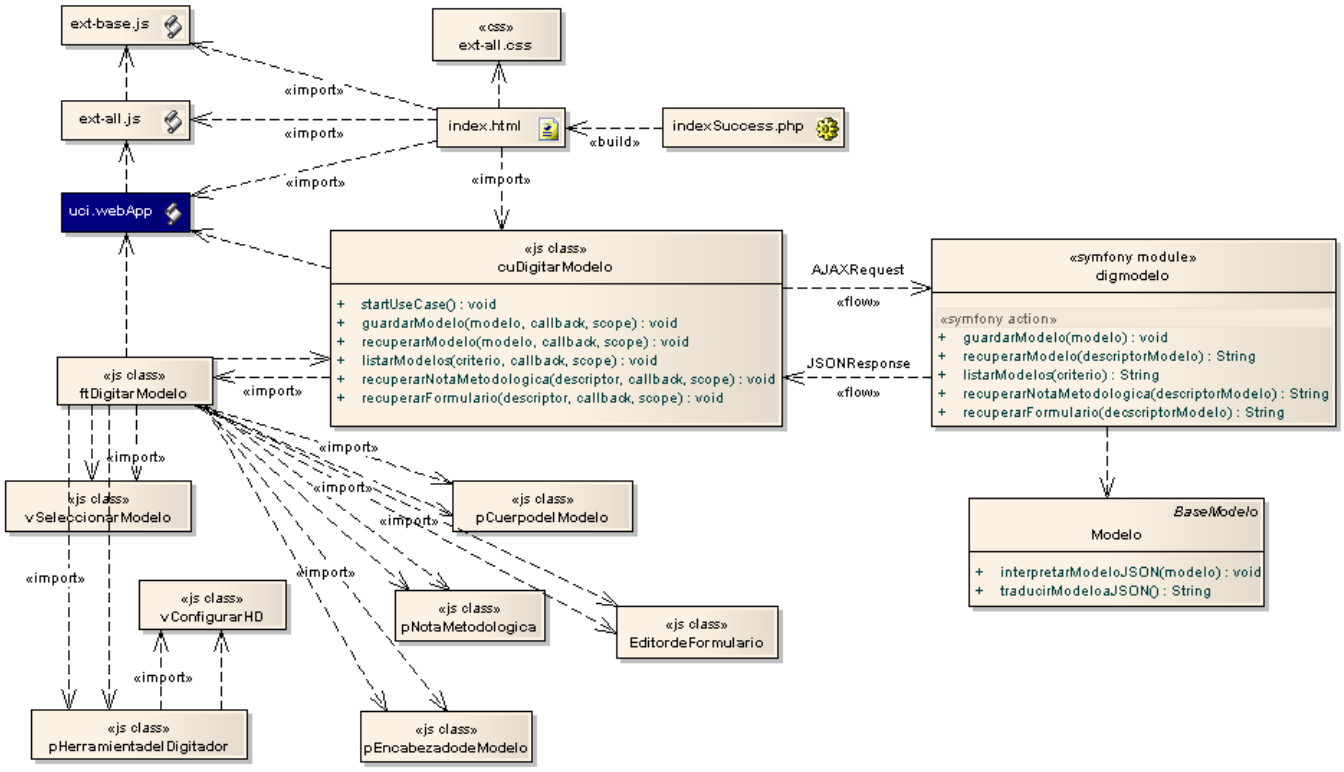


Fig. 18 Clases de la Realización del Diseño (CU Digitar Modelo)

Las principales clases son:

- **faDigitarModelo:** es el componente que muestra el área de trabajo del caso de uso. Se trata de un panel que facilitará al usuario la digitación del modelo y posterior validación. Posee 3 secciones cada una de la cual corresponde a una pestaña de trabajo, a saber: vista de digitación, vista de formulario y nota metodológica.

- **vSeleccionarModelo:** esta ventana permite explorar el universo de modelo estadísticos y de plantillas para seleccionar sobre el que se va a trabajar.

- **pEncabezadodeModelo:** este panel captura los datos primarios relativos al modelo de estadística continua.

- **pCuerpodelModelo:** este panel contiene la tabla con los datos del modelo de estadística continua.

- **pHerramientadelDigitador:** este panel permite la entrada rápida (digitación) del modelo de estadística continua.

- **vConfiguracionHD:** esta ventana muestra diferente opciones de configuración de la herramienta del digitador.

- **EditordeFormularios:** este componente se utiliza para introducir los datos en la vista de formulario de los modelos de estadística continua o de encuesta.

- **cuDigitarModelo:** es el componente donde se encuentra la implementación de las solicitudes AJAX. Se distinguen los servicios claves a implementar:

  - *guardarModelo*: envía como JSON los datos de un modelo.

- recuperar*Modelo*: dado el descriptor con los atributos que unívocamente definen un determinado modelo recupera su contenido como JSON y lo devuelve como primer parámetro a través de la invocación de la función de retorno “callback”.

- *listarModelos:* dado el conjunto de criterios se retornan los descriptores con los atributos que definen unívocamente los modelos que corresponden con dicho criterio, se devuelve como primer parámetro de la función de retorno.

- *recuperarNotaMetodologica*: dado el descriptor con los atributos que unívocamente definen un determinado modelo recupera su nota metodológica asociada y lo devuelve como primer parámetro a través de la invocación de la función de retorno “callback”.

- *recuperarFormularioPlantilla:* dado el descriptor con los atributos que unívocamente definen determinado formulario plantilla recupera su definición como JSON y la devuelve como primer parámetro a través de la invocación de la función de retorno “callback”.

- **<<symfony module>> digmodelo:** representa al módulo asociado al caso de uso, y contiene las acciones que dan respuesta a las solicitudes AJAX, a saber: *guardarModelo,* recuperar*Modelo, listarModelos, recuperarNotaMetodologica* y *recuperarFormularioPlantilla,* en todos los casos el retorno se realiza como texto (en formato JSON).

- **Modelo**: esta clase se corresponde con la entidad que se persiste

Transferir Información

Define la interfaz para la recuperación de los datos estadísticos directamente de los módulos del ERP (Fig. 19).

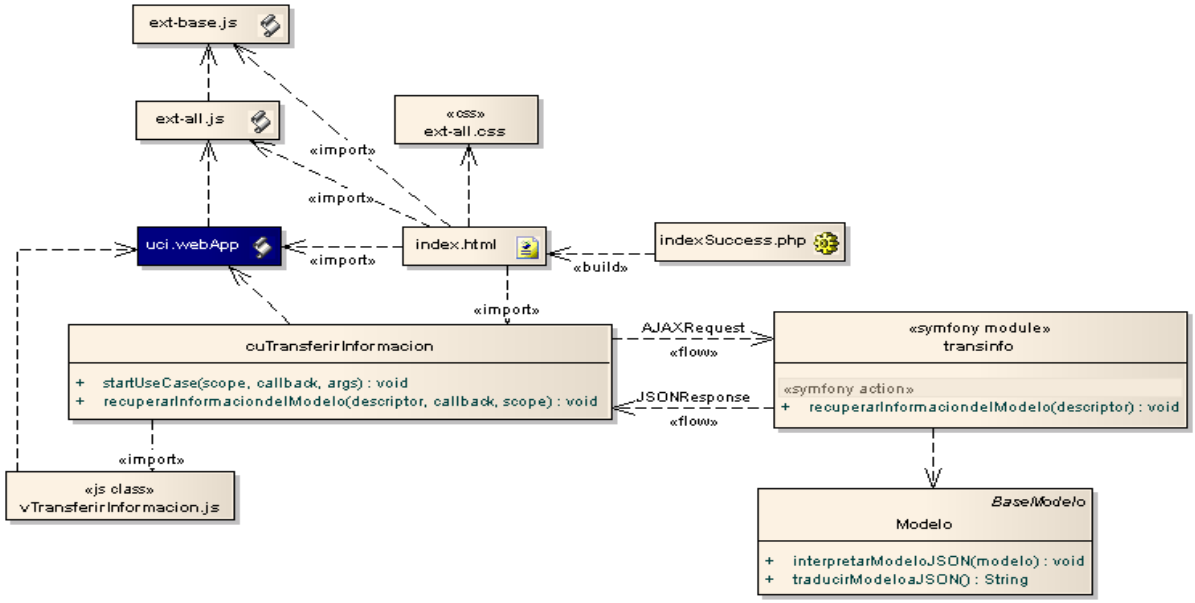


Fig. 19 Clases de la Realización del Diseño (CU Transferir Información)

Las principales clases son:

- **vTransferirInformacion:** esta ventana permite realizar la recuperación de la información directamente desde los registros en la base de datos del ERP.

- **cuTransferirInformacion:** es el componente donde se encuentra la implementación de las solicitudes AJAX. Se distinguen los servicios claves a implementar:

  - *recuperarInformaciondelModelo:* dado el descriptor con los atributos que unívocamente definen un determinado modelo recupera su información asociada desde los registros en la base de datos del ERP lo devuelve como primer parámetro a través de la invocación de la función de retorno “callback”.

- **<<symfony module>> transinfo:** representa al módulo asociado al caso de uso, y contiene la acción que da respuesta a la solicitud AJAX, a saber: *recuperarInformaciondelModelo.* El retorno se realiza como texto (en formato JSON).

Vista de Procesos

No existen procesos independientes de los servicios de infraestructura que prestan los servidores de base de datos y web, por tal razón no procede una vista de procesos de la aplicación.

Vista de Implementación

Se ha organizado la vista de implementación para presentar los diferentes componentes asociados a cada una de las capas que conforman el sistema, se distingue la capa de presentación, la capa de negocio, la de acceso a datos, la de datos e infraestructura.

Capa de Presentación

La capa de presentación (Fig. 20) está compuesta por las librerias Ext Js, OpenJacob y uci.webApp, el componente de presentación de SIGE y el de SGRD. Los componetes de software que se encuentran en el nivel inferior corresponden al explorador Mozilla Firefox y el sistema operativo Nova.

