Universidad de las Ciencias Informáticas

FACULTAD 6

"Paquete de componentes complejos para el módulo Diseñador de SIGDAT"

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

Autores:

Leonel Rivas Navarro

Rodolfo Salva Blanco

Tutores:

Ing. Michael Eduardo Marrero Clark

Ing. Yoander Iñiguez Bermúdez

La Habana, Junio 2013

"Año 55 de la Revolución"

"...el futuro de nuestra Patria, tiene que ser, necesariamente, un futuro de hombres de ciencia... "

Fidel

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2013.

Leonel Rivas Navarro

_____

Firma del Autor

Rodolfo Salva Blanco

_____

Firma del Autor

Ing. Michael Eduardo Marrero Clark

_____

Firma del Tutor

Ing. Yoander Iñiguez Bermúdez

_____

Firma del Tutor

Datos de contacto

Tutores:

Ing. Yoander Iñiguez Bermúdez: graduado de Ingeniero en Ciencias Informáticas en el año 2010.

Correo electrónico: yiniguez@uci.cu

Ing. Michael Eduardo Marrero Clark: graduado de Ingeniero en Ciencias Informáticas en el año 2011.

Correo electrónico: memarrero@uci.cu

Resumen

Los sistemas generadores de encuestas se han convertido en poderosas herramientas que les permiten a los usuarios diseñar encuestas de forma rápida y sencilla, las cuales son muy usadas en la sociedad con el objetivo de recoger estados de opinión de diferentes índoles. El objetivo del presente trabajo de diploma es realizar el análisis, diseño e implementación de un paquete de componentes complejos al Sistema Integral de Gestión de Datos (SIGDAT), perteneciente al Departamento de Integración de Soluciones del Centro de Tecnologías de Gestión de Datos (DATEC). Estos componentes son necesarios debido a que en la mayoría de los entornos en los que se aplica SIGDAT se requiere de un diseño más complejo.

Para la realización de estos nuevos componentes se emplea como metodología de desarrollo de software OpenUP. Las tecnologías empleadas fueron JavaScript, JSON, ExtJS (versión 3.4) y Symfony (versión 2.0). Los artefactos se generaron usando como lenguaje de modelado el UML (versión 2.0) y auxiliados por el Visual Paradigm (versión 8.0) como herramienta de Ingeniería de Software Asistida por Computadora (CASE) y por el NetBeans (versión 7.1) como Entorno Integrado de Desarrollo (IDE). Se empleó además el Apache (versión 2.2.20) como servidor web y PostgreSQL (versión 9.1) como Sistema Gestor de Base de Datos (SGBD).

Palabras claves: componentes complejos, DATEC, encuestas, SIGDAT, sistemas generadores de encuestas.

Tabla de contenido

TABLA DE CONTENIDO

Índice de figuras

INTRODUCCIÓN

El hombre a lo largo de la historia ha tenido constantemente la necesidad de almacenar y transmitir información; para realizar dicho proceso se ha apoyado en herramientas y programas que le ayuden a procesar la misma, los cuales van en aumento producto del desarrollo vertiginoso que han tenido las Tecnologías de la Información y las Comunicaciones (TICs) en el transcurso de la última década. Por tal motivo, cada día van en ascenso los volúmenes de información que son procesados, almacenados o accedidos de forma digital. Las tecnologías web se presentan entre las que han alcanzado mayor auge, destacando específicamente el desarrollo de aplicaciones web producto del papel que desempeña Internet actualmente en el quehacer diario de muchas personas. Dichas aplicaciones son del agrado de la mayoría de los usuarios ya que poseen páginas dinámicas, logrando así una mayor interacción, permiten ahorrar tiempo ya que se pueden realizar actualizaciones inmediatas, consumen pocos recursos y son multiplataforma.

Las aplicaciones web se utilizan actualmente en muchas esferas sociales, una de ellas es la relacionada con la captura de información mediante encuestas. Estas últimas son un instrumento muy importante y son empleadas habitualmente en innumerables campos de aplicación como pueden ser la política, sociología o la investigación, entre otros. Se pueden definir además como mecanismos que ayudan a tener un mejor conocimiento de la sociedad. Como resultado del avance tecnológico los métodos utilizados tradicionalmente para realizar encuestas han experimentado grandes cambios y sin dudas la forma más atractiva actualmente para llegar a los encuestados es mediante Internet. Ejemplo de esto es la existencia de varios sitios de carácter lucrativo que permiten el diseño y publicación de encuestas en línea, entre ellos se encuentran MySurvey (1), Toluna (2), SurveyHead (3) y GlobalTestMarket (4).

Cuba no ha quedado al margen de estos avances tecnológicos, las TICs constituyen un medio influyente y decisivo en una sociedad cada vez más informatizada, donde el aumento de los conocimientos, los rápidos cambios y las demandas de una educación de alto nivel se convierten en una exigencia permanente. Un ejemplo del desarrollo alcanzado lo constituye la Universidad de las Ciencias Informáticas (UCI) creada en el año 2002. La misma presenta como principal característica la vinculación de la docencia con la producción para formar profesionales capaces de fortalecer la industria

de software en Cuba. El Centro de Tecnologías de Gestión de Datos (DATEC) es uno de los Centros de Investigación de la UCI perteneciente a la Facultad 6. El mismo tiene la misión de proveer soluciones integrales, productos y servicios relacionados con las tecnologías de gestión de datos que sirven como apoyo a la toma de decisiones.

Actualmente la línea de Soluciones Integrales cuenta con numerosos proyectos de gran envergadura, entre ellos se destaca la herramienta Sistema Integral de Gestión de Datos (SIGDAT), un sistema de gestión de datos dinámico. Dicha herramienta tiene como objetivo realizar el diseño, publicación y captación de encuestas. Entre los módulos que lo componen se encuentra el diseñador; este posibilita el diseño de las plantillas de encuestas mediante componentes, a la vez que construye el modelo de datos en el que persistirá la información digitada. El diseñador cuenta con componentes de diseño y de entrada de datos; estos últimos tienen como característica que se mapean como una columna en una tabla del modelo de datos, lo que provoca que el modelo final obtenido cuente con una única tabla donde se persistirán los datos. Esta característica provoca que el sistema no se adapte a la mayoría de los entornos en que hoy son aplicadas las encuestas, donde la información es almacenada en modelos de datos relacionales. Para lograr que SIGDAT sea utilizado en un mayor número de escenarios y ampliar su capacidad operativa y comercial, las encuestas generadas requieren de un diseño que sea capaz de manejar conjuntos de datos que sólo se podrán almacenar en modelos de datos más complejos, con múltiples tablas relacionadas entre sí; objetivo que no es posible alcanzar con los componentes que hasta hoy se le han incorporado a su módulo diseñador.

Por lo anteriormente expuesto se plantea como **problema de la investigación**: ¿Cómo modelar entornos más complejos de captura de datos para ampliar el alcance de los diseños realizados con SIGDAT?

Por cuanto, el **objeto de estudio** de la presente investigación es: el proceso de diseño dinámico de formularios. Definiendo como **campo de acción:** componentes para el diseño de formularios avanzados en SIGDAT.

Con el fin de dar solución al problema de la investigación se propone como **objetivo general**: desarrollar el paquete de componentes complejos para el módulo diseñador de SIGDAT.

Para el cumplimiento del objetivo general se plantearon los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Realizar el análisis y diseño de los componentes complejos.
- Realizar la implementación y pruebas de los componentes complejos.

Para alcanzar dichos objetivos específicos se plantearon las siguientes **tareas de la investigación**:

- Análisis de las herramientas existentes para la realización de encuestas.
- Análisis de la metodología, herramientas y tecnologías para el desarrollo de los componentes
- Definición de las características y funcionalidades de los componentes a desarrollar.
- Análisis de la arquitectura de Lycan como base para el desarrollo de los componentes.
- Construir los artefactos requeridos en la ingeniería de software del sistema comprendidos en las fases de análisis y diseño.
- Implementación de las clases de las interfaces para la gestión del modelo de datos.
- Implementación de las clases de los componentes a desarrollar.
- Implementación de las clases para la edición de las opciones de configuración de los componentes.

- Integración de los componentes desarrollados en el módulo diseñador de encuestas.
- Elaboración de los casos de prueba a realizar.
- Realización de pruebas funcionales a los componentes desarrollados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se introducen las principales características y conceptos asociados a los sistemas generadores de encuestas. Se realiza un estudio acerca del estado de los mismos a nivel mundial, estudiándose además las principales características que presenta la herramienta SIGDAT; se analizan las herramientas, la metodología de desarrollo de software y las tecnologías que serán empleadas en el desarrollo de la solución.

1.1 Conceptos fundamentales

Encuesta

La Real Academia de la Lengua Española la define como una averiguación o pesquisa. Conjunto de preguntas tipificadas dirigidas a una muestra representativa, para averiguar estados de opinión o diversas cuestiones de hecho. (5)

También se puede definir como una investigación realizada sobre una muestra de sujetos representativa de un colectivo más amplio, utilizando procedimientos estandarizados de interrogación con intención de obtener mediciones cuantitativas de una gran variedad de características objetivas y subjetivas de la población. (6)

Existen al menos cuatro tipos de encuestas que permiten obtener información primaria y pueden clasificarse según el medio que se utiliza para realizar las mismas:

- **Encuestas basadas en entrevistas personales:** Consisten en entrevistas directas con cada encuestado. Tienen la ventaja de ser controladas y guiadas por el encuestador, además, se suele obtener más información que con otros tipos de encuestas.

- **Encuestas telefónicas:** Consisten en una entrevista vía telefónica con cada encuestado. A través de este medio se puede abarcar un gran número de personas en menos tiempo que la entrevista personal, sus costos suelen ser bajos y es de fácil administración.

- **Encuestas postales:** Consiste en el envío de un cuestionario a los potenciales encuestados, pedirles que lo llenen y hacer que lo remitan a la empresa o a una casilla de correo. Este tipo de encuesta está relacionada con la sinceridad con que suelen responder los encuestados al no tener la presión directa que supone la presencia del encuestador. Presenta ventajas tales como el bajo costo en relación con la encuesta personal o por teléfono y la amplia cobertura a la que se puede llegar siempre y cuando se disponga de una buena base de datos.

- **Encuestas por internet:** Este tipo de encuesta consiste en colocar un cuestionario en una página web o en enviarlo a los correos electrónicos de un panel predefinido. Presenta una amplia cobertura a la que se puede llegar, incluso a miles de encuestados en varios países y al mismo tiempo, donde se puede obtener miles de encuestas respondidas en cuestión de horas, los bajos costos, que son menores a las encuestas personales, por teléfono y postales y además puede utilizar medios audiovisuales durante la misma. (7)

Aplicaciones web

Son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. O sea, es una aplicación que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador. (6)

Características de las aplicaciones web:

Con el surgimiento y desarrollo de las aplicaciones web se han abierto las posibilidades en cuanto al acceso y uso de información desde lugares geográficamente distantes. Con los avances en esta tecnología cada vez se demandan aplicaciones más rápidas, ligeras y robustas.

En la actualidad, el acelerado crecimiento de los sistemas de comunicación han hecho de Internet una tecnología portadora de una gran variedad de servicios, entre los que se destacan el de correo electrónico, transferencia de ficheros, servicio de información, servicios web, de televisión y telefonía. También ha permitido que el acceso a estos servicios pueda realizarse desde gran variedad de dispositivos entre los que se encuentran teléfonos móviles, los PDA (Personal Digital Assistant, de sus siglas en ingles), las computadoras, entre otros.

Mediante las aplicaciones web es posible conocer un evento distante de forma rápida. Facilita el uso de recursos, los servicios y brinda accesibilidad independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura y localización geográfica. (8)

Estos avances han permitido que la internet sea usada en una amplia variedad de aplicaciones web y complejos sistemas que antes sólo eran posibles con soluciones cliente/servidor de escritorio. La mayoría de los sistemas de supervisión y control de procesos a distancias basados en internet han enfocado su diseño hacia aplicaciones web.

1.2 Sistemas generadores de encuestas

Cada día se está extendiendo más la realización de encuestas en línea, debido al desarrollo de la tecnología, el auge que ha ganado Internet y que son más fáciles de difundir de forma virtual que de la forma tradicional. El usuario puede crear las encuestas desde cero o utilizar plantillas existentes, que ayudan a agilizar el proceso de creación. Los resultados también son más fáciles de interpretar, ya que el ordenador los muestra de una forma clara y ordenada para que se puedan analizar, compartir y redactar informes.

Ventajas de las encuestas en línea:

- Inmediatez en el envío de las encuestas por correo electrónico.
- La amplia cobertura a la que se puede llegar, incluso a miles de encuestados en varios países y al mismo tiempo.

- Reducción drástica de costes de envío: manipulación, sobres y franqueo.
- Menor tiempo de respuesta y por lo tanto mayor número de respuestas.
- Anulación de costes de recepción y manipulación de respuestas. (9)
- Bajo coste. Ya no es necesario contratar encuestadores, ni utilizar papel para obtener los datos.
- Los resultados de las encuestas se obtienen en tiempo real.
- Naturaleza interactiva. Las cualidades del medio permiten incluir elementos multimedia, impensables en las encuestas tradicionales.

1.2.1 Sistemas generadores de encuestas en la actualidad

Hoy día se pueden encontrar varias herramientas cuya funcionalidad es la creación y diseño de encuestas, a continuación se brindan características referentes a la generación y elaboración de encuestas que presentan algunas de estas soluciones existentes.

InfoPath

Microsoft InfoPath es una aplicación usada para desarrollar formularios de entrada de datos basados en XML. La principal característica de InfoPath es la habilidad de poder crear y ver documentos XML con soporte para XML Schema. Puede verificar los campos del formulario para su validación, además los usuarios pueden anexar una firma digital. El usuario puede conectarse al servidor y enviar el formulario en forma XML. Cuando el usuario se conecta al servidor, la plantilla de formulario puede ser automáticamente actualizada. (10)

En InfoPath los componentes de los formularios son conocidos como controles, entre estos se pueden encontrar las listas con viñetas, numeradas y simples, los cuadros de lista de selección múltiple y los controles de fecha y hora. Los datos que se muestran en los controles se pueden filtrar para limitar el

número de elementos mostrados a un usuario. La principal ventaja que posee es que el usuario puede desarrollar los formularios sin conocimiento previo de programación, siendo una poderosa herramienta para realizar las encuestas. Pero por otro lado Microsoft InfoPath se encuentra incluido dentro del paquete del Microsoft Office; debido a esto sólo podrá usarse si se tiene instalado el sistema operativo Windows, el cual es un software propietario.

CSPro

El Sistema de Procesamiento de Censos y Encuestas CSPro (Census and Survey Processing System) es un software para entrada, edición, tabulación y difusión de datos recogidos a través de encuestas o censos. Se desarrolla en entorno Windows 98, NT 4.0, 2000, XP, o Vista pero no puede ser usado en otro sistema operativo como Linux o Mac OS.

