Universidad de las Ciencias Informáticas

"Facultad 6"

Título: "Diseñador y generador de formularios web v2.0"

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Nidia Dayana Risell Terry

Tutor(es): Ing. Alberto Mendoza Garnache

Ing. Héctor Luis Reyes Zaldívar

La Habana, Cuba

"Año 54 de la Revolución"

Junio 2012

“Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente.”

Che.

DECLARACIÓN DE AUTORÍA

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

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

Nidia Dayana Risell Terry

_____

Firma del Autor

Ing. Alberto Mendoza Garnache

_____

Firma del Tutor

Ing. Héctor Luis Reyes Zaldívar

_____

Firma del Tutor

DATOS DE CONTACTO

Tutores

Ing. Alberto Mendoza Garnache: graduado de Ingeniero en Ciencias Informáticas en el año 2009.

Correo electrónico: agarnache@uci.cu

Ing. Héctor Luis Reyes Zaldívar: graduado de Ingeniero en Ciencias Informáticas en el año 2009.

Correo electrónico: hlreyes@uci.cu

AGRADECIMIENTOS

En primer lugar agradezco a esta gran Revolución y a nuestro líder indiscutible Fidel Castro Ruz, por darnos esta oportunidad de formarnos como ingenieros informáticos de esta nueva era, en hombres y mujeres de ciencias, en hombre y mujeres comprometidos con la Patria.

A mis tutores Héctor (Tico) y Alberto, que me han apoyado y no dudaron de mí. A los profes de DATEC Frank, Dioleisys, Lobo, Yoander y todos los que ayudaron en mi formación en el centro.

A mis padres, que sin ellos no estarían donde estoy, a ellos que nunca dudaron de mi desempeño, y me enseñaron que lo que uno se proponga lo puede lograr. Ellos que siempre han estado pendiente de mí, cuidándome y haciéndome ver que si se puede.

A mis tías Ina, Yolanda, Elena, Esperanza, que son las mejores tías que tengo. Me han apoyado y guiado en cada uno de mis pasos, esperando siempre lo mejor de mí. Mis abuelos Mima y Papito les agradezco su cariño, amor, paciencia y sobre todo por ayudar a mis padres a criarme y educarme.

Agradezco de todo corazón a mi novio Lázaro por levantarme cuando ya me rendía, por hacerme ver que hay que luchar por lo que se quiere.

A todas mis amistades, que siempre se han preocupado por mí, Adriana, Marylien (la rubia loca), Anabel, Daimaris, Ismaris y Mairela (Musa), gracias por quererme como amiga y hacerme ver que no estoy sola.

A la familia Díaz Villarpando por darme lo mejor de ellos como personas, e inspirarme a seguir adelante.

DEDICATORIA

Por ser la luz que ilumina mis pasos, la razón de mi existir y la cúspide de este camino que estoy transitando, a ti mami querida, dedico este trabajo.

A mi brújula personal, que siempre me guía hacia adelante, por los mejores caminos, a ti papá por hacerme ver el orgullo que sientes por tu hijita, de lo que he logrado y por la confianza depositada en mí, te dedico este trabajo.

A mi sobrina Melani y a mi ahijada Gleidys, solo les muestro el camino, transitarlo no es fácil pero se puede, las quiero mucho.

A mis primos, que son lo más parecido a hermanos que tengo.

RESUMEN

Con el presente trabajo se pretende realizar la segunda versión del diseñador y generador de formularios web, perteneciente a la línea Integración de Soluciones del Centro de Tecnologías de Gestión de Datos (DATEC), centro vinculado a la facultad 6 de la Universidad de las Ciencias Informáticas (UCI). La segunda versión del Diseñador y generador de formularios web surge con el objetivo de mejorar las funcionalidades de la versión anterior, mostrándoles a los usuarios un producto con acciones nuevas que garanticen una mayor interacción con la aplicación, y además sea de fácil utilización por los usuarios. El diseñador y generador de formularios web en la versión 2.0 mantendrá los principios de diccionario de datos establecidos en la versión anterior, de esta manera no se pierden los conceptos por lo que fue creada dicha aplicación en sus inicios.