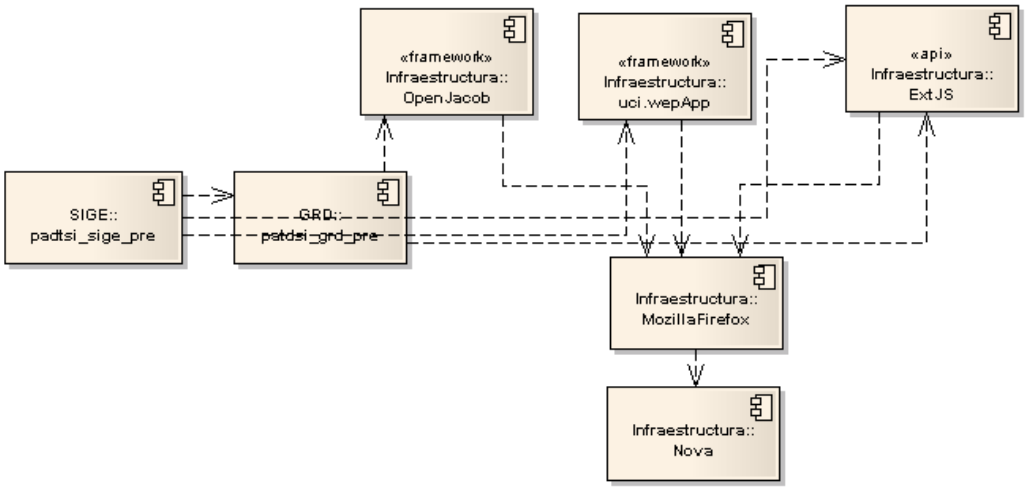


Fig. 20 Componentes de la capa de presentación

Capa de Negocio

La capa de negocio (Fig. 21) está compuesta por las librerías Symfony, pChart, PHPReport, el componente de negocio de SIGE, el de SGRD (ambos sintetizan todos los componentes que se

corresponden a las aplicaciones, los módulos y las acciones definidas en la estructura de Symfony). Los componentes de software sobre los que estos se asientan son el intérprete de PHP integrado en el servidor Apache con la extensión php5_xsl para trabajo con XSL, el sistema operativo propuesto para el servidor es Debian.

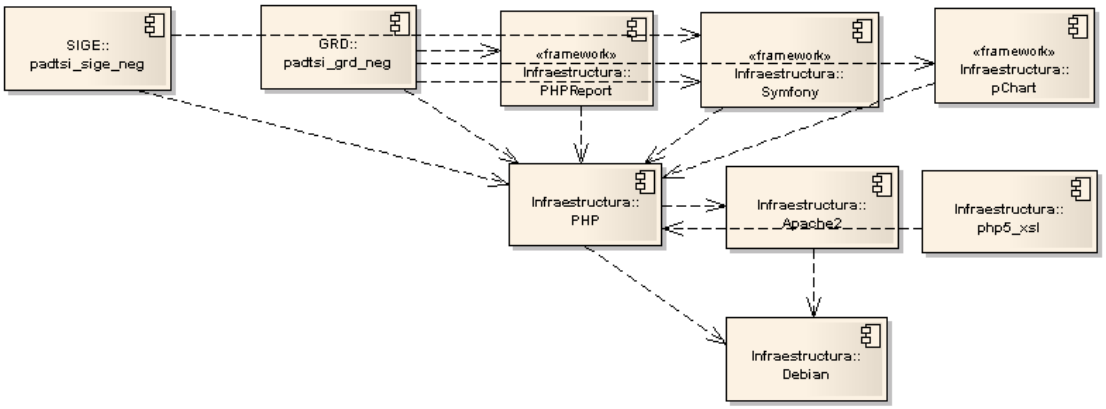


Fig. 21 Componentes de la capa de negocio

Capa de Acceso a Datos

La capa de acceso a datos (Fig. 22) está compuesta por los componentes comunes del framework de acceso a datos Propel, por la capa de abstracción Creole, y por las librerías resultado del mapeo y generadas por Propel de las respectivas bases de datos del SIGE y del GRD. De manera idéntica a la capa de negocio se utiliza el intérprete de PHP integrado al servidor Apache con la extensión php5_pgsql para acceder al servidor de base de datos PostgreSQL.

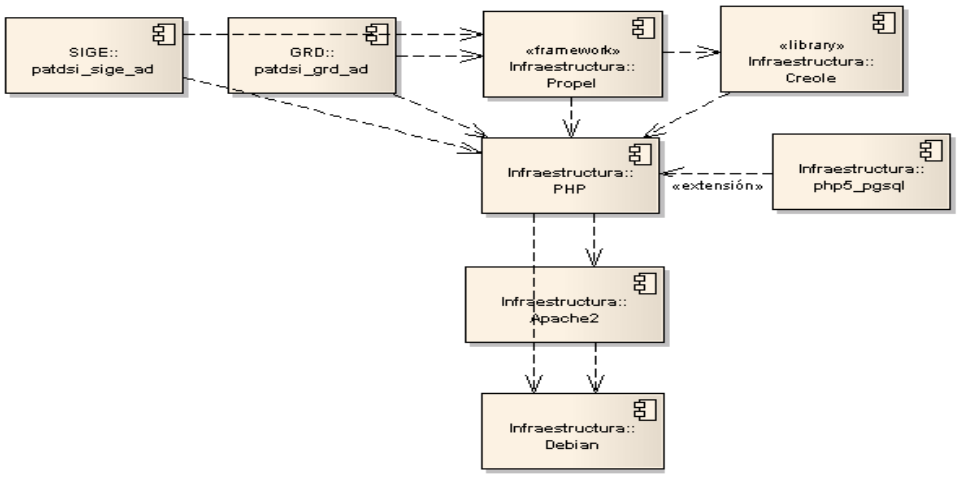


Fig. 22 Componentes de la capa de acceso a datos

Capa de Datos

La capa de datos (Fig. 23) es íntegramente la base de datos de PATDSI con los esquemas propios del SIGE y los del GRD. La base de datos se encuentra en un servidor con PostgreSQL y sistema operativo Debian.

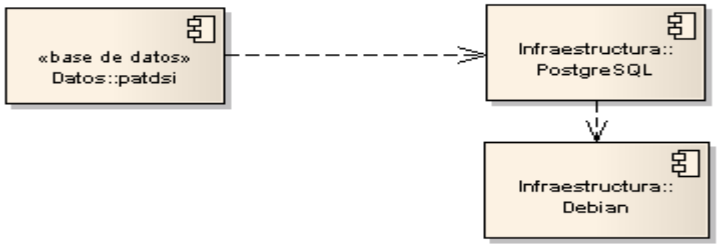


Fig. 23 Componentes de la capa de datos

Infraestructura

Los componentes de infraestructura (Fig. 24) se han presentado de conjunto con cada una de las capas y corresponden a las librerías de terceros o desarrolladas por el centro y a los componentes software del soporte como navegadores, servidores, intérpretes, extensiones, y sistema operativo.

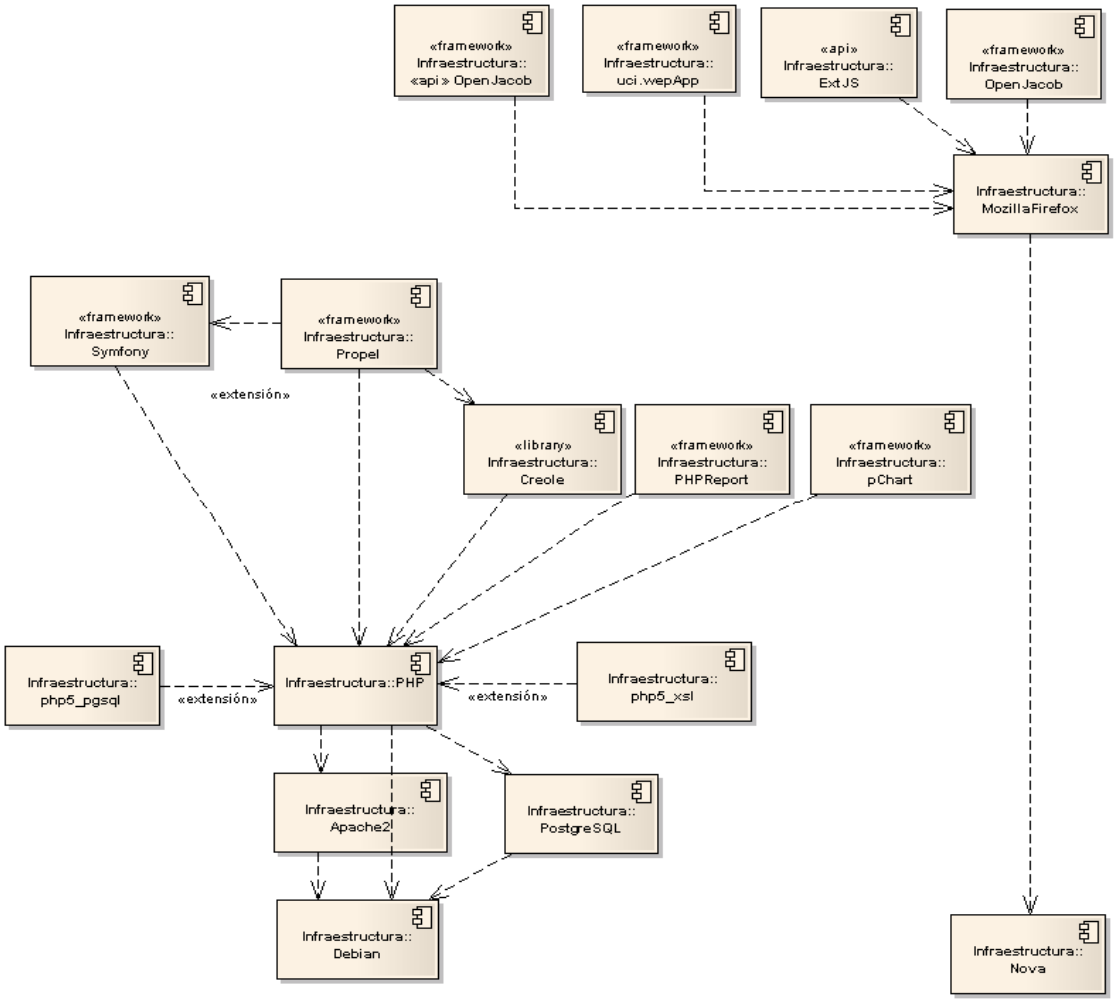


Fig. 24 Componentes de infraestructura

10.5 El sistema (como la integración de las capas)

Se puede observar el conjunto resultante de la integración de las capas (Fig. 25) destacándose las relaciones de dependencia que se establece entre el nivel superior y el inmediato inferior.