CSPro nos permite crear, modificar y ejecutar la entrada de datos, edición por lotes y aplicaciones de tabulación en un único entorno integrado de desarrollo. Se procesa los datos en una base de casos (uno o varios cuestionarios), donde un caso puede comprender uno o varios registros de datos. Los datos se almacenan en código ASCII, archivos de texto descritos por el diccionario de datos. CSPro contiene un poderoso lenguaje común para aplicar el procedimiento de control de entrada de datos y editar las reglas. Asimismo este software posibilita la función de exportar datos entrados o base de datos a Software Estadísticos cómo SPSS, SAS o Stata o bien exportarlos en un documento de texto delimitado por tabulaciones. (11)

CSPro es un paquete de software diseñado para Microsoft Windows, que combina las características de dos paquetes anteriores de software basados en DOS. A diferencia de los anteriores sistemas basados en texto, CSPro proporciona un enfoque visual para la creación y manipulación de los datos y reduce la necesidad de la mayoría de los usuarios de tener conocimientos avanzados de programación. CSPro tiene una gran flexibilidad y su funcionalidad puede ser explotada por usuarios cualificados. Esta herramienta se ha utilizado activamente en todo el mundo desde el año 2000.

EncuestaFacil

Permite a los usuarios elaborar por sí mismos, de una forma rápida y sencilla, encuestas internas y externas que ayuden en la toma de decisiones. El usuario a la hora de realizar una nueva encuesta tiene la posibilidad de comenzar una desde cero o usar plantillas ya existentes, agilizando de esta manera el proceso de creación. Está dirigida a todo tipo de empresas, organismos públicos, profesionales de la educación, instituciones académicas, estudiantes, institutos de investigación y consultores que quieran utilizar la misma. (12) El estudio de esta herramienta estuvo enfocado en el uso de los componentes de opciones, específicamente el de los botones de opción, los cuales son un elemento importante y que se debe tener en cuenta a la hora de confeccionar encuestas.

ProProfs

ProProfs ofrece diferentes servicios para los usuarios y miembros, entre los cuales se encuentra el de crear contenidos tales como concursos, encuestas, preguntas y respuestas. Su uso de los servicios está expresamente limitado al uso no comercial, personal y privado. Cuenta con clientes de gran renombre internacional, entre ellos podemos señalar SONY, Dell, Hp, Ford, Yale y Harvard. Después de haberse autenticado, el usuario puede acceder a la opción de crear una nueva encuesta, contando con varias plantillas según el tipo de encuesta seleccionado. (13) El estudio de esta herramienta estuvo enfocado en el uso de los componentes de opciones, especificamente la tabla formada por los botones de opción, la cual constituye un posible componente a desarrollar por el equipo de desarrollo.

1.2.2 Sistema Integral de Gestión de Datos (SIGDAT)

La gestión de información para la toma de decisiones es una tarea fundamental en cualquier proceso de dirección. La forma clásica de realizar la misma es a través de documentos que capturan de manera estructurada los datos primarios. En ausencia de sistemas informáticos que faciliten y agilicen la gestión de la información, esta se convierte en un proceso muy engorroso y no aprovecha las ventajas del uso de la tecnología.

SIGDAT es concebido como producto y como activo (reutilizable) de software de la plataforma tecnológica especializada del centro. Posibilita diseñar, gestionar, recuperar y procesar encuestas,

sirviendo de apoyo para la toma de decisiones de los diferentes organismos, instituciones y empresas, elevando la oportunidad de la información. Como activo es un conjunto de componentes reutilizables en múltiples contextos de desarrollo. SIGDAT comprende los componentes de un servidor de encuestas, un diseñador, un gestor de encuestas, un visor y un módulo de seguridad. El desarrollo se realiza completamente con tecnologías de software libre y DATEC garantiza su soporte y mantenimiento constante.

El Sistema Integral de Gestión de Datos Dinámico cuenta con 4 componentes principales distribuidos en 3 módulos: Módulo Seguridad con el componente Safety, el Módulo Diseñador de Encuesta con el componente Designer y el Módulo Usuario con los componentes Survey y DataSourceManage.

Módulo Seguridad: Contiene el componente Safety con 26 funcionalidades enfocadas a garantizar la seguridad del sistema de manera general: la autenticación para el acceso a la misma, la gestión de los usuarios, roles, la membrecía, la administración de los permisos sobre los recursos y las trazas, así como el proceso de terminar o cerrar sesión.

Módulo Diseñador: Contiene el componente Designer con 49 funcionalidades que permiten garantizar los procesos de diseño y gestión de las plantillas de encuesta, lo que implica la gestión de los componentes de las plantillas de encuesta, la gestión de las variables de captación la gestión de los nomencladores y artículos de nomencladores por una parte y por otra parte, la gestión de las plantillas de encuesta y las plantillas de encuesta publicadas.

Módulo Usuario: Contiene los componentes Survey y DataSourceManage con 32 funcionalidades enfocadas al trabajo con las encuestas, lo que implica garantizar los procesos de captación de encuesta, la gestión de las encuestas, así como gestionar los orígenes de datos y los modelos de datos.

Conclusiones sobre las herramientas estudiadas

En el presente epígrafe fueron estudiados un grupo de herramientas existentes que constituyen el punto de partida para el desarrollo de los componentes a incorporar en SIGDAT. Se decidió que la herramienta

InfoPath será tomada como guía para la realización de los componentes por ser la más factible, ya que la misma cuenta con varios años de experiencia y posee una gran aceptación en la comunidad de usuarios que la emplean. Se identificaron como componentes a desarrollar la lista de selección múltiple en la cual el usuario tendrá la posibilidad de seleccionar varios elementos de un grupo, el grupo de botones de opción que brindará la posibilidad de seleccionar varias opciones con respecto a un elemento, una tabla dinámica que posibilitará la captura de grupos de datos, la misma será representada como una tabla independiente en el modelo de la base de datos de la plantilla y un subformulario que permitirá agrupar un conjunto de datos en una nueva tabla que guarde relación con la principal.

1.3 Metodología de desarrollo de software

La realización de un software significa un reto para la mayoría de los desarrolladores, pues la creación de este en el menor tiempo posible y con calidad puede ser casi imposible, si no se cuenta con algún proceso que ayude a agilizar el desarrollo del mismo. Desarrollar un buen software depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo en un determinado proyecto, es trascendental. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. (14)

1.3.1 Metodología seleccionada por el grupo de arquitectura del departamento

Proceso Unificado Abierto (Open Unified Process, OpenUP)

OpenUp/Basic es un proceso de desarrollo de software de código abierto diseñado para pequeños equipos organizados que quieren tomar una aproximación ágil del desarrollo. Es iterativo, mínimo, completo y extensible; además en el mismo se valora la colaboración y el aporte de los stakeholders[1] sobre los entregables y la formalidad innecesarios. (15)

[1] Personas u organizaciones que están activamente implicadas en el negocio ya sea porque participan en él o porque sus intereses se ven afectados con los resultados del proyecto.

OpenUp/Basic se caracteriza por cuatro principios básicos que se soportan mutuamente:

- Colaboración para alinear los intereses y un entendimiento compartido.
- Balance para confrontar las prioridades (necesidades y costos técnicos) para maximizar el valor para los stakeholders.
- Enfoque en articular la arquitectura para facilitar la colaboración técnica, reducir los riesgos y minimizar excesos y trabajo extra.
- Evolución continua para reducir riesgos, demostrar resultados y obtener retroalimentación de los clientes. (15)

Características generales:

- Preserva la esencia del Unified Process.
- Desarrollo iterativo e incremental.
- Desarrollo dirigido por Casos de Uso.
- Centrado en la Arquitectura. (15)

La mayor fortaleza de OpenUp/Basic es que puede ser extendido en grandes o pequeñas formas para adicionar nuevos contenidos de desarrollo o personalizar el proceso para su entorno específico. Los practicantes de desarrollo de software pueden encontrar guías sobre lo que se requiere de ellos en los roles definidos por OpenUp/Basic. Cada rol describe un conjunto de actividades y artefactos de los cuales el rol es responsable. También se dan guías sobre cómo estos roles colaboran. Los ingenieros de procesos de software pueden extender y modificar OpenUp/Basic. Las modificaciones pueden ser tan

simples como alterar las plantillas para los productos de trabajo o tan sofisticadas como adicionar actividades necesarias para crear software en su ambiente específico. (15)

Ventajas del uso de OpenUp:

- Ya que es apropiado para proyectos pequeños y de bajos recursos permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.

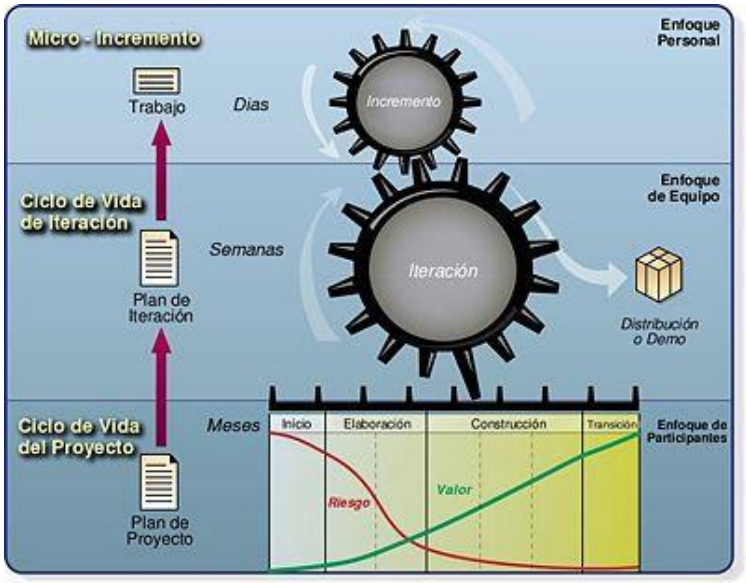


Figura 1: Ciclo de vida de OpenUp

La metodología OpenUP posee cuatro fases dentro de su ciclo de vida, las mismas se describen a continuación:

1. Concepción

Primera de las cuatro fases del ciclo de vida de un proyecto, acerca del entendimiento del propósito y objetivos y obteniendo suficiente información para confirmar que el proyecto debe hacer. El objetivo de ésta fase es capturar las necesidades de los stakeholders en el ciclo de vida del proyecto.

2. Elaboración

Es la segunda de las cuatro fases del ciclo de vida del OpenUP donde se tratan los riesgos significativos por la arquitectura. El propósito de esta fase es establecer la base de la elaboración de la arquitectura del sistema.

3. Construcción

Esta fase está enfocada al diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito de esta fase es completar el desarrollo del sistema basado en la Arquitectura definida.

4. Transición

En esta última fase de transición, a juicio del autor, el propósito es asegurar que el sistema es entregado a los usuarios, además evalúa la funcionalidad del último momento de la fase de construcción.

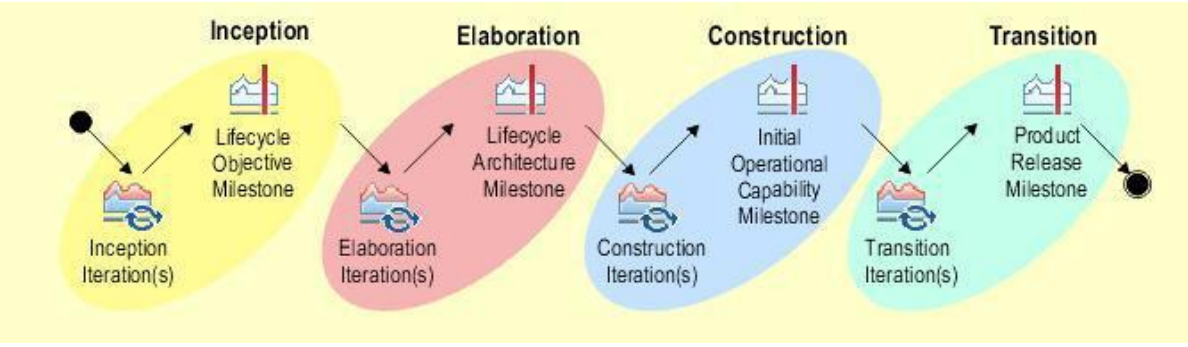


Figura 2: Fases de OpenUp

1.3.2 Justificación de la metodología seleccionada

OpenUp/Basic es la metodología de desarrollo de software que más se ajusta a las condiciones y características del proyecto debido a varias razones:

- Preserva la esencia de RUP que es la metodología más conocida por los estudiantes y profesores de la UCI debido a que fue la estudiada en los cursos de Ingeniería de Software I y II, asignatura obligatoria de la carrera.

- Es extensible y adaptable a las características del equipo de desarrollo, lo que permite modificarla teniendo en cuenta que es necesario realizar cambios a estas metodologías para que además de satisfacer las necesidades en cuanto a simplificación de tiempo, documentación y actividades, cumpla con los lineamientos de calidad y con las metas trazadas por la universidad en cuanto al nivel de madurez de la organización para el proceso de desarrollo de software.

- Modelo de desarrollo Iterativo Incremental al Igual que RUP, pero con la diferencia de que al ser un proceso mucho más ligero, permite micro incrementos en breves periodos de tiempo, lo que posibilita ir creando versiones del producto mientras se desarrolla y efectuar modificaciones a los requerimientos sin altos costos en cuanto a tiempo, precio e implementación.

1.4 Tecnologías y herramientas

Para el desarrollo de los componentes se hace necesario el estudio de las tecnologías y herramientas definidas por el grupo de arquitectura del departamento de Soluciones Integrales de DATEC. La selección de este ambiente de desarrollo se ajusta al Acuerdo No. 084 del 2004 del Comité Ejecutivo del Consejo de Ministros, donde se identificó la necesidad de ejecutar acciones que garantizarán la migración ordenada y progresiva hacia aplicaciones y plataformas de código abierto, en concordancia con el desarrollo del proceso de informatización de la sociedad como parte de la ejecución por el país de una política orientada a alcanzar la seguridad, invulnerabilidad e independencia tecnológica. (16)

1.4.1 Lenguaje de Modelado

El lenguaje de modelado es un conjunto estándar de símbolos y de formas que facilita la modelación de un diseño de software.

Lenguaje Unificado de Modelado (Unified Modeling Language, UML 2.0)

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y además, cuenta con reglas para combinar dichos elementos. Permite el modelado de sistemas con tecnología orientada a objetos, pero es importante aclarar que no es un proceso o una guía para realizar el análisis y diseño orientado a objetos. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

1.4.2 Tecnologías web

Un punto importante en el desarrollo de cualquier software es la selección correcta de las tecnologías a utilizar, pues con ello se estaría asegurando a largo plazo una mejor producción, una disminución del tiempo de trabajo y un producto final con calidad. Para dicha selección es importante tener en cuenta puntos como la disponibilidad de las herramientas, sus licencias y grado de dificultad a la hora del equipo de trabajo efectuar la interacción con estas.