Para la realización de la nueva versión de la aplicación se modelaron los flujos de trabajo de análisis, diseño, implementación y prueba utilizando la metodología de desarrollo OpenUP y el lenguaje de modelado visual UML 2.0. La aplicación ha sido implementada con tecnologías actuales como: PHP 5.2, JavaScript, ExtJS 3.2. Además se utilizó herramientas como el Visual Paradigm 8.0 para la creación de diagramas e interfaces de usuarios no funcionales, y como entorno de desarrollo integrado (IDE) NetBeans 7.1.

PALABRAS CLAVES

DATEC, diseñador, generador, formularios web.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El vertiginoso desarrollo de la ciencia y la técnica ha conquistado todos los ámbitos y niveles sociales, las nuevas tecnologías se han convertido en uno de los productos de mayor consumo en la actualidad. (1) De ellas dependen en gran escala el desarrollo de las sociedades y la economía de los países en el presente siglo.

Como consecuencia de este proceso tecnológico evolutivo, surgen las aplicaciones web, conocidas por representar páginas con contenidos dinámicos que permite al consumidor mayor interacción. Las aplicaciones web son accedidas desde navegadores web y deben su popularidad a la independencia del sistema operativo, a lo cómodo y práctico del uso del navegador y que no se distribuyen o instalan en las máquinas clientes (2), lo cual hace más fácil la actualización y mantenimiento de la aplicación. Formando parte de ellas se encuentran las que se encargan de **recibir, guardar y analizar datos** proporcionados por el usuario, siendo la captura de datos una de la tareas más importantes a llevar a cabo, por lo que se hace vital, que los sistemas electrónicos o las aplicaciones web, que tengan como principio gestionar la información que el usuario le suministre, presenten dentro de sus funciones un modelo de captura de datos eficiente.

La captura de datos es el proceso mediante el cual, en un sistema informático, se obtienen y registran los datos que van a ser procesados para ser convertidos en información útil para el usuario (3). Para realizar la captura de datos se deben utilizar herramientas que ayuden en el proceso, una de las más aceptadas para realizar la tarea son los formularios. Los mismos son utilizados en listas de correos, registro de eventos, solicitud de soporte, encuesta de satisfacción, reservaciones en líneas, solicitud de empleo y en formularios de contactos, pero el crearlos y elaborarlos podría resultar complicado si no se cuenta con los conocimientos necesarios para lograr cumplir el objetivo lo que implica aumento en tiempo y esfuerzo para quienes realicen dicha tarea.

Para evitar este tipo de inconveniente, se han venido desarrollando los diseñadores y generadores de formularios web, que dotan a los sistemas de novedosas funcionalidades. En la actualidad se hace de gran importancia contar en las entidades con este tipo de componente, capaz de diseñar y generar formularios web, ya que sin estas herramientas resultaría difícil realizar la captura de información y se vería entorpecido el aumento de la productividad de las entidades que se encuentren en dicho proceso.

La Universidad de la Ciencias Informáticas (UCI), hoy cuenta con varios centros vinculados a la producción de software, dentro de la misma se encuentra el Centro de Tecnología de Gestión de Datos

(DATEC), constituido por cuatro líneas fundamentales de producción, una de ellas es la línea de Integración de Soluciones en la cual se están desarrollando varios productos que son integrables al Paquete de herramientas para la Ayuda a la Toma de Decisiones Integrales (PATDSI), plataforma tecnológica del centro, que dentro de sus funciones se encuentra la captura de información tanto en gestión estadística como en sistema de encuesta.

Como parte del desarrollo tecnológico del departamento ha sido creado el diseñador y generador de formularios web (Disform), que permite la elaboración de formularios sencillos para la web, lo cual facilita el desarrollo de páginas dinámicas que necesiten obtener y procesar información obtenidas del usuario. Actualmente dicha aplicación no cuenta con todos los elementos de captura de datos para una mayor explotación, por lo que se hace necesario subsanar esta y otras dificultades que presenta hoy el diseñador y generador de formularios web en la versión 1.0, para lograr una herramienta de diseño con la que el usuario mantenga una interacción fluida, por lo que una versión 2.0 del Disform se hace inminente la cual va a mantener el principio de modelo relacional y diccionario de datos establecidos en la primera versión.