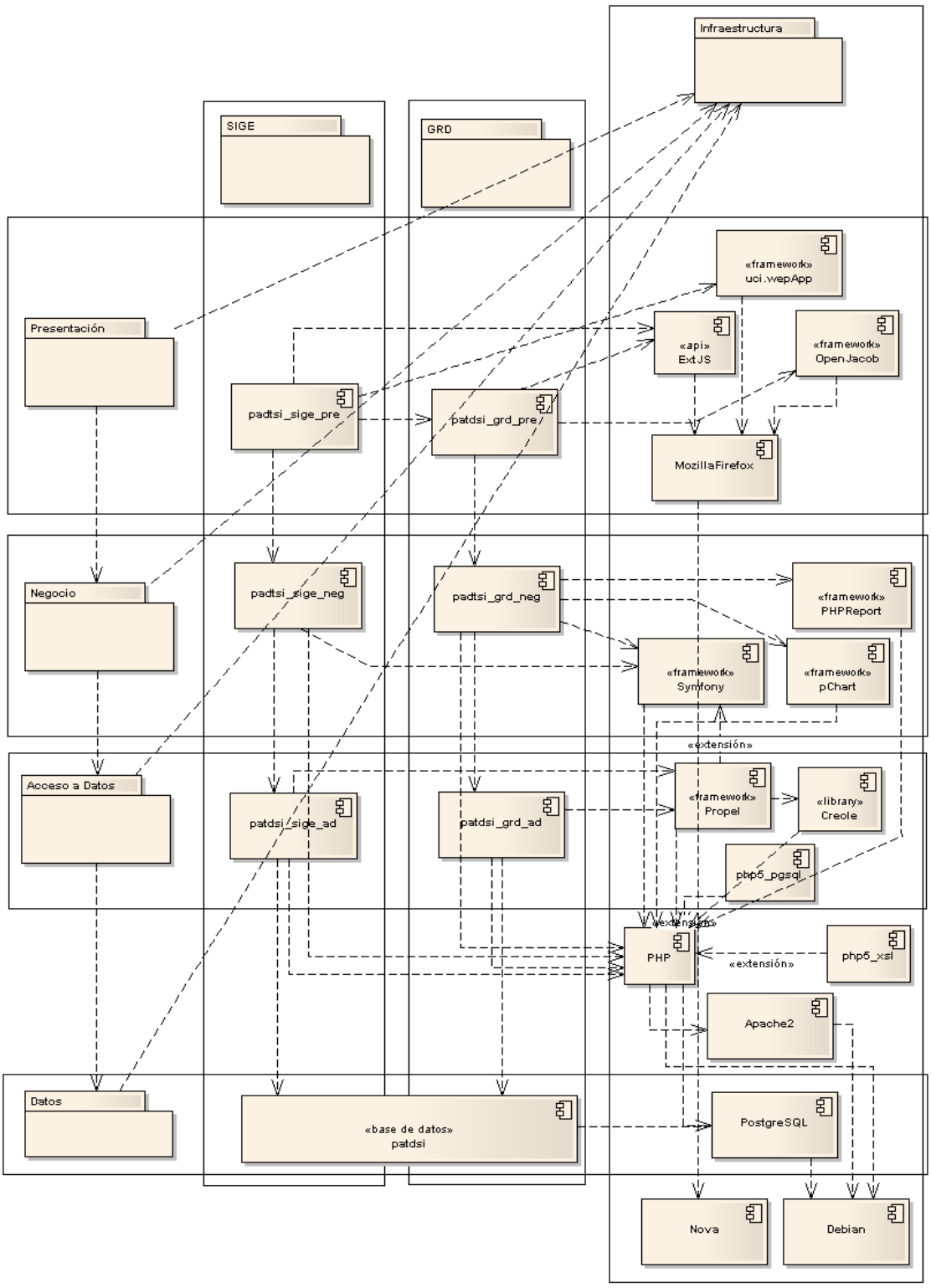


Fig. 25 Diagrama de Componentes de PATDSI - SIGE

Vista de Despliegue

Dado el carácter institucional de la ONE en forma jerárquica, y las restricciones por bajas prestaciones de la red se debe instalar el sistema en cada una de las oficinas estadísticas lo que determina 4 escenarios de despliegue: municipal (incluido Isla de la Juventud), territorial (correspondiente a cada provincia), exclusivo de la ONE nacional, integrado al ERP cubano, y adicionalmente se revisará el todo considerando la integración de los escenarios mencionados a nivel de país.

Despliegue en la ONE del Municipio

Las oficinas municipales de la ONE tienen a su disposición un set de clientes ligeros con servidor que dependiendo de la cantidad de CI del municipio oscila entre 3 y 10 clientes (Fig. 26), otro hardware posible podría ser 486, Pentium I, II, II ó IV, en todos los casos se garantiza la existencia de una red al interior de la oficina con cobertura para todas las máquinas desde las que se va a explotar el sistema incluida el servidor. Las prestaciones de la aplicación se extienden al intercambio con el ERP desde donde se podrían recuperar los datos directamente y a través del servicio correspondiente o mediante réplica se enviaría a la oficina estadística. Los CI que cuenten con posibilidades de acceso a la red de la ONE municipal y no dispongan del ERP cubano podrán acceder al sistema de forma restringida y segura (únicamente al módulo de entrada de datos)

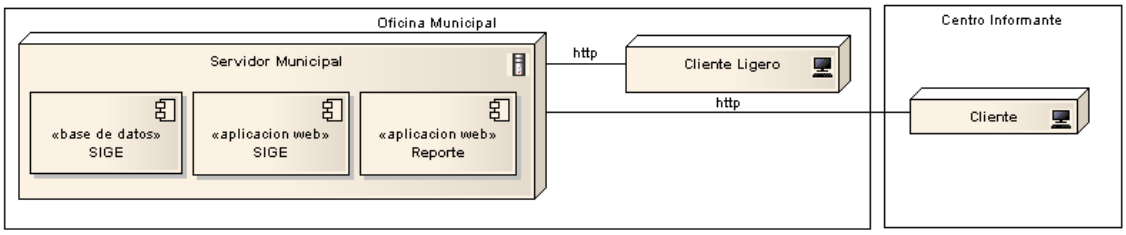


Fig. 26 Despliegue en las ONE municipales

Despliegue en la ONE de Provincia

Las ONE de provincia de manera similar a las municipales poseen el sistema completo (Fig. 27) pero además reciben la información proveniente de los municipios, en casos específicos reciben información directamente de los CI (aunque en futuro cercano no será así, debe ser tenido en cuenta). La información puede llegar por diferentes vías[7]:

1) Replicación directa entre servidores de bases de dato (cuando las prestaciones de la red lo posibiliten) entre el nivel municipal y el provincial.

2) Actualización de la base de datos a través de una copia diferencial de la base de datos de la ONE municipal a la ONE provincial, en cuyo caso el envío y la actualización se realizan de forma automatizadas a un FTP en provincia.

3) El envío por vías disimiles, en el peor de los casos, de la información digital exportada directamente a un archivo luego podría enviarse por ejemplo a través del correo o de mano a mano en un medio de almacenamiento.

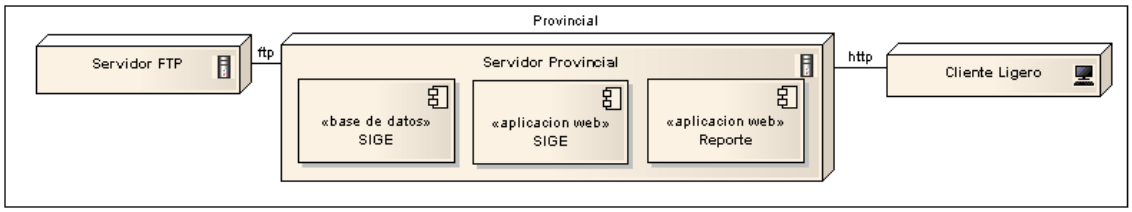


Fig. 27 Despliegue en las ONE provinciales

[7] Independientemente de la tendencia paulatina a la informatización de la sociedad es necesario considerar, dada la heterogeneidad de condiciones, varias variantes posibles que garanticen la transferencia de la información.

Despliegue Exclusivo de la ONE Nacional

La ONE nacional por su parte recepciona la información de cada una de las ONE provinciales, por tanto se consideran las mismas vías de transferencia de la información vistas para las provincias, la diferencia radica en que el concentrado de toda esta información (por demás acumulativa a lo largo de los años) determina la necesidad de formas eficientes de acceso y consulta de la misma, problema que se ha abordado a través del desarrollo de un Datawarehouse[8] (Fig. 28)

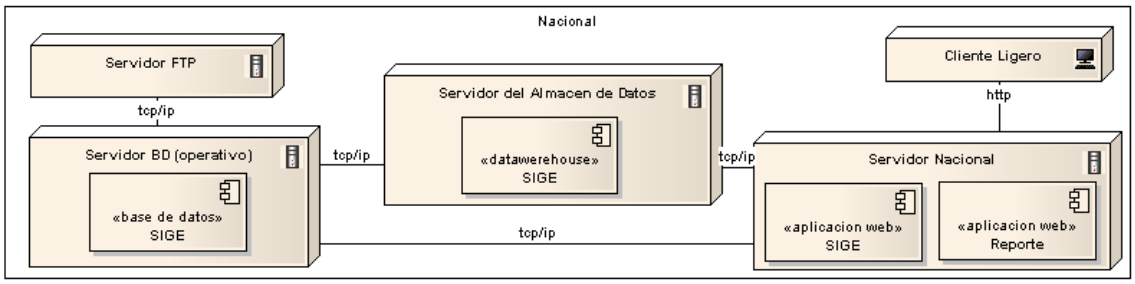


Fig. 28 Despliegue en la ONE nacional

Despliegue Integrado al ERP

La integración de la aplicación al ERP sopesa por la posibilidad de recuperar la información estadística directamente de los registros primarios, por la facilidad de que los CI tengan acceso a la digitación de la información que no pueda ser directamente obtenida (como sería el caso de las encuestas) por la garantía de disponer de los modelos estadísticos actualizados, entre otras prestaciones que podrían ser de interés para los CI. Teniendo en cuenta ello se considera que el sistema debe estar físicamente desplegado en el sitio donde resida el ERP que presta servicio a todas las empresas en el municipio, y que la ONE municipal recibirá vía réplica la información estadística correspondiente. Se encontrarían específicamente en el ERP el módulo de entrada de datos. El interés es puramente municipal ya que responde al carácter jerárquico de la organización donde es suficiente con que la base sea provista con la información requerida.