JavaScript

Es un lenguaje de programación interpretado, o sea que no requiere de compilación, dialecto del estándar *ECMAScript*. Es pequeño y ligero; no es útil como un lenguaje independiente, más bien está diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores web; por tal motivo es muy utilizado en la realización de páginas web dinámicas. No tiene que declarar todas las variables, clases y métodos, no debe preocuparse si estos últimos son públicos, privados o protegidos y no tiene que implementar sus interfaces. Los tipos de variables, parámetros y funciones de

retorno no son explícitamente definidos. Se ejecuta al lado del cliente, lo que evita la sobrecarga del servidor por las peticiones concurrentes que pueden tener respuestas localmente. La verdadera fuerza de JavaScript es la compatibilidad con los distintos navegadores; además de que es interpretado por todos los navegadores modernos.

JavaScript Object Notation (JSON)

JSON es una notación muy simple para definir objetos en JavaScript, 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. El formato JSON permite representar estructuras de datos y objetos (arrays asociativos) en forma de texto. La notación de objetos mediante JSON es una de las características principales de Javascript y es un mecanismo definido en los fundamentos básicos del lenguaje.

Hypertext Preprocessor (PHP 5.3.10)

Es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML. Lenguaje de programación interpretado, diseñado originalmente para la creación de página web dinámicas. Es usado principalmente en interpretación del lado del servidor, pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Presenta interfaces para una gran cantidad de sistemas de base de datos diferentes tales como: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otros, además de que dispone de conexión propia para estos sistemas. Entre sus ventajas se encuentra la facilidad de aprendizaje y uso que posee, portabilidad, bajo coste y las bibliotecas incorporadas para el trabajo en la web. PHP permite técnicas de programación orientada a objetos y es un lenguaje multiplataforma. Posee documentación en su página oficial la cual incluye la descripción y ejemplos de cada una de sus funciones. (17)

1.4.3 Servidor Web

Un servidor web es un software en un servidor que se encuentra en constante procesamiento, al mantenerse en espera de solicitudes de ordenadores clientes a través de navegadores web y dándole repuestas a estas mediante páginas web que se muestran en dicho navegador. Las peticiones son realizadas con el uso del protocolo HTTP y respondidas con códigos HTML que el cliente es el encargado de descifrar en fuentes, colores y formas exhibidas en dichas páginas.

Apache 2.2.20

El Servidor Apache forma parte de los software libre, su diseño le permite ser un servidor web potente y flexible, funcional en una amplia gama de plataformas y entornos. Permite ser adaptado a diferentes necesidades y es extensible debido a que al ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor. Este servidor es funcional tanto para páginas estáticas como para páginas dinámicas a través de otras herramientas soportadas que facilitan la actualización de los contenidos usando bases de datos, ficheros u otras fuentes de información.

1.4.4 Herramienta CASE

Se puede definir herramienta CASE (*Computer Aided Software Engineering*) a la aplicación de métodos y técnicas a través de las cuales las personas pueden modelar o diseñar sistemas por medio de programas, procedimientos y su respectiva documentación. Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (18)

Visual Paradigm for UML 8.0

Visual Paradigm para UML (Lenguaje Unificado de Modelado) es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta un entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de casos de usos. Soporta un conjunto de lenguajes, tanto en la generación de código como en la ingeniería inversa tales como: Java, C++, CORBA IDL, PHP, XML Schema y ADA, además puede generar código a partir de los modelos y viceversa. Cualquiera de los cambios que se realicen en el código existente puede reflejarse en el modelo. Esta herramienta visual permite construir la aplicación con mayor rapidez, mayor exactitud, mejor trabajo en equipo y fácil de utilizar, además de que aumenta las expectativas mediante la interfaz gráfica.

1.4.5 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (*Integrated Development Environment* o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Hoy en día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python, Visual Basic, entre otros). (19)

NetBeans IDE 7.1

El NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Es un producto libre y gratuito sin restricciones de uso. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma pueden ser extendidas fácilmente por otros desarrolladores de

software. Las características que presenta en el NetBeans como IDE open source[2] y multiplataforma lo convierten en uno de los más potentes actualmente. El mismo cuenta con mejoras que permite el desarrollo en PHP, JavaScript, CSS, HTML, XHTML, Java entre otras tecnologías. Este IDE permite la interacción con las librerías de ExtJS y la generación automática de códigos para los lenguajes que soporta, por lo que constituye una excelente herramienta para la implementación.

1.4.6 Marcos de trabajo

Los marcos de trabajo o framework de desarrollo web son muy utilizados, debido a que representan una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Es decir, representa una aplicación incompleta que es configurable, a la que se le puede añadir piezas concretas para la construcción de una aplicación con nuevas características. Los frameworks desde su creación han sido muy utilizados por los programadores y diseñadores porque los proveen de una mejor estructura y organización en sus proyectos. Son muy usados para simplificar y agilizar el proceso de desarrollo de aplicaciones. Proporcionan un conjunto de librerías con funcionalidades que aumentan la facilidad del trabajo y disminuyen su complejidad.

ExtJS 3.4

ExtJS es una biblioteca JavaScript ligera y de alto rendimiento completamente orientada a objetos compatible con la mayoría de los navegadores, que permite crear aplicaciones enriquecidas del lado del cliente haciendo un uso extensivo de las tecnologías AJAX, XHTML/ DHTML y DOM. Su potencia reside en la colección de componentes para el diseño de interfaces de usuario (GUIs). Brinda múltiples posibilidades para el trabajo con las validaciones, manejo de errores en el cliente y permite la personalización de temas de estilos. Basa toda su funcionalidad en JavaScript a través de librerías YUI, jQuery, o haciendo uso de la librería nativa, así en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso de DOM. En el momento que comenzó a emplearse este marco de

[2] Código abierto (en idioma inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente.

trabajo no existía mucha documentación en español, en la actualidad cuenta una comunidad bastante grande, y cada una de las versiones que se lanzan al mercado son mejores que las anteriores, además que son distribuidas bajo licencias open Source.

Symfony 2.0

Symfony 2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto. (20) Es un completo marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web utilizando el Modelo Vista Controlador como patrón de arquitectura web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Utiliza Doctrine como Mapeo Objeto-Relacional (ORM) para generar la capa de acceso a datos. Se puede ejecutar tanto en plataformas Unix (Linux, etc.) como en plataformas Windows.

1.4.7 Sistema Gestor de Base de Datos (SGBD)

Un Sistema Gestor de Base de Datos es un tipo de software que funciona como interfaz entre el usuario, la aplicación y la base de datos. Están compuestos por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta. Tiene como propósito el manejo sencillo de los datos y las entidades que los organizan con el fin de llegar más fácilmente a la información contenida dentro de la base de datos. Con la utilización del gestor de base de datos se logra: independencia de los datos y los programas de aplicación, minimización de la redundancia, integración y sincronización de las bases de datos, integridad, seguridad y protección de los datos, facilidad de manipulación de la información y control centralizado.

PostgreSQL 9.1

Es un Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos y de código abierto, brinda un control de concurrencia multi-versión (MVCC por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación. Posee características significativas del motor de datos, entre las que se pueden incluir las subconsultas, los valores por defecto, las restricciones a valores en los campos (constraints) y los disparadores (triggers). (6) Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos. Soporta bloques de código que se ejecutan en el servidor, los cuales pueden ser escritos en diferentes lenguajes de programación con la potencia que presenta cada uno, desde sus operaciones básicas tales como la bifurcación y bucles hasta las complejidades de la programación orientada a objetos o la programación funcional.

Conclusiones del capítulo

Como resultado del análisis realizado durante el presente capítulo, se identificaron las principales características de las herramientas y sistemas para la generación de encuestas analizados, así como elementos significativos de la generación de formularios a tener en cuenta en el desarrollo del sistema propuesto. Luego del estudio realizado de las herramientas y el análisis de la arquitectura definida por el Departamento de Soluciones Integrales y el arquitecto del proyecto, se seleccionó para dar solución a la problemática planteada la metodología de desarrollo OpenUP, el lenguaje de modelado visual UML 2.0, los lenguajes de programación PHP 5.3.10, JavaScript y JSON, Apache 2.2.20 como servidor web, la herramienta CASE Visual Paradigm 8.0, NetBeans 7.1 como entorno de desarrollo integrado para la implementación, como framework de desarrollo ExtJS 3.4 y Symfony 2.0 y además se seleccionó a PostgreSQL 9.1 como sistema gestor de base de datos.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se describen, a partir del modelo de dominio, los principales conceptos del entorno que serán objeto de análisis para la realización de la fase de Análisis y Diseño de los componentes propuestos. Se elabora una lista de los requisitos funcionales que deben poseer los componentes y otra de los requisitos no funcionales que debe presentar el sistema de cómputo donde se vaya a instalar la aplicación, los mismos se deben tener en cuenta para la implementación de los componentes. Se identifican los actores, casos de uso y las relaciones existentes entre ellos.

2.1 Modelo de dominio del sistema

Un modelo del domino captura los tipos más importantes de objetos en el contexto del sistema. (21) Entre los principales objetivos de este se encuentra la comprensión y descripción de las clases más importantes dentro del sistema.

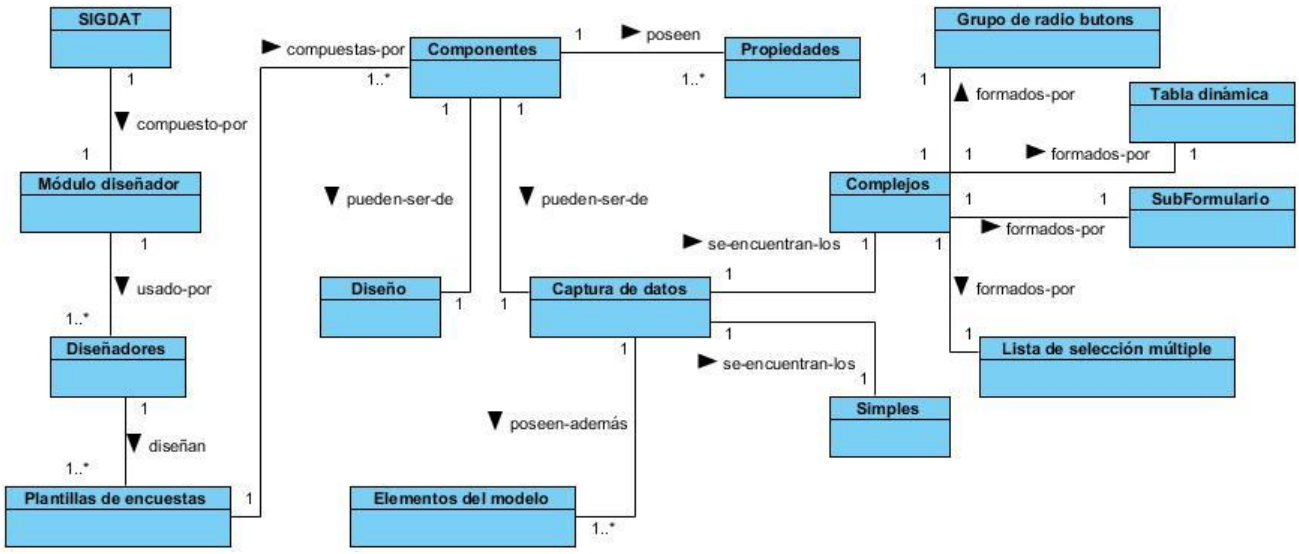


Figura 3: Diagrama del modelo de dominio

Definición de clases del modelo de dominio

SIGDAT: Sistema integral para la gestión de datos, el cual posibilita diseñar, gestionar, recuperar y procesar encuestas.

Módulo diseñador: Módulo que contiene al componente Designer con 49 funcionalidades que permiten garantizar los procesos de diseño y gestión de las plantillas de encuesta.

Diseñadores: Personas encargadas de realizar todo el proceso referente al diseño y gestión de las plantillas de encuestas.

Plantillas de encuestas: Encuestas que son diseñadas y gestionadas por los diseñadores, haciendo uso de la herramienta SIGDAT, las cuales podrán utilizarse en múltiples esferas sociales.

Componentes: Elementos que conforman o pueden conformar a las plantillas de encuestas. Los mismos pueden ser de diseño o de captura de datos.

Propiedades: Atributos de configuración que poseen los componentes de las plantillas de encuestas.

Diseño: Representa las configuraciones de todos los componentes presentes en el diseño de las plantillas de encuestas.

Captura de datos: Representa a los componentes asociados con la captura de datos en las plantillas de encuestas.

Elementos del modelo: Atributo de configuración que poseen los componentes de captura de datos de las plantillas de encuestas.

Simples: Componentes de captura de datos que están presentes en SIGDAT que se mapean como una columna en la base de datos donde es publicada la plantilla.

Complejos: Componentes de captura de datos que están presentes en SIGDAT que se mapean como una tabla en la base de datos donde es publicada la plantilla.

2.2 Especificación de los requisitos del sistema

Los requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento, no son más que propiedades o restricciones determinadas de forma precisa que deben satisfacerse. (22)

2.2.1 Requisitos Funcionales:

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Además, se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los requisitos funcionales definen las funciones que el sistema será capaz de realizar. (23)

RF 1: Adicionar componente

> **Descripción:** El objetivo del requisito consiste en adicionar un nuevo componente en la plantilla de encuesta.

RF 2: Editar componente

> **Entrada:** Atributos de configuración del componente seleccionado.
>
> **Salida:** Modificación de los atributos de configuración del componente.
>
> **Descripción:** El objetivo del requisito consiste en editar las propiedades de un componente de la plantilla de encuestas.

RF 3: Eliminar componente

Descripción: El objetivo del requisito consiste en eliminar uno o varios componentes en la plantilla de encuesta.

RF 4: Editar modelo de datos

Entrada: tipo de dato, alias, etiqueta.

Salida: Modificación de los atributos tipo de datos, alias y etiqueta.

Descripción: El objetivo del requisito consiste en modificar el modelo de datos de un componente de captura de datos.

RF 5: Adicionar columna

Entrada: nombre.

Salida: Nueva columna.

Descripción: El objetivo del requisito consiste en adicionar una nueva columna en el modelo de datos del componente seleccionado.

RF 6: Editar columna

Entrada: nombre.

Salida: Modificación del atributo nombre.

Descripción: El objetivo del requisito consiste en modificar una columna del modelo de datos.

RF 7: Eliminar columna

Descripción: El objetivo del requisito consiste en eliminar una columna del modelo de datos.

RF 8: Adicionar nomenclador

Entrada: nombre, campo a mostrar, campo del valor, tabla.

Salida: Nuevo nomenclador.