El estudio de la problemática anterior desencadena el siguiente **problema a resolver**: ¿Cómo mejorar las funcionalidades del diseñador y generador de formularios web?

Para llevar a cabo la investigación se ha definido como **objeto de estudio**: Diseñador y generador de formularios web.

Campo de acción: Componentes para el diseño y generación de formularios web.

Siendo el **objetivo general**: Desarrollar el componente Diseñador y generador de formularios web v2.0.

Para alcanzar el objetivo trazado se debe dar cumplimiento a los siguientes **objetivos específicos**:

- Realizar análisis y diseño del componente Diseñador y generador de formularios web v2.0.
- Implementar las funcionalidades del componente Diseñador y generador de formularios web v2.0.
- Realizar pruebas al Diseñador y generador de formularios web v2.0.

Para dar solución a la problemática planteada y poder lograr los objetivos, se proponen las siguientes tareas de investigación.

Tareas de investigación

- Investigación de las tecnologías actuales de diseñadores y generadores de formularios web.
- Estudio de los widgets en los formularios.
- Definición de los requisitos funcionales y no funcionales del componente a implementar.
- Selección de los artefactos que se obtendrán según la metodología de desarrollo que se seleccione.
- Diseño de la aplicación web en su versión 2.0.
- Implementación y pruebas de las nuevas funcionalidades del componente de diseño.

El trabajo está estructurado en tres capítulos, como se describe a continuación:

Capítulo 1: Fundamentación teórica. En este capítulo se realiza un estudio acerca de los formularios, describiendo las principales causas de los malos diseños de los mismos, así como el principio que se debe tener en cuenta en la realización del componente para la interacción con los usuarios. De igual manera se realiza un estudio de algunas soluciones existentes en el mundo que diseñan y generan formularios web. Concluye el capítulo con la selección del mecanismo más indicado para desarrollar la aplicación.

Capítulo 2: Análisis y diseño. En este capítulo se hace una descripción del sistema a automatizar, en el cual se mostrará el diagrama de casos de uso del sistema para su mejor entendimiento. Se definen los actores del sistema, así como los requisitos funcionales que el sistema debe poseer y los requisitos no funcionales mínimos que debe tener el cómputo donde se va a instalar el sistema. Además se describen los patrones de diseño y de arquitectura que se tuvieron en cuenta durante el desarrollo de la aplicación y se evidencia su utilización. Se realiza el diagrama de clases del diseño de los casos de uso significativos, así como los diagramas de secuencia por escenarios de los mismos.

Capítulo 3: Implementación y Prueba. En este capítulo se muestra el diagrama de componentes generados en la implementación del sistema, mostrándose la relación entre los archivos que componen el diagrama. Se describe los estándares de codificación a utilizar en la implementación y se realizan pruebas de Caja Blanca a código fuente para validar la aplicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se realiza una breve reseña acerca de los formularios web, se referencian conceptos relacionados con los diseñadores y generadores de formularios web que resultan de vital importancia para entender la solución que se propone. Se abordará sobre el análisis del estado de arte de estos componentes, así como los principales paradigmas que se tienen en cuenta en la realización de la aplicación.

1.1. Diseño web

El diseño es uno de los procesos más importantes dentro de la elaboración de productos de software, sin este elemento no se podría asegurar de algún modo la calidad de los servicios a brindar, ya que define el proceso previo de configuración mental en la búsqueda de una solución en cualquier campo (4), es decir, es una actividad creativa orientada a proyectar y confeccionar algo, que no ha existido con anterioridad, para la solución de problemas.