[8] Dado que es otro de los proyectos asumidos por el CENTALAD se han omitido los detalles específicos de este desarrollo para más información consultar la documentación respectiva.

Despliegue Nacional (como integración de los anteriores)

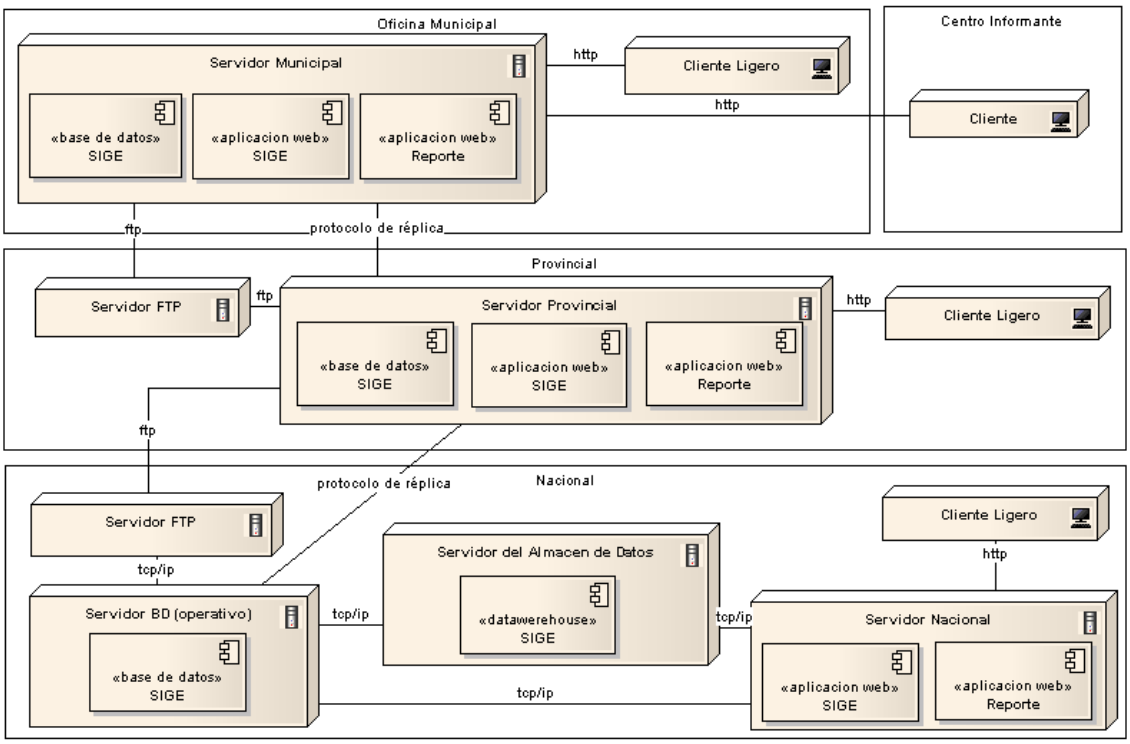


Fig. 29 Despliegue nacional

Vista de Datos

El diseño del modelo de datos se ha elaborado teniendo en cuenta la necesidad de una flexibilidad total en la representación de la información, puesto que el sistema debe soportar el diseño de los modelos estadísticos tanto de la estadística continua como las encuestas periódicas de forma dinámica. Es posible identificar varios subsistemas de entidades que responden a funciones específicas:

Subsistema de Definición de Modelo: el conjunto de entidades que contiene la información que define la estructura de los modelos (Fig. 31), o sea almacena la información de la digitalización de los modelos.

Subsistema de Definición de Encuesta: el conjunto de entidades que contiene la información que define la estructura de las encuestas, o sea almacena la información de la digitalización de las encuestas.

Subsistema de Registros y Clasificadores: este conjunto de entidades contiene la información de los registros, clasificadores, clasificaciones, centros informantes e historial de cambios de las clasificaciones de los centros informantes.

Subsistema de Compatibilización: este conjunto de entidades contiene la información necesaria para facilitar la importación y exportación a Microset NT.

Subsistema de Almacenamiento de Datos de los Modelos: este conjunto de entidades contiene la información de registro de entradas a cada uno de los modelos completados con información estadística de un determinado centro informante, además está formado par cada una de las tablas correspondientes a las páginas de los modelos, que se crean dinámicamente cada vez que elabora un nuevo modelo.

Subsistema de Almacenamiento de Datos de las Encuestas: este conjunto de entidades contiene la información de registro de entradas a cada uno de las encuestas completadas con información estadística de un determinado centro informante, así como la información contenida en la encuesta propiamente.

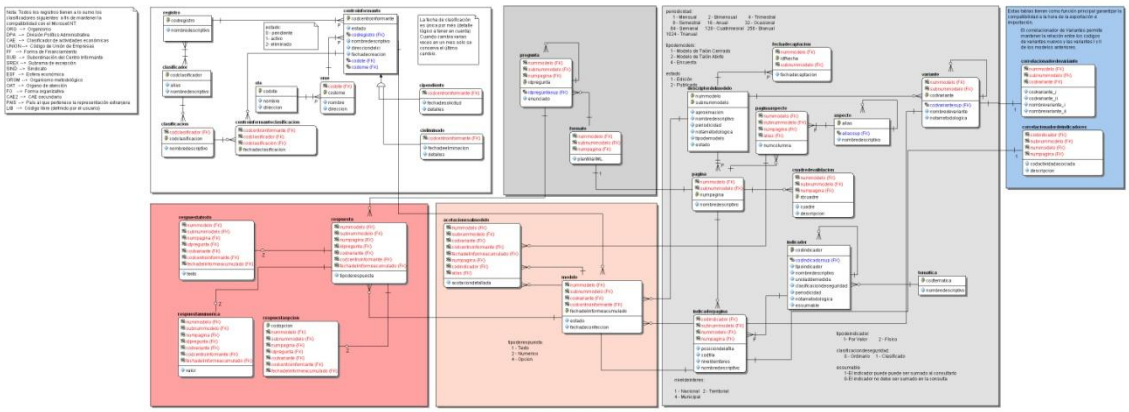


Fig. 30 Modelo de la base de datos

Breve Descripción de los Elementos que Conforman el Modelo de Datos

Subsistema de Definición de Modelo

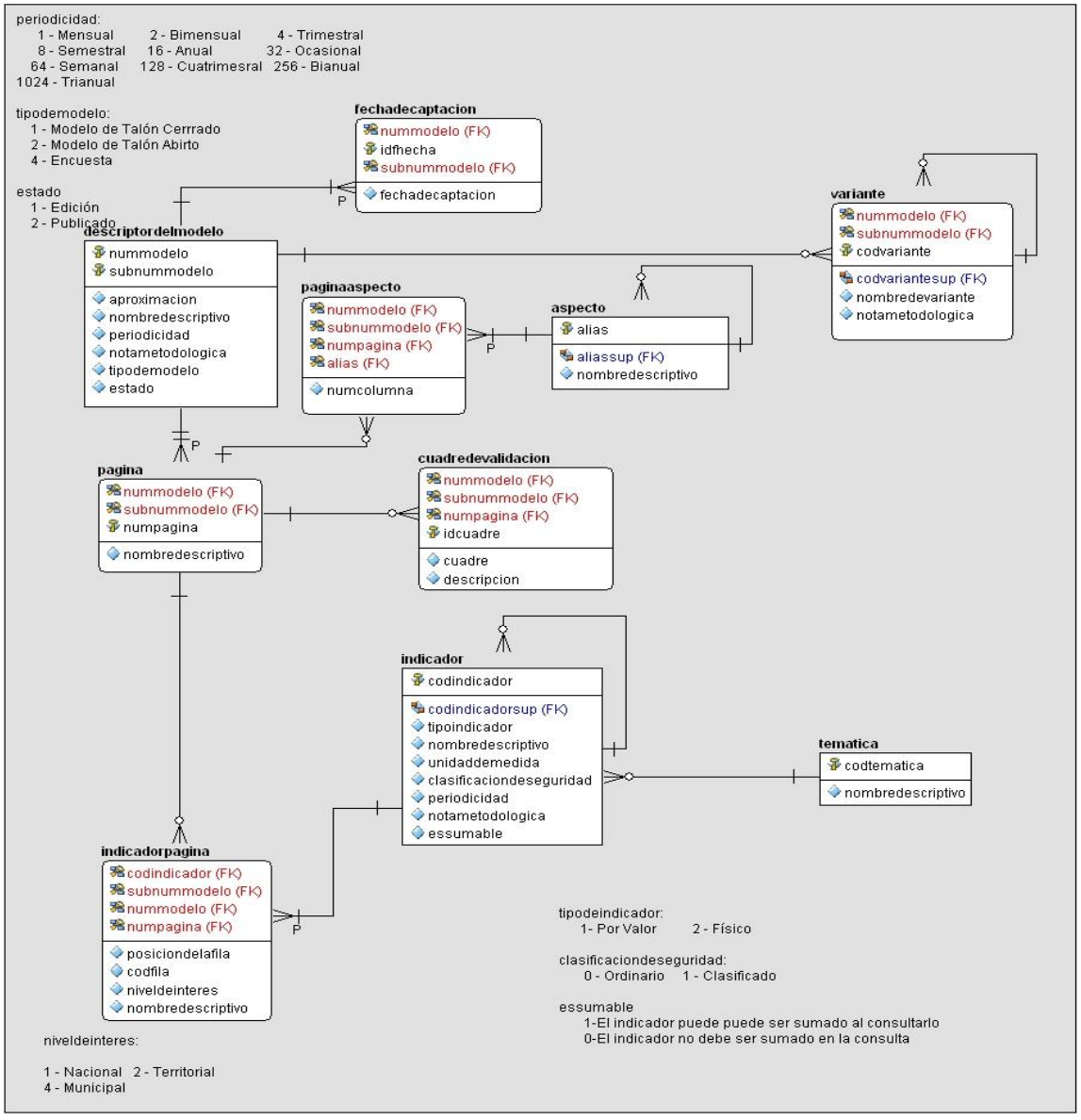


Fig. 31 Subsistema de Definición de Modelo

Elemento	Identificador	Descripción
Tablas	descriptordelmodelo	Define los diferentes modelos que conforman el SIEN tanto los de estadística continua como los de encuestas periódicas.
	fechadecaptacion	Define las diferentes fechas de captación del modelo en función de su periodicidad.
	variante	Define las posibles variantes que puede tener un modelo.
	pagina	Define las páginas asociadas al modelo.
	paginaaspecto	Define los aspectos (o columnas) correspondientes a una página determina, así como los atributos específicos de esta relación.
	aspecto	Define el conjunto de aspectos. Esta tabla contiene el clasificador de Aspectos de la ONE.
	cuadredevalidacion	Define los criterios para la validación del modelo.
	indicadorpagina	Define los indicadores (o filas) correspondientes a una página determinada, así como los atributos específicos de esta relación.
	indicador	Define el conjunto de indicadores estadísticos. Esta tabla contiene el clasificador único de indicadores (CUI) de la ONE.
	tematica	Define la temática a la que pertenece un conjunto de indicadores.