Descripción: El objetivo del requisito consiste en adicionar un nuevo nomenclador en el panel de los nomencladores.

RF 9: Editar nomenclador

Entrada: nombre, campo a mostrar, campo del valor, tabla.

Salida: Modificación de los atributos nombre, campo a mostrar, campo del valor y tabla.

Descripción: El objetivo del requisito consiste en editar un nomenclador del panel de nomencladores.

RF 10: Eliminar nomenclador

Descripción: El objetivo del requisito consiste en eliminar un nomenclador del panel de nomencladores.

RF 11: Adicionar artículo

Entrada: nombre, valor.

Salida: Nuevo artículo.

Descripción*:* El objetivo del requisito consiste en adicionarle un nuevo artículo al nomenclador seleccionado.

RF 12: Editar artículo

Entrada: nombre, valor.

Salida: Modificación de los atributos nombre y valor.

Descripción*:* El objetivo del requisito consiste en editar las propiedades que posee un artículo del nomenclador seleccionado.

RF 13: Listar artículos del nomenclador

Entrada: nombre, valor.

Salida: Se muestran los atributos nombre y valor que poseen los artículos.

Descripción: El objetivo del requisito consiste en listar los artículos que posee un nomenclador.

RF 14: Eliminar artículo

Descripción*:* El objetivo del requisito consiste en eliminar un artículo del nomenclador seleccionado.

RF 15: Publicar plantilla

Entrada: descripción, nombre de publicación y fuente de datos.

Salida: Modificación de los atributos descripción, nombre de publicación y fuente de datos.

Descripción: El objetivo del requisito consiste en publicar una plantilla de encuesta.

RF 16: Digitar encuesta

Descripción: El objetivo del requisito consiste en digitar una encuesta determinada.

RF 17: Guardar encuesta

Entrada: nombre.

Salida: Modificación del atributo nombre.

Descripción: El objetivo del requisito consiste en guardar la encuesta en la base de datos de SIGDAT.

RF 18: Terminar encuesta

Descripción: El objetivo del requisito consiste en guardar la encuesta en el gestor de base de datos después de haber verificado la misma.

RF 19: Editar encuesta guardada

Descripción: El objetivo del requisito consiste en editar una encuesta que se encuentre guardada.

RF 20: Editar encuesta terminada

Descripción: El objetivo del requisito consiste en editar una encuesta que se encuentre terminada.

RF 21: Eliminar encuesta guardada

Descripción: El objetivo del requisito consiste en eliminar una encuesta que se encuentre guardada.

RF 22: Eliminar encuesta terminada

Descripción: El objetivo del requisito consiste en eliminar una encuesta que se encuentre terminada.

RF 23: Visualizar encuesta guardada

Descripción: El objetivo del requisito consiste en visualizar una encuesta que se encuentre guardada.

RF 24: Visualizar encuesta terminada

Descripción: El objetivo del requisito consiste en visualizar una encuesta que se encuentre terminada.

2.2.2 Requisitos No Funcionales:

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Se refieren a las características que hacen al producto atractivo, usable, rápido o confiable. Por lo general los requisitos no funcionales son fundamentales en el éxito del producto; normalmente están vinculados a los requisitos funcionales, es decir, una vez que se conoce lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades o propiedades debe tener. Los requisitos no funcionales son importantes para que los clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (23)

Usabilidad

- RNF 1: Los componentes serán fácil de manipular por cualquier persona, incluyendo aquellos que no cuenten con mucho conocimientos informáticos. Deben ser intuitivos y tener un alto nivel de usabilidad, permitiendo que usuarios con nivel básico en la informática puedan explotarlos al máximo en aproximadamente 30 días.

Soporte

- RNF 2: Se le debe dar mantenimiento los componentes y corregir cualquier problema que surja con los mismos.

- RNF 3: Se deberá prever futuras versiones de los componentes y la actualización de las tecnologías empleadas en el desarrollo.

Restricciones de diseño

- RNF 4: Entorno integrado de desarrollo (IDE) NetBeans 7.1.

- RNF 5: Servidor de bases de datos PostgreSQL 9.1.

- RNF 6: Lenguaje de programación JavaScript.

- RNF 7: Lenguaje de modelado UML 2.0.

- RNF 8: Framework ExtJS 3.4.

- RNF 9: Visual Paradigm 8.0 como herramienta CASE.

Interfaces de usuario

- RNF 10: Los componentes cuentan con una interfaz amigable, fácil de usar, sencilla e interactiva.

- RNF 11: Los componentes se encuentran optimizados para una resolución de 1024x768.

Interfaces Hardware

- RNF 12: Para el desarrollo: PC Intel Pentium 4 o superior, CPU 2.5GHZ o superior, 1GB de RAM o superior, 160 GB HDD o superior.
- RNF 13: Para explotación del servidor: CPU Dual Core 2.5GHZ o superior, memoria RAM de 4 GB, 250 GB HDD.
- RNF 14: Para explotación del cliente: PC Pentium 4 o superior, CPU 2GHZ o superior, 512MB RAM mínimo o superior.

Interfaces Software

Un servidor en el que sólo se instale SIGDAT con los siguientes elementos:

- RNF 15: Sistema operativo GNU/Linux Ubuntu 11.4 o versión superior.
- RNF 16: Servidor de aplicaciones Apache versión 2.2.20 con PHP 5.3.10.
- RNF 17: Sistema Gestor de Base de Datos PostgreSQL, versión 9.1.
- RNF 18: En las PC clientes Mozilla Firefox 13.0 o superior.

Requisitos Legales, de Derecho de Autor y otros.

- RNF 19: La plataforma para el desarrollo de los componentes debe basarse en la licencia GPL.

2.3 Modelo de casos de uso del sistema

Se conforma por actores, casos de uso y la relación que existe entre ellos. Representa un esquema que recoge las funcionalidades del sistema que se automatizan y determina el uso que tendrá desde el punto de vista del usuario, ya que su construcción es en base a las necesidades del mismo.

2.3.1 Definición de los actores del sistema

Los actores representan un tipo de usuario del sistema. Se entiende como usuario cualquier elemento externo que interactúa con el sistema. El conjunto de casos de uso al que un actor tiene acceso define su rol global en el sistema y el alcance de su acción. Además son generalmente responsables de realizar actividades que serán automatizadas en el sistema. (24)

Actor	Descripción
Diseñador	Su responsabilidad es realizar todo el proceso referente al diseño y gestión de las plantillas de encuestas. Ejecutando acciones tales como gestionar los componentes de encuesta, editar el modelo de datos de los componentes y terminar las encuestas realizadas.
Digitador	Su responsabilidad es realizar todo el proceso referente a la digitación de las plantillas de encuestas, acción mediante la cual se procede a insertar los datos en las mismas. Ejecutando además las acciones de guardar y terminar las encuestas una vez se encuentren confeccionadas totalmente.

2.3.2 Diagrama de Casos de Uso del Sistema

Un caso de uso es una tarea que debe poder llevarse a cabo con el apoyo del sistema que se está desarrollando. Se representan mediante un óvulo y proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. En otras palabras, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

A continuación se representa el Diagrama de Casos de Uso del Sistema obtenido luego de agrupar los 24 requisitos funcionales que se identificaron para los componentes a desarrollar y aplicando los patrones de casos de uso.

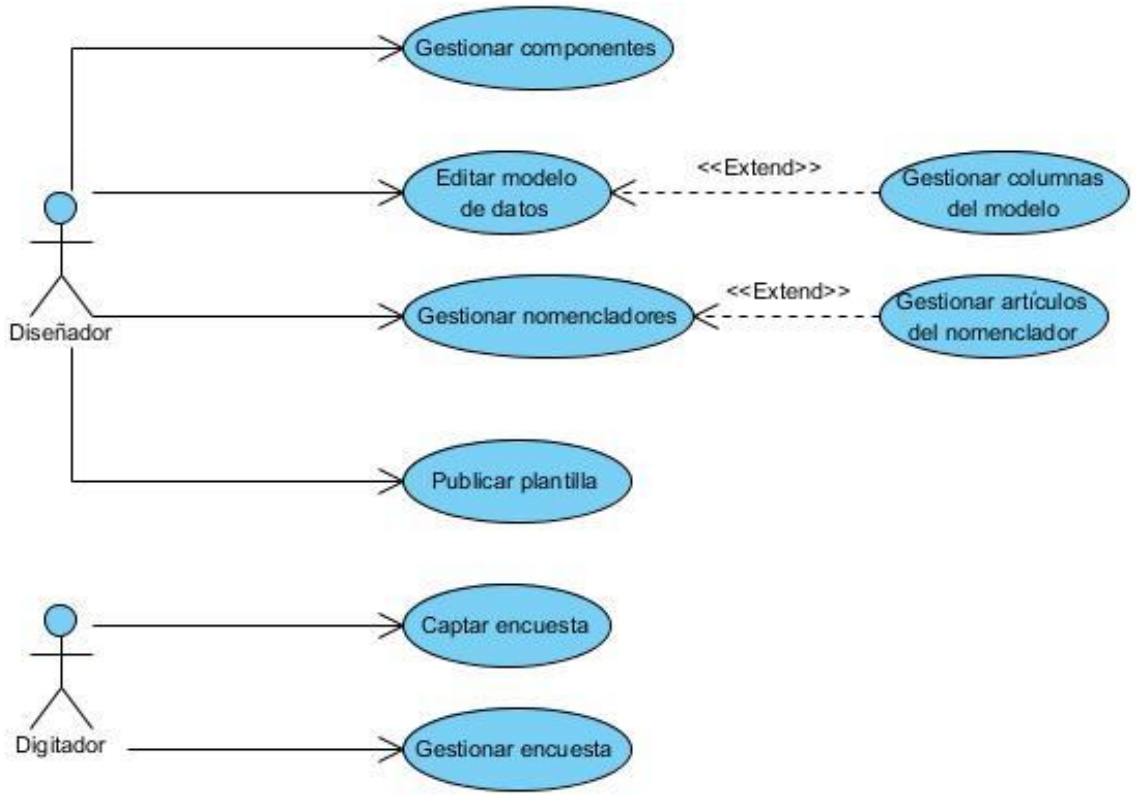


Figura 4: Diagrama de caso de uso del sistema

2.3.3 Patrones de casos de uso utilizados

Los patrones de casos de uso ayudan a identificar casos de uso a partir de las características del negocio o de los requisitos del sistema, de una forma más eficiente. Un patrón de casos de uso captura técnicas para que el modelo sea renovable, reusable y entendible. Estos patrones tienen un enfoque hacia el diseño y las técnicas son utilizadas en modelos de alta calidad. En la realización del diagrama de casos de uso del sistema se utilizaron los patrones CRUD, CRUD parcial y extensión concreta.

El patrón CRUD es definido a partir de agrupar determinados requisitos funcionales, que representan las acciones de adicionar, editar y eliminar determinada información, en este caso es evidenciado en los casos de uso gestionar componentes, gestionar columnas del modelo, gestionar nomencladores y gestionar encuesta de forma parcial; y en el caso de uso gestionar artículos del nomenclador de forma total. El patrón extensión concreta se evidencia en la posible inicialización del caso de uso gestionar columnas del modelo, a partir de que el diseñador haya seleccionado de editar el modelo de datos de un determinado componente; y en la posible inicialización del caso de uso gestionar artículos del nomenclador, a partir de que el diseñador seleccione un nomenclador.

2.3.4 Descripción del caso de uso Gestionar componentes

Caso de Uso:	**Gestionar componentes**
Actores:	**Diseñador**
Resumen:	El Caso de Uso se inicia cuando el diseñador decide realizar algunas de estas acciones. **Adicionar componente:** El diseñador selecciona un componente de la paleta de componentes y lo adiciona en el área de diseño de la plantilla.

	Editar componente: El diseñador selecciona un componente del área de diseño de la plantilla y modifica sus atributos de configuración en el panel de propiedades. **Eliminar componente:** El diseñador selecciona un componente del área de diseño de la plantilla y lo elimina.
Precondiciones:	El usuario debe estar autenticado en el sistema y debe tener permiso para gestionar componentes en la plantilla de encuesta. La plantilla debe estar previamente abierta.
Referencias	RF 1, RF 2, RF 3.
Prioridad	Crítico.

Flujo Normal de Eventos	
Sección "Adicionar componente"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador selecciona un componente de la paleta de componentes y lo arrastra hacia la plantilla en el área de diseño.	2. El sistema muestra la ventana para editar el modelo de datos del componente seleccionado.
3. Completa los datos y selecciona la	4. El sistema salva los datos del

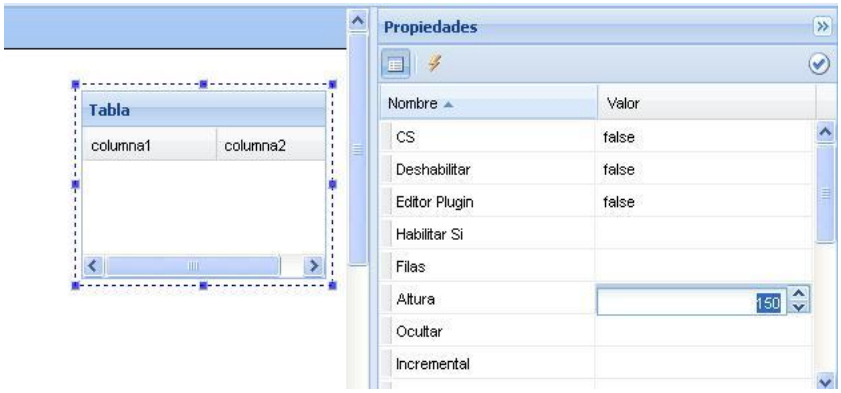
opción guardar.	componente añadido.

Prototipo de Interfaz

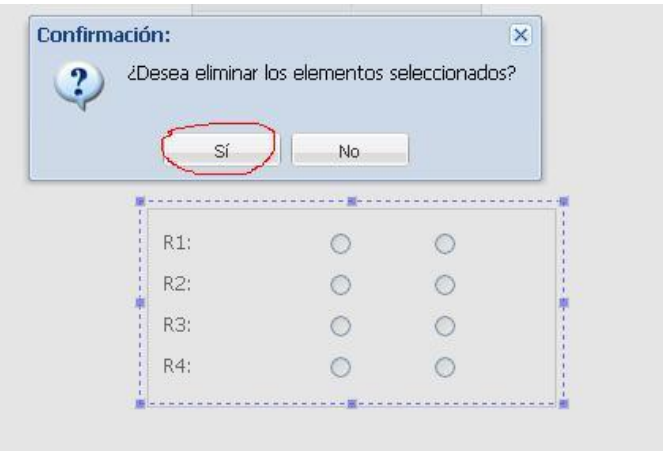
Flujos Alternos

Acción del Actor	Respuesta del Sistema
1. El diseñador selecciona un componente de la paleta de componentes y lo arrastra hacia la plantilla en el área de diseño.	2. El sistema muestra la ventana para editar el modelo de datos del componente seleccionado.
3. El diseñador presiona el botón “Cancelar” de la ventana de edición del modelo.	4. El sistema cancela la operación de insertar el componente.