Como concepto es aplicado en diferentes disciplinas; en la informática se encuentra el **diseño web**, que es la habilidad de planificar, diseñar e implementar sitios y páginas web. No es una aplicación más del diseño tradicional, sino que es un diseño que tiene dentro de su concepto elementos tales como la navegabilidad, usabilidad, arquitectura de información e interactividad (5).

1.2. Diseño de formularios web

Como parte del proceso de desarrollo de aplicaciones y páginas web, se debe tener en cuenta el diseño de herramientas como los formularios, que representan una parte importante dentro de las aplicaciones que mantienen interacción directa con los usuarios y necesitan de la información que se recopilan a través de ellos.

El formulario es definido como un fragmento de espacio de la pantalla, habitualmente rectangular, que puede utilizarse para prestar información al usuario y para aceptar las entradas que realice (6), lo que permite gestionar la información introducida. Utilizar los formularios tiene sus ventajas debido a que presentan la información de forma clara, ofrece control de los datos y asegura potencialmente la captura de información. Además presentan una visión ordenada de múltiple información, ofrecen gran

flexibilidad en los procesos que tienen que ver con cargar, editar y eliminar datos de la base de datos. (7)

Los formularios web se ven en todas partes, en buscadores como el de google en el cual se debe llenar un formulario para indicar la búsqueda. Para la confección de formularios web se pueden combinar varios lenguajes de programación o scripts para ser procesados tanto en el lado del cliente como al lado del servidor. Al lado del cliente es muy utilizado JavaScript para la validación de datos, con este lenguaje el script es ejecutado en el navegador del usuario. Mientras tanto al lado del servidor existen varios lenguajes que permiten autorizar el ingreso a un sistema, almacenar la información en la base de datos, realizar búsquedas. Entre los lenguajes más utilizados al lado del servidor son PHP, Java, C, Perl y ASP.

1.3. Causas de malos diseños

Actualmente el diseño de los formularios es bastante crítico, dado a que una mala concepción de él, puede convertirlo en una barrera para la interacción e inducir al usuario a cometer errores potenciando la frustración en el desempeño de sus tareas (7). Dentro de las principales causas de los malos diseños se encuentran:

El programador es quien hace habitualmente el diseño

Mientras que para el diseño de páginas web se considera necesaria la participación de especialistas en el diseño de interfaces, no ocurre lo mismo cuando debe diseñarse algún formulario. Es muy habitual considerar que el formulario no es más que el resultado del análisis de datos que debe realizar todo analista o programador. Lo que se obtiene al final es un formulario que no tiene en cuenta la terminología del usuario y que por lo general guía poco y controla mucho. (7)

No existen guías completas de diseño

La mayor parte de la literatura relativa a formularios se limita a listas de directrices como la de Shneiderman[1] (ver Anexo 1). En algunos casos, las empresas productoras proporcionan sus propias

[1] Shneiderman: es un estilo de interacción conocido en lengua anglosajona como Form Filling y una de las más aceptadas actualmente en el mundo.

guías de diseño, ejemplo las guías para Windows y las guías de Sun para Java. Pero estas guías se quedan la mayor parte de las veces en el diseño externo sin entrar en la parte de interacción y en muchos casos entran en contradicción las unas con las otras. (7)

Se crean por copia.

Aunque existen herramientas de ayuda para el diseño y codificación de formularios, la mayor parte de los desarrolladores prefieren, porque acostumbran a ser más productivo, empezar el diseño de un formulario a partir de la copia de otro anterior. Las consecuencias es que van heredando tanto los buenos como malos diseños. (7)

Es posible minimizar las causas de malos diseños, si se cuenta con especialistas en el tema que ayuden a definir los aspectos a tener en cuenta en la realización de un formulario y cómo debe ser la interacción con los usuarios, además contar con herramientas que ayuden a agilizar el proceso de diseño de formularios web, permitiendo que el diseñador se centre en donde ubicar cada uno de los elementos contenidos en el formulario, propiciando mejores resultados. Las herramientas que se utilizan para realizar la tarea más rápido son denominados diseñadores y generadores de formularios web.