Subsistema de Definición de Encuesta

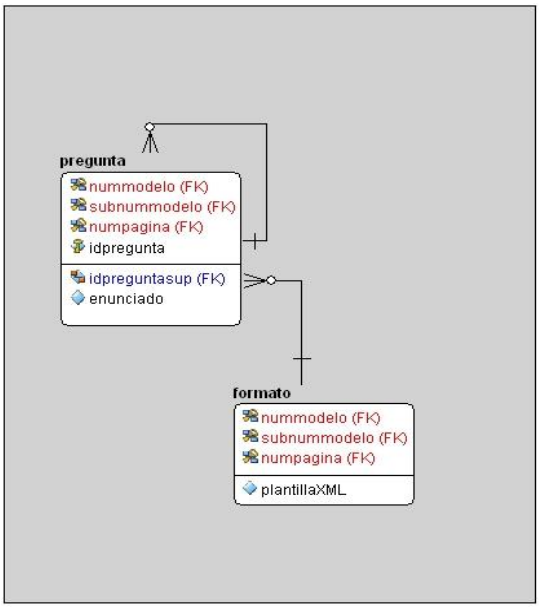


Fig. 32 Subsistema de Definición de Encuesta

Elemento	Identificador	Descripción
Tablas	formato	Contiene el diseño de la encuesta representado en XML.
	pregunta	Define cada una de las preguntas que conforman la encuesta.

Subsistema de Registros y Clasificadores

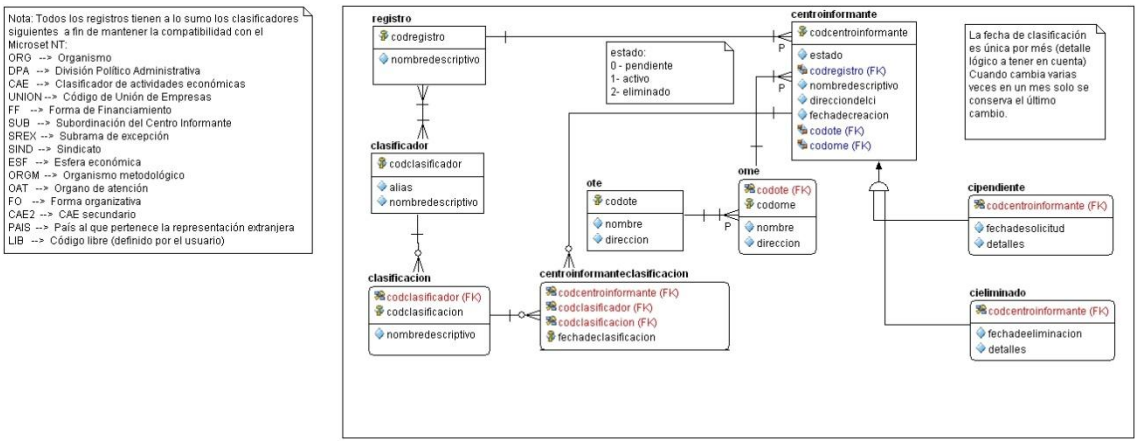


Fig. 33 Subsistema de Registros y Clasificadores

Elemento	Identificador	Descripción
Tablas	registro	Define los registros establecidos en los que se agrupan los centros informantes.
	clasificador	Define los clasificadores que manejan los registros.
	clasificacion	Define los valores (clasificaciones) que puede tener un clasificador.
	centroinformante	Define un centro informante.
	ote	Define a una oficina estadística territorial.
	ome	Define a una oficina estadística municipal.
	centroinformanteclasificacion	Define la clasificación asociada a un centro informante, conservando el histórico de estos cambios.
	cipendiente	Define los centros informantes pendientes de aprobar
	cieliminado	Define los centros informantes que han sido eliminados.

Subsistema de Compatibilización

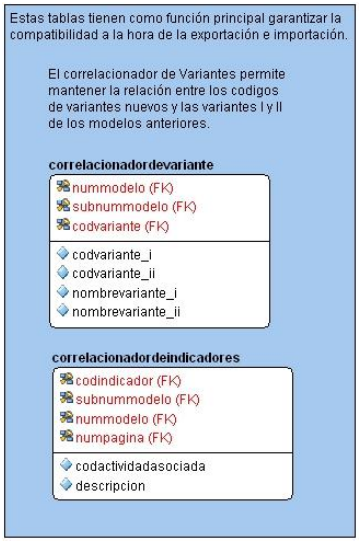


Fig. 34 Subsistema de Compatibilización

Elemento	Identificador	Descripción
Tablas	correlacionadordevariantes	Define la variante de MicroSet NT, garantizando la compatibilidad con el nuevo formato.
	correlacionadordeindicadores	Define el indicador estadístico de MicroSet NT, garantizado la compatibilidad con el nuevo formato.

Subsistema de Almacenamiento de Datos de los Modelos

La persistencia de los datos estadísticos propiamente comprende las dos tablas que se presentan a continuación y el conjunto de las tablas que se crean dinámicamente para cada página de cada nuevo modelo publicado en la aplicación, de esta forma se obtienen mejores resultados en el rendimiento y se optimiza el espacio ocupado.

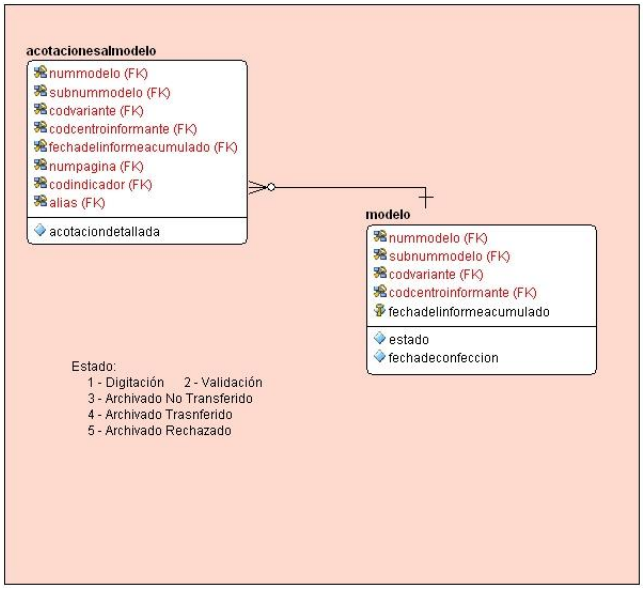


Fig. 35 Subsistema de Almacenamiento de Datos de los Modelos

Elemento	Nombre	Descripción
Tablas	modelo	Define las entradas a cada uno de los modelos estadísticos entregados por los centros informantes.

	acotacionesalmodelo	Define las acotaciones (aclaraciones especiales) realizadas al modelo entregado por el centro informante en cuestión.

Subsistema de Almacenamiento de Datos de las Encuestas

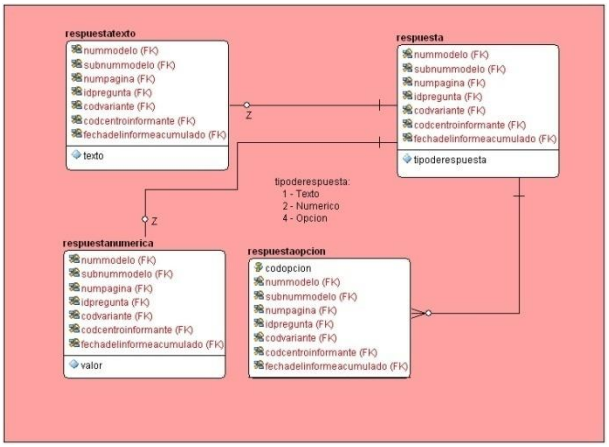


Fig. 36 Subsistema de Almacenamiento de Datos de las Encuestas

Elemento	**Nombre**	**Descripción**
Tablas	respuesta	Define el tipo de respuesta dada a una determinada pregunta.
	respuestatexto	Contiene el valor de la respuesta tipo textos para una pregunta.
	respuestanumerica	Contiene el valor de la respuesta tipo numérico para una pregunta.
	respuestaopcion	Contiene el valor de la respuesta tipo opción para una pregunta.

Conclusiones Parciales

En este capítulo se realizaron las propuestas de entorno de desarrollo y de arquitectura del sistema utilizando el enfoque de las 4+1 vistas que propone RUP teniendo en cuenta los objetivos y restricciones impuestas, en resumen:

- Se presentaron las herramientas y tecnologías libres sobre las que se elabora y soporta la aplicación.

- Se identificaron los elementos arquitectónicos, estilos y patrones presentes en la solución realizandose una descripción de los estilos multicapas, modelo vista controlador, maquina virtual y orientado a componentes para el desarrollo sobre la web, la relación entre ellos en la concepción del sistema y el ambiente sobre el que se desempeñará la solución.

- Se establecieron las librerías y frameworks a utilizar que aportan los principios que orientan el diseño y garantizan la evolución del sistema con una alta calidad.