Flujo Normal de Eventos

Sección "Editar componente"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador selecciona un componente en el área de diseño.	2. El sistema refresca las propiedades del componente seleccionado.
3. Edita las propiedades del componente seleccionado.	4. El sistema salva los cambios y refresca las áreas de diseño.
Prototipo de Interfaz 	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador selecciona un	2. El sistema refresca las

componente en el área de diseño.	propiedades del componente seleccionado.
3. Comienza a editar las propiedades del componente seleccionado. 4. Presiona la tecla ¨Esc¨.	5. El sistema cancela la operación de editar las propiedades del componente.
Flujo Normal de Eventos	
Sección "Eliminar componente"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador selecciona un/os componente/s en el área de diseño de la plantilla.	2. El sistema valida si el componente seleccionado puede ser o no eliminado.
3. Selecciona la opción de eliminar el/los componente/s.	4. El sistema muestra un mensaje para confirmar si el diseñador está de acuerdo en eliminar el/los componente/s.
5. El diseñador selecciona el botón "Sí" del mensaje.	6. El sistema elimina el/los componente/s. 7. El sistema refresca las áreas de

	diseño.
Prototipo de Interfaz 	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador selecciona un/os componente/s en el área de diseño de la plantilla.	2. El sistema valida si el componente seleccionado puede ser o no eliminado.
3. Selecciona la opción de eliminar el/los componente/s.	4. El sistema muestra un mensaje para confirmar si el diseñador está de acuerdo en eliminar el/los componente/s.
5. El diseñador selecciona el botón	6. El sistema cancela la operación

“No” del mensaje.	de eliminar el componente.
Poscondiciones	Componente/s adicionado o editado o eliminado/s.

2.4 Modelo de diseño

El modelo de diseño es una abstracción de la implementación del sistema. Se utiliza para concebir y para documentar el diseño del sistema de software. Es un producto de trabajo integral y compuesto que abarca todas las clases de diseño, subsistema, paquetes, colaboraciones y las relaciones entre ellos.

2.4.1 Diagramas de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces de una aplicación. Normalmente contienen información acerca de las clases, asociaciones, atributos, métodos y dependencias.

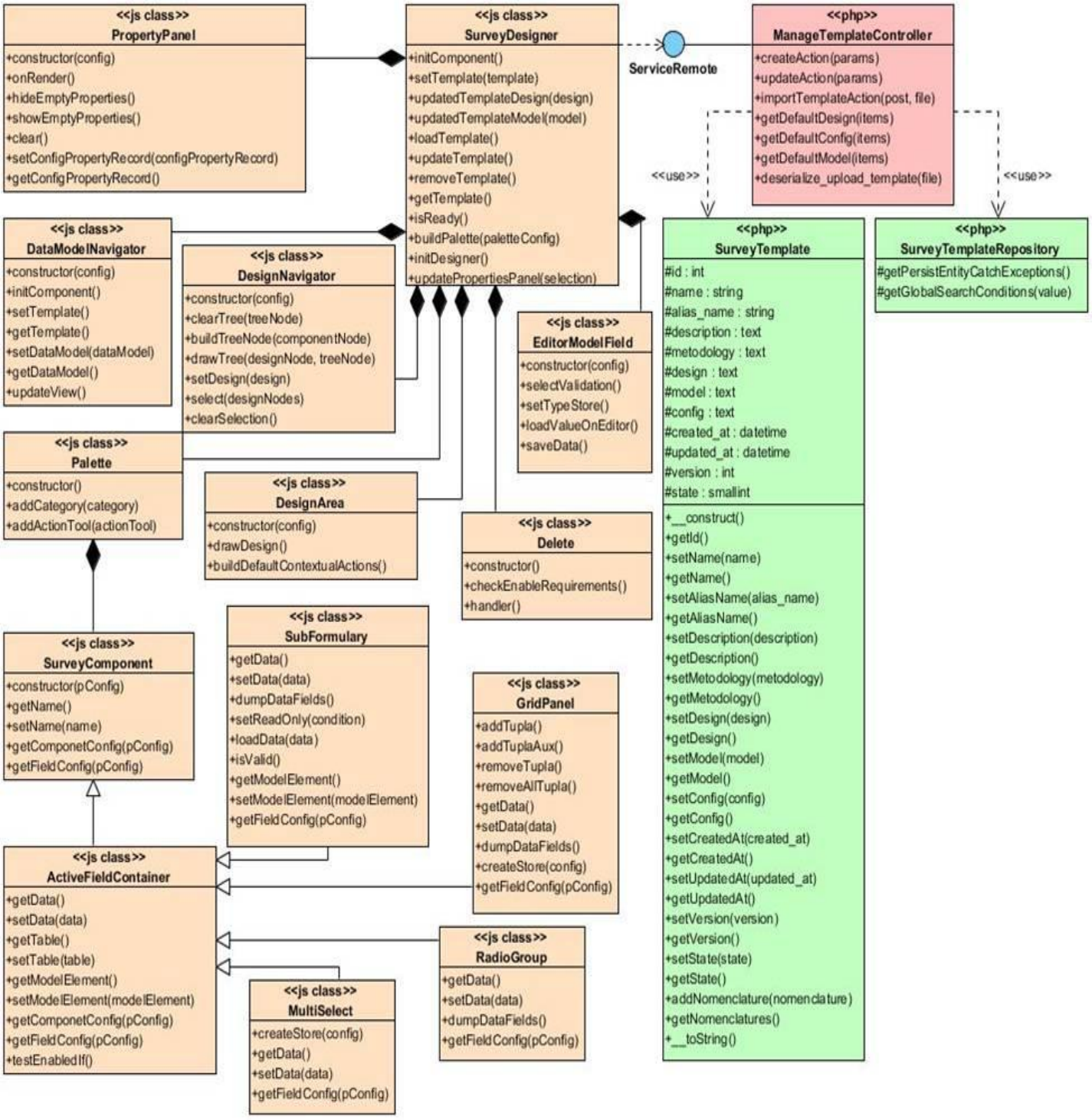

<<js class>>
PropertyPanel
+constructor(config)
+onRender()
+hideEmptyProperties()
+showEmptyProperties()
+clear()
+setConfigPropertyRecord(configPropertyRecord)
+getConfigPropertyRecord()
<<js class>>
SurveyDesigner
+initComponent()
+setTemplate(template)
+updatedTemplateDesign(design)
+updatedTemplateModel(model)
+loadTemplate()
+updateTemplate()
+removeTemplate()
+getTemplate()
+isReady()
+buildPalette(paletteConfig)
+initDesigner()
+updatePropertiesPanel(selection)
ServiceRemote
<<php>>
ManageTemplateController
+createAction(params)
+updateAction(params)
+importTemplateAction(post, file)
+getDefaultDesign(items)
+getDefaultConfig(items)
+getDefaultModel(items)
+deserialize_upload_template(file)
<<use>>
<<use>>
<<js class>>
DataModelNavigator
+constructor(config)
+initComponent()
+setTemplate()
+getTemplate()
+setDataModel(dataModel)
+getDataModel()
+updateView()
<<js class>>
DesignNavigator
+constructor(config)
+clearTree(treeNode)
+buildTreeNode(componentNode)
+drawTree(designNode, treeNode)
+setDesign(design)
+select(designNodes)
+clearSelection()
<<js class>>
EditorModelField
+constructor(config)
+selectValidation()
+setTypeStore()
+loadValueOnEditor()
+saveData()
<<php>>
SurveyTemplate
#id : int
#name : string
#alias_name : string
#description : text
#metodology : text
#design : text
#model : text
#config : text
#created_at : datetime
#updated_at : datetime
#version : int
#state : smallint
+__construct()
+getId()
+setName(name)
+getName()
+setAliasName(alias_name)
+getAliasName()
+setDescription(description)
+getDescription()
+setMetodology(metodology)
+getMetodology()
+setDesign(design)
+getDesign()
+setModel(model)
+getModel()
+setConfig(config)
+getConfig()
+setCreatedAt(created_at)
+getCreatedAt()
+setUpdatedAt(updated_at)
+getUpdatedAt()
+setVersion(version)
+getVersion()
+setState(state)
+getState()
+addNomenclature(nomenclature)
+getNomenclatures()
+__toString()
<<php>>
SurveyTemplateRepository
#getPersistEntityCatchExceptions()
#getGlobalSearchConditions(value)
<<js class>>
Palette
+constructor()
+addCategory(category)
+addActionTool(actionTool)
<<js class>>
DesignArea
+constructor(config)
+drawDesign()
+buildDefaultContextualActions()
<<js class>>
Delete
+constructor()
+checkEnableRequirements()
+handler()
<<js class>>
SurveyComponent
+constructor(pConfig)
+getName()
+setName(name)
+getComponetConfig(pConfig)
+getFieldConfig(pConfig)
<<js class>>
SubFormulary
+getData()
+setData(data)
+dumpDataFields()
+setReadOnly(condition)
+loadData(data)
+isValid()
+getModelElement()
+setModelElement(modelElement)
+getFieldConfig(pConfig)
<<js class>>
GridPanel
+addTupla()
+addTuplaAux()
+removeTupla()
+removeAllTupla()
+getData()
+setData(data)
+dumpDataFields()
+createStore(config)
+getFieldConfig(pConfig)
<<js class>>
ActiveFieldContainer
+getData()
+setData(data)
+getTable()
+setTable(table)
+getModelElement()
+setModelElement(modelElement)
+getComponetConfig(pConfig)
+getFieldConfig(pConfig)
+testEnabledIf()
<<js class>>
RadioGroup
+getData()
+setData(data)
+dumpDataFields()
+getFieldConfig(pConfig)
<<js class>>
MultiSelect
+createStore(config)
+getData()
+setData(data)
+getFieldConfig(pConfig)

Figura 5: Diagrama de clases del diseño CU Gestionar componentes

Principales clases

SurveyDesigner: Representa el marco completo de trabajo del diseñador, contiene todas las áreas de trabajo del mismo. Funciona como intermediario entre todas las interfaces.

PropertyPanel: Representa el nodo en el cual se podrán editar las propiedades de los componentes que conformen la plantilla de encuesta.

DataModelNavigator: Representa el nodo en el cual se muestran los elementos del modelo que posee un determinado componente de captura de datos que conforme la plantilla de encuesta.

DesignNavigator: Representa el árbol de los componentes presentes en el área de diseño. Permite la selección múltiple de componentes y notifica los cambios de selección.

EditorModelField: Representa la ventana que posibilita gestionar los elementos del modelo de un determinado componente de captura de datos que conforme la plantilla de encuesta.

Palette: Representa el nodo en el cual se encuentran los componentes que posee SIGDAT, los cuales son usados para la gestión de las plantillas de encuestas.

DesignArea: Representa el área en la cual se diseñan las plantillas de encuestas mediante la gestión de los componentes y del modelo de datos de los mismos.

Delete: Representa la clase encargada de eliminar los componentes del área de diseño y refrescar las áreas de trabajo.

SurveyComponent: Representa la clase padre de todos los componentes. La misma contiene los elementos de configuración que poseen los mismos de forma general.

ActiveFieldContainer: Representa al grupo de componentes que poseen un modelo de datos complejos, es decir, que se mapean en la base de datos en forma de tabla, la cual está relacionada con la tabla principal de la plantilla. Hereda de SurveyComponent, redefiniendo las funciones para obtener las opciones de configuración de los componentes, agregándole un grupo de funcionalidades adicionales que son necesarias para los componentes de este tipo.

GridPanel: Representa una tabla dinámica la cual nos posibilita relacionar los elementos del modelo de forma más rápida.

RadioGroup: Representa un grupo de botones de opciones que brindará la posibilidad de seleccionar varias opciones con respecto a un elemento.

MultiSelect: Representa una lista en la cual el usuario tendrá la posibilidad de seleccionar varios elementos de una misma columna.

SubFormulary: Representa un contenedor que permitirá agrupar un conjunto de datos en una nueva tabla que guarde relación con la principal.

ManageTemplateController: Representa la clase controladora, contiene todas las funcionalidades para la gestión de las plantillas y demás funciones que sean necesarias.

SurveyTemplate: Contiene la estructura de la tabla que se va a guardar en la base de datos y las funciones de acceso para cada uno de esos campos.

SurveyTemplateRepository: Contiene todas las funcionalidades para la gestión de datos de la entidad y otras funciones que sean necesarias.

Patrones de arquitectura

Los patrones arquitectónicos, o patrones de arquitectura, son patrones de diseño de software que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una

descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. (25)

Modelo Vista Controlador (MVC)

Symfony 2.0 como framework de desarrollo que se emplea toma lo mejor de la arquitectura MVC el cual permite un desarrollo de forma modular y un mejor soporte para futuras actualizaciones del producto, dividiendo la aplicación en tres componentes distintos: el modelo, la vista y el controlador, lo cual tiene como resultado final una aplicación muy robusta, de forma tal que, con los cambios que se realicen en una capa de la misma las demás no se vean afectadas. La capa del **modelo** define la lógica de negocio, la base de datos pertenece a esta capa. La **vista** es lo que utilizan los usuarios para interactuar con la aplicación, los gestores de plantillas pertenecen a esta capa. El **controlador** recibe las entradas, traducidas a solicitudes de servicio para el modelo. Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. (26)

Patrones de diseño utilizados

Patrones GRASP

GRASP es el acrónimo de **General Responsability Assignment Software Patterns**, que traducido al español significa: Patrones Generales de Software para Asignar Responsabilidades. Los mismos son considerados como uno de los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

- **Alta Cohesión:** Es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad,

combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos. Este patrón se encuentra presente en la clase EditorModelField, la cual se dedica solamente a la configuración del campo del modelo.

- **Controlador:** Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. El mismo se encuentra presente en la clase SurveyDesigner, la cual construye una única área y se encarga de manejar los eventos y notificaciones que se realizan de un área hacia otra. (27)

Patrones GOF

Los patrones GOF Gang-of-Four (“pandilla de los cuatro”) descritos en el libro *Design Patterns* definieron un catálogo con 23 patrones básicos. Seguidamente se describe el patrón Observer el cual se pone de manifiesto en SIGDAT.

- **Observer (Observador):** Define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes. Este patrón suele observarse en los marcos de interfaces gráficas orientados a objetos, en los que la forma de capturar los eventos es suscribir listeners[3] a los objetos que pueden disparar eventos. El mismo se encuentra presente en la clase DesignArea, la cual se encuentra constantemente a la espera del lanzamiento de algún evento para ejecutar las acciones pertinentes. (28)

[3] Son sentencias que escribimos directamente en el código y nos permiten saber cuándo ocurre un evento. En caso de que dicho evento ocurra, ejecuta una función que nosotros mismos debemos programar. Literalmente es un ‘escuchador’ que está pendiente todo el tiempo que nosotros queramos, de que se realice cierta acción.

2.4.2 Diagramas de secuencias

Los diagramas de secuencias muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela por cada sección de la descripción de los casos de uso. Contienen los detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y los mensajes entre los objetos. Está constituido por la línea de vida que representa la existencia de un objeto a lo largo de un período de tiempo. El foco de control que es el rectángulo delgado que representa el período de tiempo durante el cual el objeto ejecuta una acción. Para la realización de los casos de usos en el diseño se utilizan los diagramas de secuencia debido a que representan el flujo de acciones con mayor claridad.

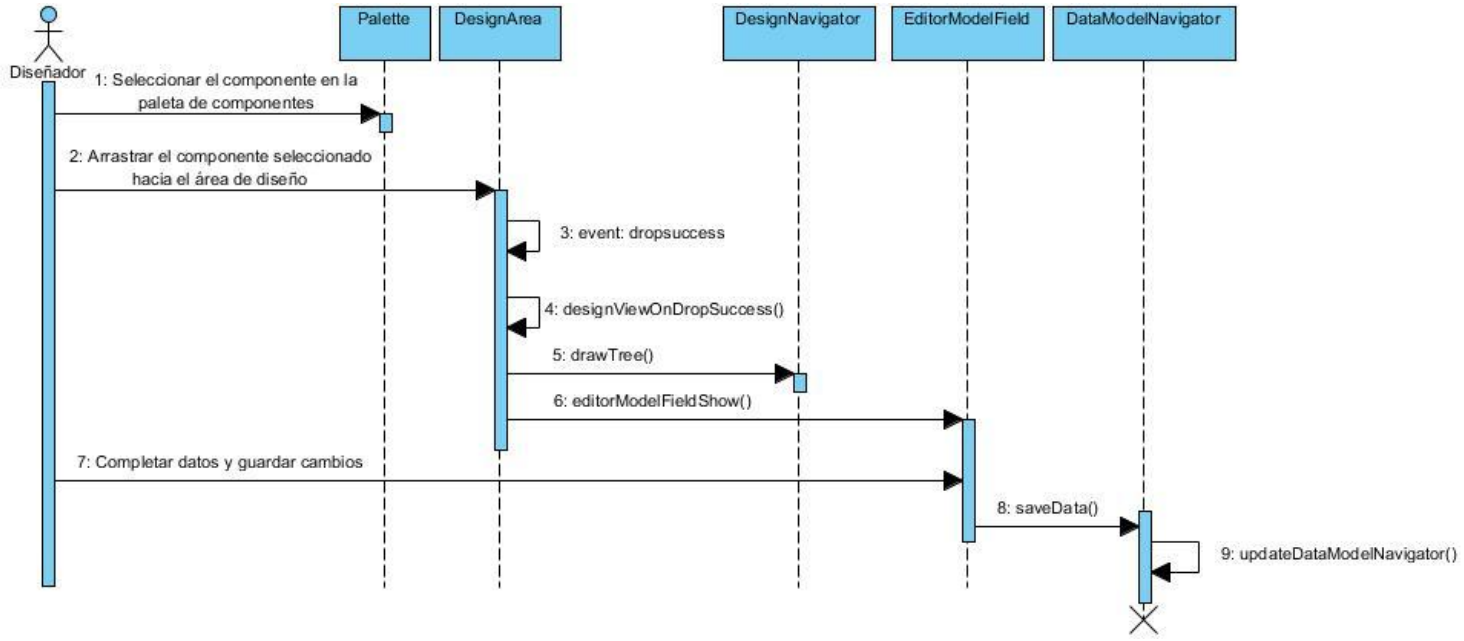


Figura 6: Diagrama de Secuencia CU Gestionar componentes Escenario: Adicionar componente

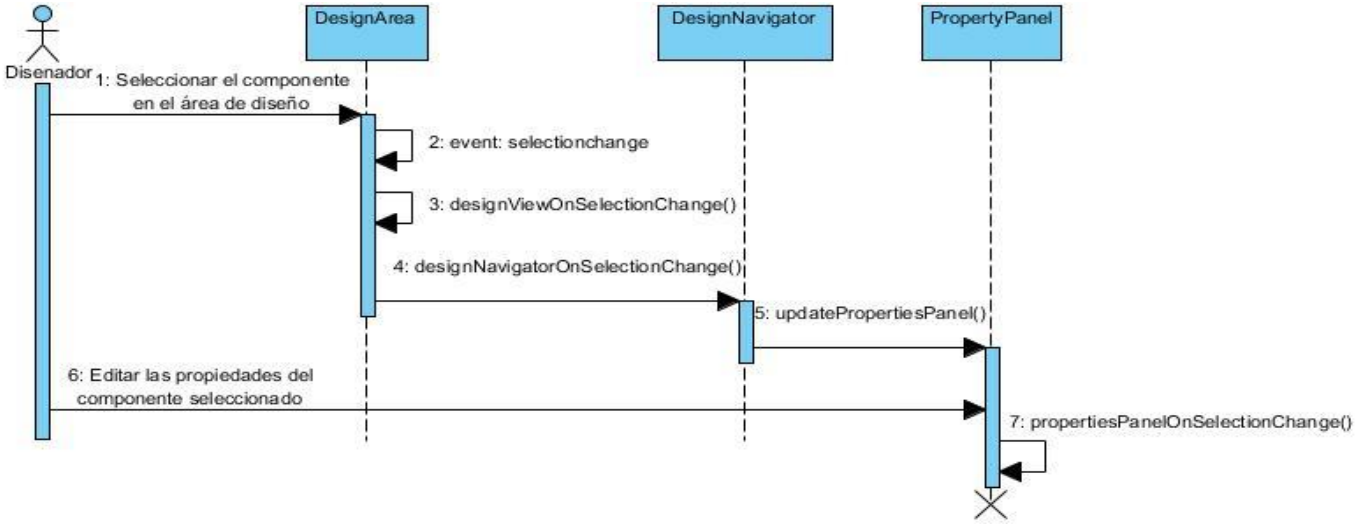


Figura 7: Diagrama de Secuencia CU Gestionar componentes Escenario: Editar componente

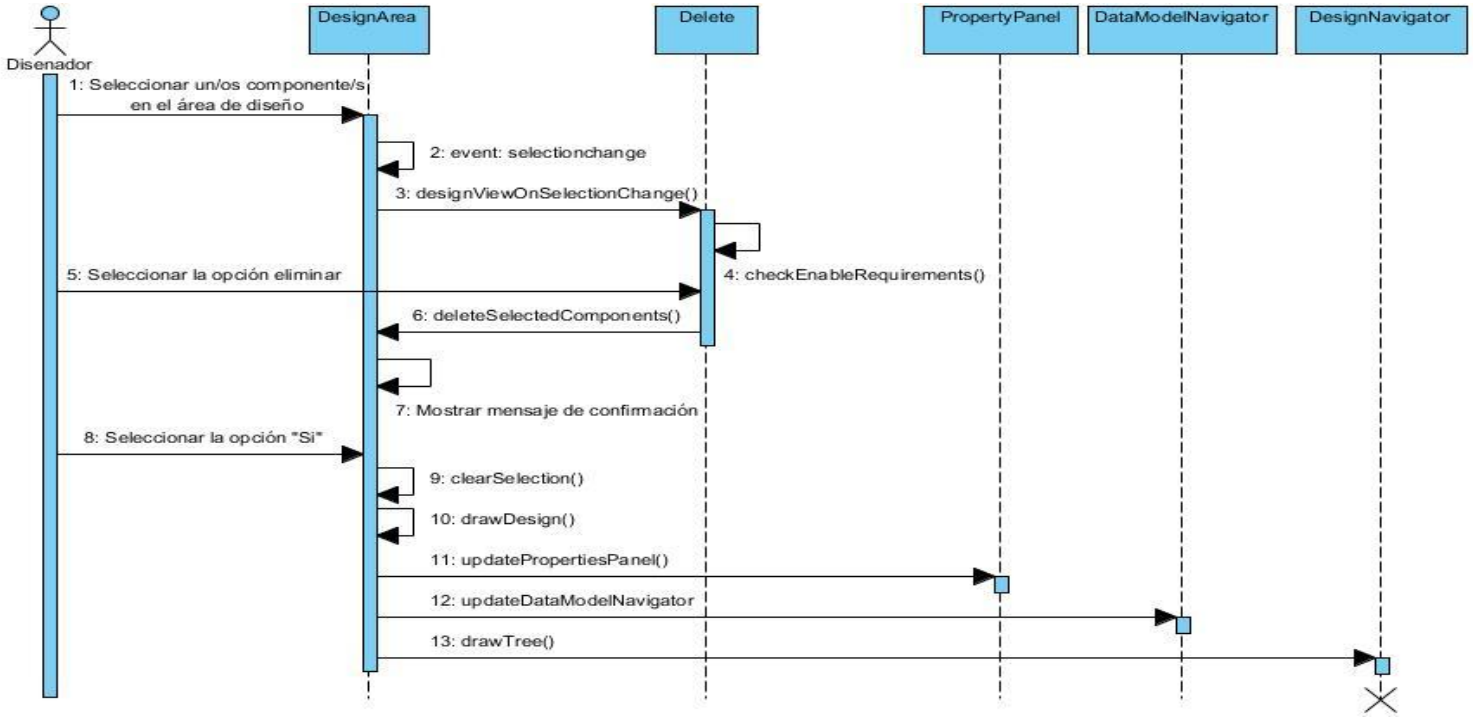


Figura 8: Diagrama de Secuencia CU Gestionar componentes Escenario: Eliminar componente

2.4.3 Diagrama de clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes representan información de larga duración o que persiste en el tiempo, es decir, es la información que se requiere almacenar para gestionarla en cualquier momento. Describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Generalmente, estas clases tienen como origen las clases clasificadas como entidad porque ellas modelan la información del sistema y el comportamiento asociado de algún fenómeno o concepto. (29)

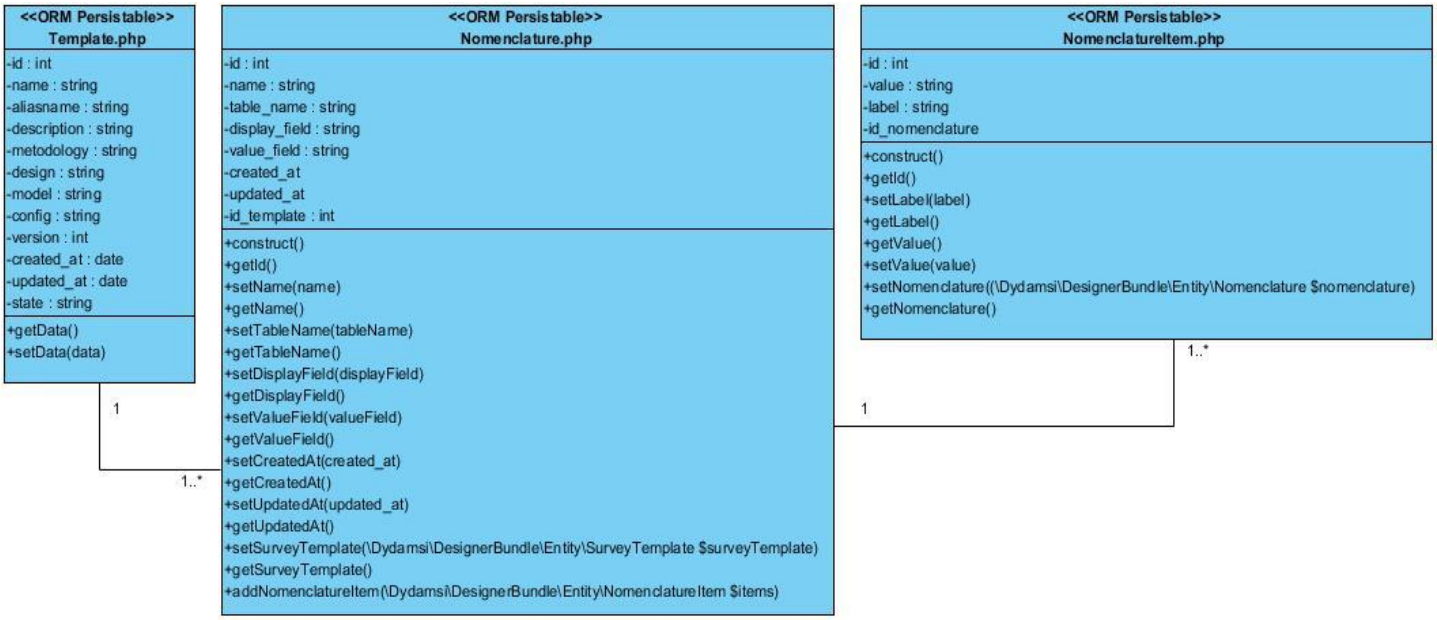


Figura 9: Diagrama de clases persistentes

2.4.4 Modelo de datos

El modelo de datos es un lenguaje utilizado para mostrar la descripción de una base de datos. Por lo general, un modelo de datos permite describir el tipo de datos que incluye la base y la forma en que se relacionan, así como las restricciones de integridad y las operaciones de manipulación de los datos. El modelo de datos describe la representación lógica y física de los datos persistentes.

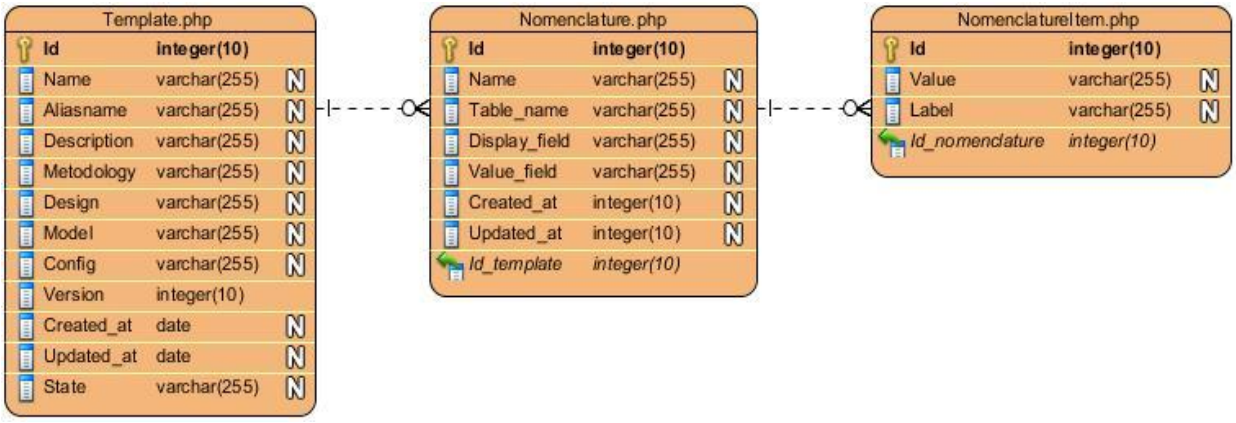


Figura 10: Modelo de datos

Descripción de las tablas

Template: Representa una instancia de la entidad plantilla en la cual se definen los campos que contienen las plantillas del sistema, declarando los nombres de cada uno de ellos y sus tipos correspondientes. Proporcionando los métodos de acceso y modificación de los datos y el CRUD completo.