CAPÍTULO 3: Evaluación de la Propuesta de Arquitectura

En este capítulo se realiza la evaluación de la arquitectura partiendo primeramente del análisis del impacto de cada una de las decisiones arquitectónicas al aplicar los diferentes estilos, posteriormente se aplicará la técnica de evaluación basada en escenarios con la aplicación del método de evaluación ATAM.

Misión de la Evaluación

Dado que el proyecto se encuentra en fase de desarrollo se ha basado la evaluación en el diseño temprano de la arquitectura puesto que a pesar de que con el término de la fase de elaboración se venció el hito de la definición de la arquitectura, la misma no está completamente implementada. Según Kazman, el primer paso para la evaluación de una arquitectura es conocer qué es lo que se quiere evaluar, de esta forma, es posible establecer la base para la evaluación, la cuestión es por tanto valorar en qué medida la aplicación es multiplataforma, robusta, flexible y extensible como establece el objetivo del presente trabajo. Las salidas que se obtienen responden a la aplicación del método ATAM que produce el conjunto de escenarios priorizados, el árbol de utilidad, los puntos sensibles, las relaciones entre atributos, los riesgos y los no riesgos.

Evaluación Basada en Escenarios

Un escenario consta de tres partes: el estímulo, el contexto y la respuesta. El estímulo es la parte del escenario que explica o describe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema. El contexto describe qué sucede en el sistema al momento del estímulo. La respuesta describe, a través de la arquitectura, cómo debería responder el sistema ante el estímulo. Entre las ventajas de su uso están:

- Simples de crear y entender
- Poco costosos y no requieren mucho entrenamiento

- Efectivos

Las técnicas de evaluación basadas en escenarios cuentan con dos instrumentos de evaluación relevantes: el árbol de utilidad propuesto por Kazman y los perfiles, propuestos por Bosch. A continuación se explica el instrumento de evaluación árbol de utilidad por ser el utilizado en ATAM.

Instrumento de Evaluación Árbol de Utilidad

Un Árbol de Utilidad (Utility Tree) es un esquema en forma de árbol que presenta los atributos de calidad de un sistema de software, refinados hasta el establecimiento de escenarios que especifican con suficiente detalle el nivel de prioridad de cada uno. La intención del uso del árbol de utilidad es la identificación de los atributos de calidad más importantes para un proyecto particular. El nodo raíz representa la utilidad general del sistema. Los atributos de calidad asociados al mismo conforman el segundo nivel del árbol, los cuales se refinan hasta la obtención de un escenario lo suficientemente concreto para ser analizado y otorgarle prioridad a cada atributo considerado. Cada atributo de calidad perteneciente al árbol contiene una serie de escenarios relacionados, y una escala de importancia y dificultad asociada, que será útil para efectos de la evaluación de la arquitectura. Se impone la necesidad de una revisión inicial de cuáles son los atributos de calidad importantes para la arquitectura.

Atributos de Calidad para la Arquitectura del Sistema

Los escenarios proveen un vehículo que permite concretar y entender atributos de calidad identificados a través de los requisitos no funcionales que por sí solos son demasiado imprecisos. En (Barbacci et al., 1995) la calidad de software se define como el grado en el cual el software posee una combinación deseada de atributos. Tales atributos son requerimientos adicionales del sistema (Kazman et al., 2001), que hacen referencia a características que éste debe satisfacer, diferentes a los requerimientos funcionales. Estas características o atributos se conocen con el nombre de atributos de calidad, los cuales se definen como las propiedades de un servicio que presta el sistema a sus usuarios (Barbacci et al. 1995) y están íntimamente relacionados con los requisitos no funcionales. El estándar ISO/IEC 9126 ha sido desarrollado en un intento de identificar los atributos clave de calidad para un producto de software (S. Pressman, 2000), referente a este estándar (Losavio et al., 2003) propone una adaptación del modelo ISO/IEC 9126 de calidad de software para efectos de la evaluación de arquitecturas de

software. El modelo se basa en los atributos de calidad que se relacionan directamente con la arquitectura: funcionalidad, confiabilidad, eficiencia, mantenibilidad y portabilidad.

Atributo	**Sub-atributo**	**Descripción**
Funcionalidad (*Functionality*)	Adecuación	Asociado al refinamiento de los diagramas de secuencia
	Exactitud	Asociado a los componentes con las funciones responsables de los cálculos
	Interoperabilidad	Asociado a la identificación de conectores de comunicación con sistemas externos
	Seguridad	Asocaido a la existencia de los mecanismos o dispositivos que garantizan explícitamente los principios de seguridad.
Confiabilidad (*Reliability*)	Tolerancia a fallas	Asociados a la existencia de mecanismos o dispositivos de software para manejar excepciones.
	Recuperabilidad	Asociado a la existencia de mecanismos o dispositivos de software para restablecer el nivel de desempeño y recuperar datos
Eficiencia (*Performace*)	Desempeño	Asociado a la complejidad de los componentes involucrados en un flujo de ejecución para una funcionalidad
	Utilización de recursos	Asociado a la relación de los componentes en términos de espacio y tiempo
Mantenibilidad (*Maintainability*)	Acoplamiento	Asociado a las interacciones entre componentes
	Modularidad	Asociado al número de componentes que dependen de un componente
Portabilidad (*Portability*)	Adaptabilidad	Asociado a la presencia de mecanismos de adaptación.
	Instalabilidad	Asociado a la presencia de mecanismos de instalación
	Coexistencia	Asociado a la presencia de mecanismos que faciliten la coexistencia
	Reemplazabilidad	Asociado a la existencia de componentes reemplazables para cada componente

Funcionalidad: habilidad del sistema para realizar el trabajo para el cual fue concebido.

Confiabilidad: es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo bajo ciertas condiciones de estrés.

Eficiencia (Performance[9]): es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12). El desempeño de un sistema se refiere a aspectos temporales del comportamiento del mismo, a la capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o el número de eventos procesados en un intervalo de tiempo además a la cantidad de comunicación e interacción existente entre los componentes del sistema.

Mantenibilidad: posibilidad de someter a un sistema a reparaciones y evolución y capacidad de modificar el sistema de manera rápida y con un bajo costo.

Portabilidad: Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación tanto hardware, software o una combinación de los dos.

Método ATAM

Como se explicó anteriormente ATAM se concentra en la identificación de los estilos arquitectónicos utilizados, los cuales representan los medios empleados por la arquitectura para alcanzar los atributos de calidad previamente identificados en correspondencia con los objetivos y restricciones, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas.

La parte principal del ATAM consiste de nueve pasos. Estos pasos se dividen cuatro grupos:

- Presentación, donde la información es intercambiada.
- Investigación y análisis, donde se valoran los atributos claves de calidad requeridos, uno a uno con las propuestas arquitectónicas.
- Pruebas, donde se revisan los resultados obtenidos contra las necesidades relevantes de los interesados o "*stakeholders*".

[9] Performace: s. funcionamiento, rendimiento; ejecución, cumplimiento, desempeño, realización; interpretación, representación; función, acto, actuación. Se ha traducido en el contexto como el atributo de calidad eficiencia con los sub atributos desempeño y utilización de recursos.

- Informes, donde se presentan los resultados del ATAM.

Con más nivel de detalle en cada uno de estos pasos para cada grupo se procede como define ATAM y se explica a continuación:

Presentación

Paso #1 Presentar el ATAM: el líder del equipo de evaluación describe el método a los participantes, fija las expectativas y responde las preguntas que puedan surgir.

Paso #2 Presentar las pautas del negocio: un representante del proyecto (por ejemplo, el gestor de proyecto o el cliente) describe las metas del negocio que motivan el esfuerzo de desarrollo y que se convertirán en las pautas primordiales de la arquitectura (por ejemplo, alta disponibilidad o alta seguridad).

Paso #3 Presentar la arquitectura: el arquitecto describe la arquitectura, enfocándose en cómo esta sigue las pautas del negocio.

Investigación y análisis

Paso #4 Identificar las propuestas arquitectónicas: las propuestas arquitectónicas son identificadas por el arquitecto, pero no son analizadas.

Paso #5 Generar el árbol de utilidad de los atributos de calidad: los atributos de calidad que comprometen la utilidad del sistema son obtenidos, especificados en escenarios (con estímulos y respuestas) y priorizados.

Paso #6 Analizar las propuestas arquitectónicas: basados en los escenarios de mayor prioridad identificados en el paso 5, las propuestas arquitectónicas que cumplen con estos escenarios, son obtenidas y analizadas (por ejemplo, una propuesta arquitectónica que logra una meta de performance,

será objeto de un análisis de performance). Durante este paso los riesgos arquitectónicos, los no riesgos, los sensitivity points y tradeoff points son identificados.

Pruebas

Paso #7 Lluvia de ideas y priorización de escenarios: un gran conjunto de escenarios es obtenido por todos los interesados. Este conjunto es priorizado mediante votación.

Paso #8 Analizar las propuestas arquitectónicas: en este paso se reitera las actividades del paso 6 utilizando el ranking de escenarios del paso 7. Estos escenarios se consideran casos de prueba para confirmar el análisis realizado hasta ahora, se pueden descubrir nuevas propuestas arquitectónicas, riesgos, no riesgos, puntos sensibles y relaciones entre atributos de calidad, que son documentados.

Informes

Paso #9: Presentar los resultados: basados en la información recogida durante el ATAM (propuestas, escenarios, árbol de utilidad, riesgos, no riesgos, puntos sensibles y relaciones entre atributos de calidad), el equipo de ATAM presenta los resultados a los interesados.

La aplicación de los pasos antes mencionado nos permitió definir el árbol de utilidad, la identificación de los escenarios, los riesgos y no riesgos, los puntos sensibles y las relaciones entre los atributos que a continuación se analizan

Evaluación de la Arquitectura a través de ATAM

Escenarios

En la ejecución de la evaluación como resultado de las actividades de aplicar el método ATAM se obtiene en el paso 5 el árbol de utilidades, que se enfoca fundamentalmente en 5 atributos de calidad (correspondientes a la adaptación del modelo ISO/IEC 9126 propuesta por Losavio): desempeño, funcionalidad, mantenimiento, portabilidad y confiabilidad. Como nodos hijos de estos se encuentran los sub-atributos que constituyen un refinamiento de estos atributos de calidad.