Nomenclature: Representa una instancia de la entidad nomenclador en la cual se definen los campos que contienen los nomencladores de la plantilla, declarando los nombres de cada uno de ellos y sus tipos correspondientes. Proporcionando el CRUD completo.

NomenclatureItem: Representa una instancia de la entidad artículo en la cual se definen los campos que contienen los artículos de los nomencladores de la plantilla, declarando los nombres de cada uno de ellos y sus tipos correspondientes. Proporcionando el CRUD completo.

2.4.5 Modelo de despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Cada hardware es representado como un nodo; un nodo es un elemento donde se ejecutan los

componentes. El diagrama correspondiente al sistema desarrollado, ver Figura 12, consta con una computadora cliente, en la cual los usuarios pueden visualizar e interactuar con la aplicación que se encuentra en el servidor de aplicaciones. El servidor de aplicaciones es el encargado de responder las peticiones de las computadoras clientes y a su vez está conectado al servidor de base de datos, el cual es el encargado de administrar los datos necesarios para el funcionamiento correcto de la aplicación.

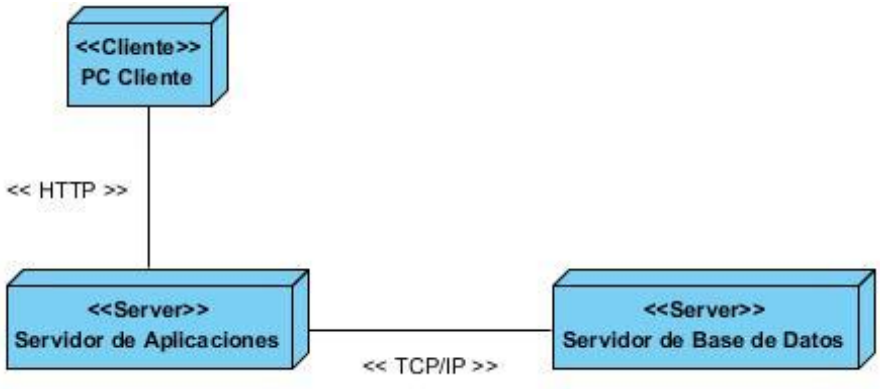


Figura 61: Modelo de despliegue SIGDAT

Conclusiones del capítulo

En este capítulo se obtuvo un mayor entendimiento del negocio, definiendo así los requisitos funcionales y no funcionales que permitieron identificar las funcionalidades que deberán cumplir los componentes a desarrollar. Los casos de uso fueron relacionados mediante un diagrama de casos de uso del sistema y se les realizó una descripción detallada logrando así un mayor acercamiento a lo que el sistema deberá cumplir. Además, en este capítulo se generaron los diagramas de clases del diseño y los diagramas de secuencia, los cuales se realizaron para cada uno de los escenarios, brindando una panorámica de cómo el usuario interactúa con el mismo. A partir del diseño de clases persistentes y la estructuración del diagrama de clases persistentes se obtuvo el modelo de datos, donde este describe la representación lógica y física de los datos persistentes y la realización del diagrama de despliegue propició una visión de cómo se encuentra distribuido físicamente el Sistema Integral de Gestión de Datos.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

Propiciar la calidad en el software es una actividad que ha surgido como consecuencia de la fuerte demanda de sistemas de software en todos los procesos que se desarrollan en la actualidad. Las pruebas representan un elemento importante para garantizar la calidad y constituyen una revisión final de las especificaciones. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. El presente capítulo se centra en el flujo de trabajo implementación para dar solución a los requisitos especificados. Se expone el artefacto diagrama de componentes, que describe los elementos físicos del sistema y sus relaciones. Además se muestran fragmentos del código fuente de la construcción de la herramienta y se validan los componentes desarrollados mediante la realización de pruebas de software.

3.1 Modelo de implementación

El Modelo de Implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, etc. Describe también cómo se organizan los componentes de acuerdo a los mecanismos de estructuración disponibles en el entorno de implementación y lenguaje o lenguajes de implementación empleados, y cómo dependen los componentes unos de otros. (29) Esta descripción resulta muy útil a la hora de implementar el sistema, ya que facilita la organización del trabajo y lo hace más entendible para los desarrolladores.

3.1.1 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, ya sean componentes de código fuente, binarios o ejecutables. Los diagramas de componentes son muy parecidos a los diagramas de casos de uso y son utilizados para modelar la vista estática de un sistema, estos representan las organizaciones y relaciones de dependencias entre los componentes del software.

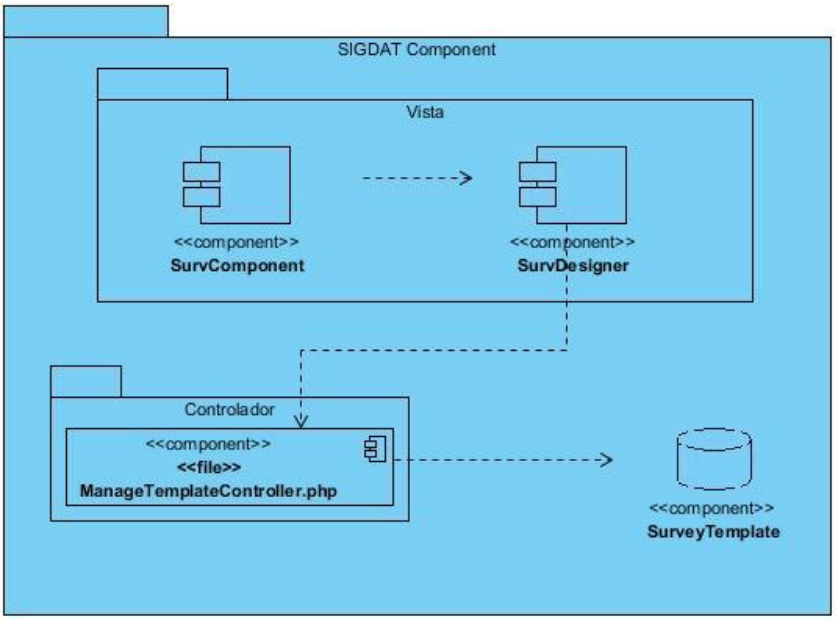


Figura 72: Diagrama de componentes

Descripción de los componentes del sistema

SurvComponent: Representa los diferentes componentes de ExtJS que fueron añadidos a la aplicación SIGDAT: GridPanel, MultiSelect, RadioGroup y Subformulary. Se incluyen además las clases de las que heredan dichos componentes.

SurvDesigner: Representa todas las interfaces que componen el área de trabajo principal: SurveyDesigner.

ManageTemplateController: Clase controladora encargada de la gestión de las plantillas de encuestas.

SurveyTemplate: Representa la estructura de la tabla que va a ser almacenada en la base de datos y las funciones para gestionar los datos. Incluye las clases SurveyTemplate y SurveyTemplateRepository.

3.2 Generación del modelo de datos a partir del diseño

La confección del modelo de datos en la herramienta SIGDAT ocurre de forma dinámica, apoyándose en los atributos **design** y **model** que posee una plantilla de encuesta. Durante el proceso de confección de la plantilla se va conformando un JSON referente a la misma, mediante el cual una vez publicada la encuesta se generan las tablas en la base de datos.

3.2.1 Proceso de construcción del modelo en el diseño

Cuando se comienza a trabajar en una nueva plantillla de encuesta, se crea una instancia de la clase Template.js y se guarda en la tabla survey_template de la base de datos local de SIGDAT. Entre las columnas que definen a esta entidad se encuentra **model**, la cual contiene un JSON en el cual se va almacenando el modelo que se comienza a formar en la plantilla a medida que se le incorporen nuevos componentes y se modifiquen los existentes. Entre los atributos que componen este JSON se encuentra **root**, que es por el cual se guiará la creación de las entidades que sean necesarias para la encuesta realizada. Este posee además otros atributos tales como **type** que puede ser de tipo 1 representando una tabla y si es de tipo 2 representa una columna, como se trata inicialmente de la plantilla general es type 1; **children** que es un arreglo donde se almacenan los JSON de las columnas de la tabla y **name** que es el nombre que va a tener la columna o la tabla en la base de datos Lo anteriormente expuesto se puede comprender con mayor claridad en la siguiente figura:

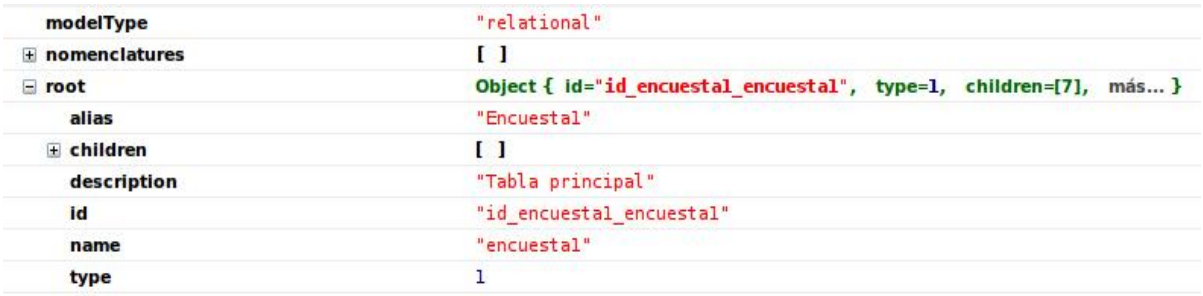


Figura 83: Ejemplo del JSON de una nueva plantilla

Luego de creada la plantilla por cada componente que posea la misma, se realiza una instancia de la clase EditorModelField.js encargada de realizar los JSON de los mismos e incluírselo como una de sus propiedades. Anteriormente esta clase solo podía generar JSON de tipo ModelColumn.js que posteriormente se les agregarían como un children al único JSON de tipo ModelTable.js que sería el de la entidad principal, esto implicaba que solamente se podía generar una única tabla por cada encuesta creada. Con los nuevos componentes incorporados, ha sido posible realizar JSON de ambos tipos, permitiendo que el modelo final obtenido contenga una mayor estructura y complejidad.

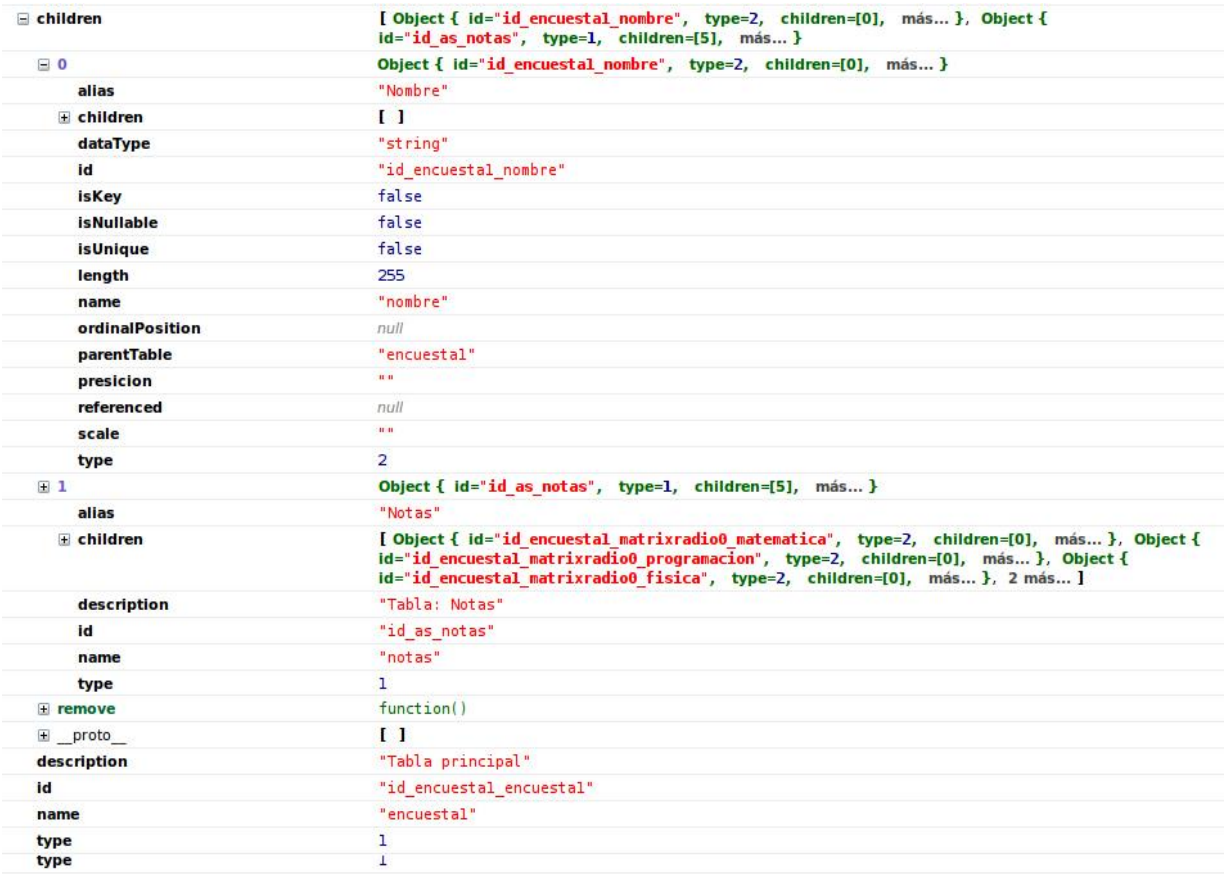


Figura 14: Ejemplo del JSON de una plantilla con componentes simples y compuestos

3.2.2 Proceso de creación del modelo físico

Luego de diseñada la plantilla de encuesta, se procede a su publicación; función de la cual se encarga la clase controladora PublishedSurveyTemplateController. Durante este proceso la clase, a partir del atributo Model de la entidad SurveyTemplate donde se almacenó el diseño del modelo en formato JSON, construye en la fuente de datos especificada por el usuario para su publicación, las tablas con sus respectivos atributos y relaciones. Para ello se utiliza una clase auxiliar denominada modelControl, encargada de recorrer el JSON del diseño del modelo y construir el esquema auxiliándose de las funcionalidades que provee la capa de abstracción de base de datos de Doctrine. (DBAL Database Abstracción Layer).

3.3 Código fuente

El código fuente es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento. Estas instrucciones son escritas en un lenguaje de programación que consiste en un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

3.3.1 Estándares de codificación

Los estándares de codificación son por lo general reglas que se deben seguir para realizar la confección del código fuente. La principal ventaja que posee es que otros programadores puedan entender el código generado. Los estándares definen buenas prácticas de programación para lograr un código robusto, lo que ayuda en la calidad del software. Muchos lenguajes de programación tiene definidas sus propias reglas de codificación. Para el desarrollo de la aplicación se tuvieron en cuenta las siguientes reglas:

Estándares definidos para JavaScript:

- Se aplica en estándar UpperCamelCase, de modo que los nombres de las clases comienzan con mayúscula y no se usan separadores entre las palabras. Ejemplo: SurveyComponent.

- Se aplica en estándar LowerCamelCase, de modo que los nombres de las variables comienzan con mayúscula excepto la primera palabra. Si tiene una sola se escribe con minúscula. Ejemplo: componentConfig, field.

- Se aplica en estándar LowerCamelCase, de modo que los nombres de los métodos comienzan con mayúscula excepto la primera palabra. Si tiene una sola se escribe con minúscula. Ejemplo: createField(), constructor().

- Cada vez que hay un nuevo bloque se diferencia por un espacio. Ejemplo:

```
getName: function() {
      return this.name;
},
```

Estándares definidos para PHP:

- Se aplica en estándar UpperCamelCase, de modo que los nombres de las clases comienzan con mayúscula y no se usan separadores entre las palabras. Si las clases son controladoras terminan en Controller. Ejemplo: ManageTemplateController.

- Se aplica en estándar LowerCamelCase, de modo que los nombres de las variables comienzan con mayúscula excepto la primera palabra. Si tiene una sola se escribe con minúscula. Ejemplo: dataSource, template.

- Se aplica en estándar LowerCamelCase, de modo que los nombres de los métodos comienzan con mayúscula excepto la primera palabra. Si tiene una sola se escribe con minúscula. Si los métodos son remotos terminan en Action. Ejemplo: getNomenclatureItemRepository(), createAction().

- Cada vez que hay un nuevo bloque se diferencia por un espacio. Ejemplo:

```
'children' => array(
                array(
```

```
                    'attributes' => array(
                        'name' => "page1",
                        'type' => "dydamsi.designer.view.comp.SurveyPage",
                        'configOptions' => new stdClass(),
                        'events' => new stdClass()
                    ),
                    'children' => array()
            )
    )
```

- Uso de comentarios para describir la funcionalidad de los métodos, los parámetros del mismo y el valor que retorna. Ejemplo:

```
/**
 * Modificar los campos del usuario.
 * @remote
 * @param array $params
 * @param $id
 * @return array
 */
```

3.3.2 Ejemplo de código fuente

```
Ext.namespace("dydamsi.designer.view.components");

dydamsi.designer.view.components.GridPanel = Ext.extend(dydamsi.designer.view.components.ActiveFieldContainer, {
    fieldClass: Ext.grid.EditorGridPanel,
    storee: null,
    cmm: null,
    listar: null,
    isListado: null,
    isIncremental: null,
    filas:null,
    getFieldConfig: function(pConfig){
        var config = dydamsi.designer.view.components.GridPanel.superclass.getFieldConfig.call(this, pConfig)
        this.listar = pConfig.listar || false;
        this.isIncremental = pConfig.incremental || false;
        this.filas = pConfig.filas || null;
        this.storee = new Ext.data.ArrayStore({
            autoDestroy: true,
            fields: eval(eval(pConfig.storee)) || [{
                name: 'd'
            }
            ]
        });

        this.cmm = new Ext.grid.ColumnModel(eval(eval(pConfig.cmm)) || [{
            header:'defecto',
            dataIndex:'0',
            editor:true
        }]);

        var sm = pConfig.sm||new Ext.grid.RowSelectionModel({
            singleSelect:false
        });
```

Figura 15: Ejemplo de código de la clase GridPanel

3.4 Pruebas de software

El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca el fallo humano son enormes. Los errores pueden empezar a darse desde el inicio del proceso, en el que los objetivos pueden estar especificados de forma errónea o imperfecta, así como dentro de pasos de diseño y desarrollo posteriores. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad. La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación en la cual el componente es ejecutado bajo ciertas condiciones específicas, los resultados observados son registrados y se realiza una evaluación del componente. (29)

Con el objetivo de lograr un producto con la menor cantidad de errores posible y que funcione de forma correcta, se decidió aplicar a nivel de pruebas las pruebas de desarrollador, ya que estas son diseñadas e implementadas por el equipo de desarrollo. Se aplicará además el nivel de pruebas de integración, asegurando que los componentes funcionen de forma correcta ya sea de forma independiente o en conjunto con el sistema. Para estos niveles de prueba en la aplicación SIGDAT se aplica la técnica de pruebas de funcionalidad, este tipo de técnicas concentra su atención en la validación de las funciones, métodos, servicios y casos de uso. Además asegura que el sistema sea utilizado solamente por los usuarios deseados. Para garantizar la calidad del software se utilizan en la aplicación las pruebas funcionales como tipos de prueba a realizar, el objetivo fundamental es probar que los componentes cumplan con las funciones específicas plasmadas en los requisitos.

Existen métodos de prueba independientemente de la técnica que se utilice o el nivel en que se enmarquen estas técnicas. Estos proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. El método seleccionado para garantizar la calidad de los componentes es el de caja negra, este se basa en la especificación del programa o componente a ser probado para elaborar los casos de prueba. Se centra principalmente en los requisitos funcionales del software. Para el desarrollo del mismo se utilizará la técnica de partición de equivalencia.

3.4.1 Diseño de casos de prueba

Se entiende por caso de prueba, según la Ingeniería del software, al conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. A partir de los diseños de casos de prueba asociados a cada caso de uso se identificaron diferentes no conformidades existentes, las cuales no propician que los componentes desarrollados funcionen correctamente, por lo que fueron revisadas continuamente durante el desarrollo de los mismos para darle solución. Para un total de 8 casos de uso se detectaron 8 no conformidades en la primera iteración de las cuales fueron resueltas satisfactoriamente, y en la segunda iteración se redujo las no conformidades llegando sólo a 5 que también fueron resueltas, número que se redujo hasta la tercera iteración donde no se obtuvieron no conformidades.

Conclusiones del capítulo

En este capítulo se definió el diagrama de componentes establecido en la aplicación, lo cual muestra una visión de cómo está distribuido físicamente el sistema. Se definió el uso de un estándar de codificación, el cual es de gran importancia para la calidad del software y para obtener un buen rendimiento. También fueron confeccionados los casos de pruebas pertinentes y se realizaron las pruebas de caja negra utilizando la técnica de la partición de equivalencia.

CONCLUSIONES GENERALES

Al término de la investigación se cumplieron todos los objetivos propuestos y se puede concluir que:

- A partir de un estudio de los principales sistemas generadores de encuestas, se identificaron propiedades a tener en cuenta para la realización de los componentes.

- Mediante la utilización de las herramientas, lenguajes y tecnologías propuestas por el grupo de arquitectura del departamento se logró correspondencia con las políticas de la Universidad y del país, en aras de lograr la independencia tecnológica.

- A partir de los requisitos funcionales identificados se diseñaron los componentes que fueron incluidos en la herramienta SIGDAT.

- Como resultado de la implementación se incorporaron nuevos componentes en la aplicación, los cuales permiten generar modelos de datos más complejos.

- La evaluación de las funcionalidades de los componentes haciendo uso del método de prueba Caja Negra, garantizó el correcto funcionamiento de los mismos.

RECOMENDACIONES

Después de haber alcanzado los objetivos que se trazaron al principio de este trabajo los autores recomiendan:

- Trabajar en mejoras en cuanto al diseño del sistema, teniendo en cuenta la opinión de los usuarios
- Se continúe perfeccionando los componentes de encuestas existentes en SIGDAT.
- Estudiar otros posibles componentes que se pudieran agregar al módulo diseñador.

REFERENCIAS BIBLIOGRÁFICAS

1. MySurvey. [En línea] http://es.mysurvey.com/.

2. Toluna. [En línea] http://es.toluna.com/.

3. SurveyHead. [En línea] https://www.surveyhead.com/.

4. GlobalTestMarket. [En línea] https://www.globaltestmarket.com/.

5. Real Academia Española. [En línea] http://www.rae.es/rae.html.

6. EcuRed. [En línea] http://www.ecured.cu/index.php/EcuRed:Enciclopedia_cubana.

7. *La encuesta estadística. Tipos de encuesta. Organización y diseño de cuestionarios. Casos prácticos.*

8. **Rodrigo.** [En línea] http://culturacion.com/2010/01/caracteristicas-de-una-aplicacion-web/.

9. EncuestaTick. [En línea] http://portaldeencuestas.com/ventajas-encuestas-online.php.

10. Office. [En línea] office.microsoft.com/es-es/infopath.

11. Blog del Centro de Investigación y Desarrollo Estadístico. [En línea] http://cides-sanmarcos.blogspot.com/2010/05/cspro-40-software-para-el-ingreso-de.html.

12. EncuestaFacil. [En línea] http://www.encuestafacil.com/.

13. ProProfs. [En línea] http://www.proprofs.com.

14. **Solis, C; Figueroa, R.** Metodologías Tradicionales vs. Metodologías Ágiles. [En línea] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515.

15. openUP/Basic. [En línea] http://epf.eclipse.org/wikis/openupsp/.

16. *RESOLUCIÓN No. 15/2011Dirección de Servicios Jurídicos Universidad de las Ciencias Informáticas2011Resolución.*

17. PHP. [En línea] http://www.php.net/.

18. Herramientas CASE. [En línea] Entorno Virtual de Aprendizaje. Material Básico de Bibliografía Básica de Ingeniería de Software.

19. netbeans.org. [En línea] http://netbeans.org/index_es.html.

20. **Eguiluz, Javier.** *Desarrollo web ágil con Symfony2 - Primera edición.* 2011.

21. [En línea] http://www.eumed.net/libros-gratis/2009c/583/Representacion del Modelo de Objetos de Dominio.htm.

22. **Agut, R, M.** *Especificación de Requisitos Software según el estándar IEEE 830. 2001.*

23. **Sommerville, I.** *Ingeniería del software. Séptima edición.* Madrid : s.n., 2005.

24. **Tello, Jesús Cáceres.** *Diagramas de Casos de Uso.* Universidad de Alcalá Departamento de Ciencias de Computación : s.n.

25. **Pressman, R. S.** *Software Engineering, a practitioner's approach 7ma ed.* s.l. : McGraw-Hill.

26. **Potencier, Fabien y François Zaninotto.** *Symfony La Guía Definitiva.* 2008.

27. **Larman, Craig.** *UML y Patrones.*

28. Entorno Virtual de Aprendizaje. [En línea] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Diseno_de_software/Patrones_de_diseno._Especificaciones.pdf.

29. **Jacobson, Ivar y Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Adison-Wesley.

BIBLIOGRAFÍA

1. **Agut, R. M.** Especificación de Requisitos Software según el estándar IEEE 830. 2001.

2. Blog del Centro de Investigación y Desarrollo Estadístico. [En línea] http://cides-sanmarcos.blogspot.com/2010/05/cspro-40-software-para-el-ingreso-de.html.

3. **Castillo, I Perez y Mesa, R Valiente.** Lims de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo de la Base de Datos del Módulo Sección de Mejoramiento de la Calidad. Ciudad de la Habana : s.n., 2009.

4. Census and Survey Processing System (CSPro).

5. **Delgado, Cecilia Milián.** *Sistema Integrado de Gestión Estadística (SIGE): Componente para la captación de encuestas económincas.* 2011. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

6. EcuRed. [En línea] http://www.ecured.cu/index.php/EcuRed:Enciclopedia_cubana.

7. **Eguiluz, Javier.** *Desarrollo web ágil con Symfony2 - Primera edición.* 2011.

8. EncuestaFacil. [En línea] http://www.encuestafacil.com/.

9. EncuestaTick. [En línea] http://portaldeencuestas.com/.

10. GlobalTestMarket. [En línea] https://www.globaltestmarket.com/.

11. Herramientas CASE. [En línea] Entorno Virtual de Aprendizaje. Material Básico de Bibliografía Básica de Ingeniería de Software.

12. **Jacobson, Ivar y Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Adison-Wesley.

13. *La encuesta estadística. Tipos de encuesta. Organización y diseño de cuestionarios. Casos prácticos.*

14. Larman, Craig. *UML y Patrones.* La Habana: Félix Varela, 2004. Tomo I y II.

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

16. **Delgado, Cecilia Milián.** *Sistema Integrado de Gestión Estadística (SIGE): Componente para la captación de encuestas económincas.* 2011. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

17. MySurvey. [En línea] http://es.mysurvey.com/.

18. netbeans.org. [En línea] http://netbeans.org/index_es.html.

19. Office. [En línea] office.microsoft.com/es-es/infopath.

20. openUP/Basic. [En línea] http://epf.eclipse.org/wikis/openupsp/.

21. Patrones de diseño. Especificaciones. [En línea] Entorno Virtual de Aprendizaje. Material Básico de Bibliografía Básica de Ingeniería de Software.

22. PHP. [En línea] http://www.php.net/.

23. **Potencier, Fabien y François Zaninotto.** *Symfony La Guía Definitiva.* 2008.

24. **Pressman, R. S.** *Software Engineering, a practitioner's approach 7ma ed.* s.l. : McGraw-Hill.

25. ProProfs. [En línea] http://www.proprofs.com.

26. Real Academia Española. [En línea] http://www.rae.es/rae.html.

27. RESOLUCION No. 15/2011Dirección de Servicios JurídicosUniversidad de las Ciencias Informáticas2011Resolución.

28. **Rodrigo.** [En línea] http://culturacion.com/2010/01/caracteristicas-de-una-aplicacion-web/.

29. **Sanchez, Linet Lores y Roque, Diana Monné.** *Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica.* Ciudad de la Habana, Junio 2009. Trabajo de Diploma para optar por el título de Ingeniero Informático.

30. **Solis, C; Figueroa, R.** Metodologías Tradicionales vs. Metodologías Ágiles. [En línea] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515.

31. **Sommerville, I.** *Ingeniería del software. Séptima edición.* Madrid : s.n., 2005.

32. SurveyHead. [En línea] https://www.surveyhead.com/.

33. **Tello, Jesús Cáceres.** *Diagramas de Casos de Uso*. Universidad de Alcalá Departamento de Ciencias de Computación: s.n.

34. Toluna. [En línea] http://es.toluna.com/.

35. **Vilches, Osvaldo Ernesto Stable y Cabrera, Yandris Mata.** *Desarrollo de un módulo para la validación de modelos estadísticos en el Paquete de Ayuda a la Toma de Decisiones y Soluciones Integrales (PATDSI).* 2010.