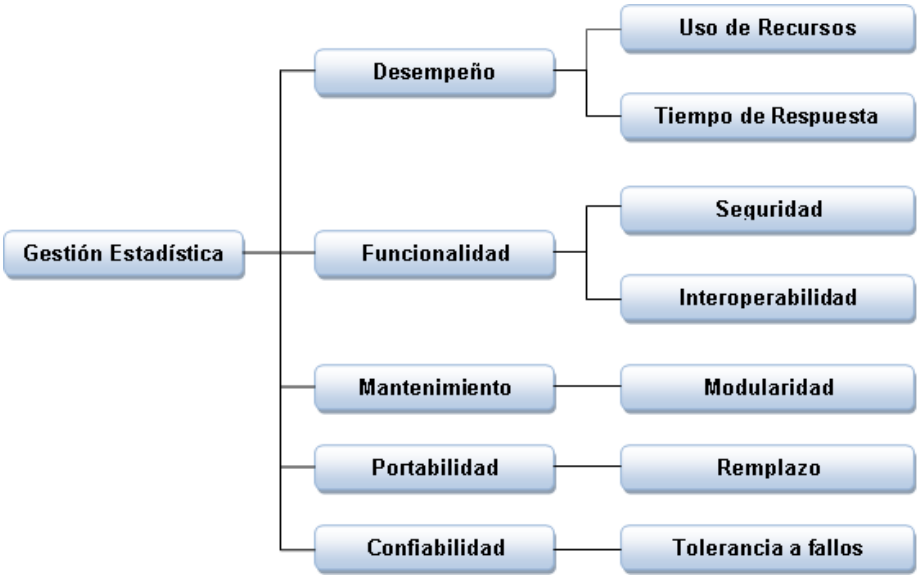


Fig. 37 Árbol de utilidad del sistema

Como establece el paso 7 de ATAM en una sesión de tormenta de ideas con el equipo de evaluación fueron identificados los diferentes escenarios que responden a los intereses plasmados por los clientes y que motivaron el desarrollo. Como parte de la refinación de los mismos fueron agrupados y categorizados tomando en consideración el árbol de generación de utilidades obtenido, procediéndose posteriormente a la asignación de la prioridad de cada uno de ellos teniendo en cuenta la importancia del atributo y la estimación de la dificultad para llevarlo a cabo.

Análisis de los escenarios

Se procede al análisis de los escenarios de mayor prioridad arquitectónica, teniendo en consideración cómo reacciona la arquitectura del sistema. Como resultado se obtiene un listado con los puntos sensibles, las relaciones entre atributos, riesgos y no riesgos.

A continuación se muestra el análisis detallado realizado a los principales escenarios involucrados. Se describe para cada uno el estímulo, el ambiente en el que se encuentra el sistema, la respuesta y una explicación de cómo responde la arquitectura a este evento.

El escenario 12 (Tabla 8) evalúa que la arquitectura propuesta es multiplataforma al analizar cómo los estilos multicapa y máquina virtual potencian el atributo Portabilidad – Remplazo, entonces es posible identificar en este escenario:

No riesgos

- La arquitectura propuesta es multiplataforma.

Puntos sensibles

- Si el intérprete de Java Script no implementa el estándar ECMAScript 262 entonces la implementación del componente de presentación podría no ser portable. (Ej.: IE no siempre implementa los estándares completamente, incluso versiones antiguas de otros navegadores pudiesen no implementar o implementar parcialmente el estándar).

El escenario 11 (Tabla 9) evalúa que la arquitectura propuesta es flexible al analizar cómo los estilos multicapa y orientado a componentes potencian el atributo Mantenimiento – Modularidad, además se tiene en cuenta la decisión de concebir un sistema modular por lo que es posible identificar en este escenario:

No riesgos

- La arquitectura propuesta es flexible respecto a los requisitos del cliente.

El escenario 10 (Tabla 10) evalúa que la arquitectura propuesta es extensible (se pone de ejemplo este escenario pero además se tuvo en cuenta el 9) al analizar cómo el estilo orientado a componentes potencian el atributo Mantenimiento – Modularidad.

No riesgos

- La arquitectura propuesta es extensible lo que facilita agregar nuevas funcionalidades sin un mayor impacto sobre el sistema.

Tabla 8 Análisis del Escenario 12

Escenario #12:	**Descripción:**	Se decide instalar la apliación en otra plataforma, para ello se instalan los componentes de remplazo sobre la plataforma que garantizan la portabilidad. (A, A)
Atributo(s):	Portabilidad - Remplazo[10]	
Ambiente:	Despliegue del sistema	
Estímulo:	Se despliega el sistema sobre una plataforma dada	
Respuesta:	Se remplazan las tecnologías sobre las que se asienta el sistema de manera transparente para este.	
Decisiones arquitectónicas	**Razonamiento**	
Estilos multicapa y máquina virtual	Los estilos y tecnologías utilizados en la aplicación determinan la portabilidad del sistema, a saber: Estilo multicapa: la capa de presentación consume las acciones de la capa de negocio a través de las solicitudes AJAX para ello utiliza el protocolo HTTP por lo que es independiente de la plataforma en la que se despliegue las restantes capas siempre que se garanticen las acciones. De manera similar la capa de negocio consume los servicios de la de acceso a datos, quien a su vez se independiza del tipo de base de datos a través de una capa de abstracción del gestor específico realizándose el intercambio vía el protocolo TCP/IP, por tanto no es determinante la plataforma sobre la que se encuentra la capa de datos incluso si se cambia el tipo de gestor que la soporta. Máquina virtual: la ejecución del sistema queda abstraída de la plataforma de hardware y del middleware provisto por el sistema operativo siempre que se disponga de una implementación de la máquina virtual para la plataforma. Se destaca que la implementación de la capa de presentación está programada en JavaScript lenguaje interpretado y estandarizado por el ECMAScript 262 3ra Edición disponible en la mayoría de los navegadores actuales y la capa de negocio implementada en PHP5 lenguaje interpretado desarrollado por el PHP Group y estándar de facto del lenguaje. Tecnologías: Los intérpretes de los lenguajes de JavaScript y PHP se encuentran implementados para la mayoría de las plataformas por tanto son reemplazables.	

[10] Portabilidad y remplazo responden como atributos de calidad al objetivo planteado de que el diseño sea multiplataforma.

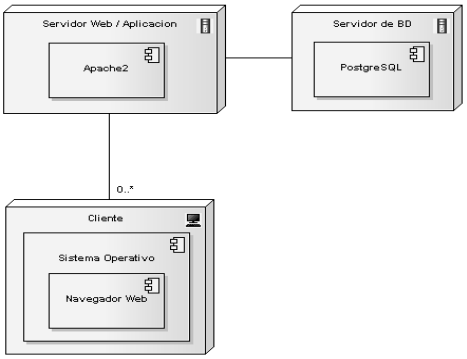
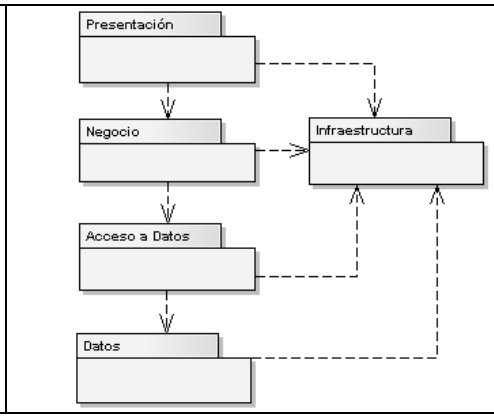
Diagrama Arquitectura		

Tabla 9 Análisis del Escenario 11

Escenario #11:	**Descripción:**	Se desea intalar solo la parte del sistema que funcionalmente responde a sus intereses (A, A)
Atributo(s):	Mantenimiento - Modularidad	
Ambiente:	Despliegue del sistema	
Estímulo:	Se despliega una parte del sistema de acuerdo con los intereses de uso del usuario	
Respuesta:	La parte del sistema desplegada es autónoma en su función.	
Decisiones arquitectónicas	**Razonamiento**	
Estilos multicapa y orientado a componente. Decisión diseño modular	El estilo multicapa y el orientado a componentes junto al diseño modular potencian que las partes que brindan un valor funcional completo y operen de manera autónoma e independiente de las demás, a saber: Estilo multicapa: es posible centralizar los servicios, de esta manera varias clases de usuarios (en el contexto del negocio pudieran ser tanto oficinas, como centros informantes) podrían utilizar los servicios de alojamiento de datos (servidor de base de datos) o de alojamiento de sistemas web (servidor web / aplicación) para desplegar los elementos comunes a la(s) parte(s) que se desean instalar. Orientado a componente: favorece gracias al encapsulamiento que las funcionalidades prestadas por los componentes estén concentradas al interior de estos dependiendo en el peor de los casos de otros componentes cuyos servicios consumen, de esta forma es posible prescindir de aquellos componentes que no se requieren para la parte de la aplicación que se desea instalar. Diseño modular: favorece que todas aquellas funcionalidades que agregan un valor completo para el usuario estén agrupadas en una parte de muy baja granularidad (Ej.: el módulo de entrada de datos, y el generador de reportes dinámicos son los más utilizado)	

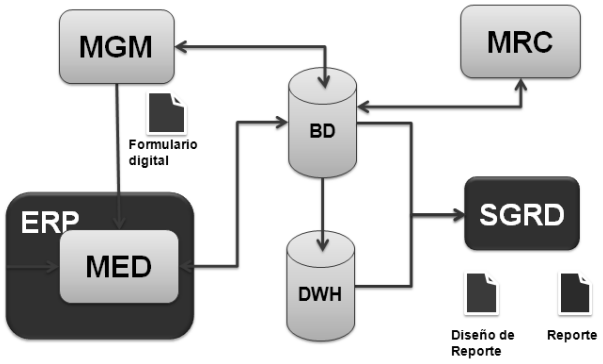
Diagrama Arquitectura	

Tabla 10 Análisis del Escenario 10

Escenario #10:	**Descripción:**	Desplegar un nuevo componente en el sistema sin volver a desplegar la aplicación que está corriendo(A, A).
Atributo(s):	Mantenimiento - Modularidad	
Ambiente:	Extensión del sistema	
Estímulo:	El usuario despliega un nuevo componente desarrollado sin necesidad de reinstalar el sistema completo.	
Respuesta:	El sistema incorpora el componente y las funcionalidades provistas por este si afectar el resto del sistema.	
Decisiones arquitectónicas	**Razonamiento**	
Estilo orientado a componente.	El estilo orientado a componente potencia los atributos Mantenimiento Modularidad con lo cual se logra el objetivo de la extensibilidad del sistema. De manera concreta esto se logra en la arquitectura a través de la creación de componentes pre-configurados o de plugins en Ext JS para la extensión de la capa de presentación, del lado del servidor se pueden agregar acciones a los módulos existentes, así como nuevos módulos y aplicaciones.	
Diagrama Arquitectura		

Resultados

Se analizaron para la arquitectura propuesta un total de 13 escenarios identificándose:

Puntos Sensibles

- Soportarse sobre navegadores que no implementen adecuadamente los estándares, de manera particular el estándar ECMAScript 262 3ra Edición

Puntos de relación entre atributos

Riesgos

- El incremento de la cantidad de código del lado del cliente podría afectar la robustez del sistema, en tanto que podría demorar considerablemente la carga inicial del sistema. Esto puede mitigarse implementando técnicas de carga tardía, y utilizando etiquetas iframes.
- Incompatibilidad entre los conceptos nomeclados para el SIGE 2.0 y los nomeclados para el ERP dificulten o imposibiliten la recuperación automática de los datos y con ello la adecuada integración entre ambos. Este riego puede mitigarse llegando a un acuerdo entre los negocios para que se estandaricen los conceptos nomeclados.

No Riesgos

- La arquitectura propuesta es multiplataforma.
- La arquitectura propuesta es flexible respecto a los requisitos del cliente.
- La arquitectura propuesta es extensible lo que facilita agregar nuevas funcionalidades sin un mayor impacto sobre el sistema.

Conclusiones Parciales

En este capítulo se analizó la influencia de los estilos seleccionados, se aplicó el método de evaluación de arquitecturas ATAM y la técnica basada en escenarios con el instrumento árbol de utilidad, en aras de determinar cómo la propuesta arquitectónica cumple con los atributos de calidad propuestos como objetivo. Como resultado de este proceso se identificaron algunos riesgos, no riesgos y puntos sensibles presentes en la arquitectura del sistema, con lo que se demuestra que la arquitectura propuesta satisface las expectativas del objetivo formulado.

CONCLUSIONES

Se realizó el estudio del estado del arte de algunos sistemas estadísticos, llegando a la conclusión de que si bien existen 2 sistemas desarrollados en Cuba que se ajustan adecuandamente uno es obsoleto y el segundo depende de software propietario, el resto no cumple cabalmente con los requisitos solicitados por el cliente.

- Se seleccionaron las herramientas y tecnologías libres a utilizar en el ambiente de desarrollo.

- Se describió la arquitectura de software del sistema utilizando las diferentes vistas que reccorren los niveles de abstracción, haciéndose uso de estilos y patrones que potencian atributos de calidad.

- Los diferentes entornos de despliegue del sistema le aportan flexibilidad a la arquitectura propuesta, una vez que puede ser adaptado a las necesidades particulares un centro informante que que pueda automatizar la gestión de su información estadística.

- Se confeccionó el documento de Descripción de la Arquitectura quedando plasmado las características del ambiente de desarrollo del sistema así como los elementos esenciales del Modelo de Casos de Uso, Modelo de Diseño, Modelo de Despliegue, Modelo de Implementación y Modelo de Datos.

- Se evaluó la arquitectura con la aplicación del método ATAM demostrando cómo esta potencia los atributos de calidad propuesto como objetivos.

- Se demostró que con la selección adecuada de los elementos arquitectónicos, el establecimiento de las relaciones entre estos y el ambiente, y la identificación de los principios que orienten el diseño y evolución posterior del sistema se obtuvo un diseño de arquitectura multiplataforma, robusta, flexible y extensible para el Sistema Integrado de Gestión Estadísticas 2.0 Nuragas en su desarrollo en software libre y sobre la web.

RECOMENDACIONES

- Elaborar una propuesta que formalice adecuadamente en base a la arquitectura propuesta una arquitectura tipo para el desarrollo de aplicaciones empresariales para el CENTALAD.

- Desarrollar en el marco del proyecto Paquete de Apoyo a la Toma de Decisiones Inteligentes (PATDSI) las extensiones necesarias que permitan el procesamiento matemático de la información estadística.

- Determinar los componentes elaborados para este sistema potencialmente reutilizables para que pasen a formar parte del repositorio de componentes del centro.

ANEXOS

Tabla 11 Escenarios identificados y priorizados

Atributo de Calidad	Refinamiento	Escenarios
Desempeño	Uso de Recursos	E#01: El usuario trabaja con la apliación en un entorno de bajas prestaciones (cliente ligero), el sistema tiene una demanda mínima para el hardware del cliente dado que el grueso de la aplicación se ejecuta del lado del servidor.
	Tiempo de Respuesta	E#02: El usuario realiza una acción que implica un procesamiento del lado del servidor, el sistema responde en menos de 5 segundos.
Funcionalidad	Seguridad	E#03: No se ha recibido ninguna petición de una sesión de usuario por más de 10 min. El sistema toma esta sesión como potencialmente insegura e invalida las credenciales del usuario direccionándolo a la página de autenticación (M,B)
		E#04: Un usuario no autorizado intenta acceder al sistema sin tener los permisos, el sistema no lo autoriza mostrando el mensaje pertinente (A,B)
	Interoperabilidad	E#05: Integración con la suite de herramientas para el Análisis Inteligente de Datos (M, M).
		E#06: Intercambio de información importación y exportación al formato de MicroSet NT (A,A)
		E#07: Integración con el ERP para la recuperación de la información estadística a partir de los registros de este.
Mantenimiento	Modularidad	E#09: Incorporar nuevas funcionalidades a los módulos sin detener la explotación del sistema (M, A).
		E#10: Desplegar un nuevo componente en el sistema sin volver a desplegar la aplicación que está corriendo(A, A).
		E#11: Se desea intalar solo la parte del sistema que funcionalmente responde a sus intereses.
Portabilidad	Remplazo	E#12: Se decide instalar la apliación en otra plataforma, para ello se instalan los componentes de remplazo sobre la plataforma que garantizan la portabilidad. (A, A)
Confiabilidad	Tolerancia a fallos	E#13: El usuario realiza una acción, una falla en la ejecución del lado del sistema, el usuario es informado y el sistema se recupera. (A,A)

BIBLIOGRAFÍA

Arquitectura de software. Arquitectura orientada a servicios. **Espinal Martín, Yanet. 2008.** La Habana, Cuba : s.n., 2008.

Brockwell, Pete y Richard, Davis. 2002. *Introduction to time series and forecasting.* New York : Springer-Verlag, 2002.

Camacho, Erika, Caedeso, Fabio y Nuñez, Gabriel. 2004. *ARQUITECTURAS DE SOFTWARE - GUÍA DE ESTUDIO.* 2004.

Equipo de desarrollo de PostgreSQL. 2005. *Manual de Usuario de PostgreSQL.* 2005.

Fayad, M., Schmidt, D. y Johnson, R. E. 1997. *Application Frameworks.* s.l. : Wiley & Sons, 1997.

Fowler, Martin, y otros. 2002. *Patterns of Enterprise Application Architecture.* s.l. : Addison Wesley, 2002. 0-321-12742-0.

Gamma, Erich, y otros. 1994. *Design Patterns. Elements of Reuseable Object-Oriented Software.* 1994.

Introducción a la Arquitectura de Software. **Billy Reynoso, Carlos. 2004.** Buenos Aires, Argentina : http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp, 2004.

—. **Cristiá, Maximiliano. 2007.** Rosario, Argentina : s.n., 2007.

Johnson, E. 1997. *Frameworks = Components + Patterns.* s.l. : Communications of the ACM, 1997.

Mateus, Carlos. 2004. *Desarrollo de Aplicaciones Web.* Barcelona, España : s.n., 2004.

Minitab Inc. 2009. Características de Minitab Statistical Software. *Minitab. Software para mejora de la calidad.* [En línea] Quality. Analysis. Results.®, 2009. [Citado el: 14 de mayo de 2009.] http://www.minitab.com/es-MX/products/minitab/features/.

OMG. 1999. *The Common Object Request Broker: Architecture and Specification.* 1999.

Proposal for a universal computer-oriented language. **Conway, M. 1958.** 10, s.l. : Communications of the ACM, Octubre de 1958, págs. 5-8.

Robert Lobo, Armando, Ortiz Sierra, Julio Ernesto y García Pérez, Omar Ahmed. 2008. *Sistema Integrado para la Gestión Estadística en Cuba.* La Habana, Cuba : 6ta Jornada Científica Estudiantil de la Universidad de las Ciencias Informáticas, 2008.

Rogerson, D. 1997. *Inside COM.* s.l. : M. Press, 1997.

R-project. 2009. What is R? *The R Project for Statistical Computing.* [En línea] 2009. [Citado el: 14 de mayo de 2009.] http://www.r-project.org/.

S. Pressman, Roger. 2000. *Ingeniería del Software. Un enfoque práctico.* s.l. : Mc. Graw Hill, 2000.

Shaw, Mary. *"Some Patterns for Software Architecture" en Pattern .*

Shaw, Mary y Garlan, David. 1996. *Software Architecture: Perspectives on an Emerging Discipline.* s.l. : Pretince Hall, 1996.

SPSS Inc. 2009. PASW Statistics. *SPSS, Data Mining, Statistical Analysis Software, Predictive Analysis, Predictive Analytics, Decision Support Systems.* [En línea] SPSS Inc., 2009. [Citado el: 14 de mayo de 2009.] http://www.spss.com/es/statistics/?source=homepage&hpzone=tech.

Statistical Solutions Ltd. 2005. BMDP. *SOFTWARE shop / Estadística / Statistical Solutions Ltd. / BMDP.* [En línea] 2005. [Citado el: 12 de mayo de 2009.] http://www.software-shop.com/in.php?mod=ver_producto&prdID=327&tab=0.

Szyperski, C. 2000. *Components and Architecture.* 2000.

Trujillo Casañola, Yaimí. 2006. *Evaluación teórica de la adopción del enfoque de Factorías de Software en la Universidad de las Ciencias Informáticas.* Boyeros, Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2006.

Wolf, Dewayne, Perry, E. y L., Alexander. 2002. *Foundations for the study of software architecture.* s.l. : ACM SIGSOFT SOftware Engineering Notes, 2002.