1.4. Diseñadores y Generadores de formularios web

El Diseñador de formularios se puede definir como, un espacio de trabajo que reservan algunos entornos de desarrollo de los lenguajes de programación visual, donde despliegan un formulario por defecto para la creación de la interfaz de usuario. En este espacio denominado “diseñador del formulario”, es donde el programador inserta los controles que formarán la interfaz de usuario. (6)

Los diseñadores y generadores de formularios web son herramientas que ayudan exponencialmente al desarrollo de formularios web, garantizan que el usuario pueda crearlos sin poseer un nivel avanzado de lenguajes de programación, evitando así que se encuentre todo el tiempo implementando una herramienta de captura de datos a su medida. Entre las ventajas que presentan los diseñadores y generadores de formularios se pueden encontrar:

- Realizan la mayor parte de trabajo.
- Se pueden adaptar fácilmente a necesidades específicas.

- Se producen resultados más rápidos, sin efectos sobre la calidad del producto. (8)

Los diseñadores y generadores de formularios como sistemas que interactúan con los usuarios a gran escala deben seguir el principio de Visibilidad.

Principio de Visibilidad[2]

La visibilidad consiste en hacer que la interfaz de usuario muestre los componentes del modelo mental sobre el que se ha construido el sistema, es decir, que estimule a los sentidos del usuario. Una interfaz puede construirse de muchas formas, pero es necesario que sus objetos se vean, se oigan o que presenten cualquier característica que los haga perceptibles para que el usuario no sólo pueda interactuar con el sistema sino que también se vaya formando un modelo mental correcto del mismo. (9)

Seguir este principio tiene sus ventajas, debido a que la comunicación con los usuarios resultaría fructífera, ya que como clientes a la aplicación, sabrán qué es lo que deben hacer con solo ver iconos o textos que le ayuden entender las funcionalidades para la cual ha sido creado el sistema, de esta manera se llega a la interacción con el usuario sin tener que mostrarles ventanas de error todo el tiempo, que en ocasiones resulta molesto, porque se llega a controlar la acciones del usuario en vez de guiarlos por la aplicación.

1.4.1 Sistemas Diseñadores y Generadores de formularios web en la actualidad.

El estudio de los sistemas Diseñadores y Generadores de formularios existentes, facilitan la toma de decisión de cuál de las herramientas usar o cuáles de los sistemas se adecuan a las necesidades presentadas por los clientes. Cada una de las herramientas que se describen a continuación son muy utilizadas por las particularidades que implementan y los servicios que brindan.

Form1 Builder: esta herramienta según las bibliografías estudiadas es conocida porque corre en cualquier navegador de Internet y además permite construir fácilmente formularios web seguros. Entre las características que presenta se encuentran: interfaz amigable y construcción con un solo clic. Presenta la capacidad de desarrollar formularios básicos y complejos de forma fácil y rápida. Permite diseñar formularios de varias páginas. Posee las características de casi todas las páginas web y es

[2] Principio de visibilidad desarrollado por Donald Norman en su libro “La Psicología de los objetos cotidianos (1990)”

que para poder correr Form1 el servidor en el que se vaya a instalar debe soportar el lenguaje de programación PHP. La mayoría de los servidores web soportan PHP.

DesignExpert: a través de la investigación se pudo constatar que este software permite a usuarios diseñar formularios para reproducción de lectura por marcas ópticas (OMR) o aplicaciones de proceso de formularios por imágenes. Con esta herramienta se puede crear y personalizar formularios que se ajusten a necesidades específicas del usuario.

FormLogix: según las bibliografías esta aplicación permite crear formularios online, sin necesidad de saber nada de programación. Permite crear los formularios usando una interfaz 2.0 de alta calidad. Presenta compatibilidad con algunos navegadores como Firefox, Internet Explorer, Opera, Safari y Chrome. A pesar de ser una herramienta muy potente en la web, debemos destacar que tiene limitaciones, pues no es totalmente gratis, la versión gratuita después de pasado algún tiempo anula los formularios enlazados y se eliminan las cajas de herramientas para administración de información.

Wufoo: según lo investigado esta herramienta sirve para confeccionar formularios online. Tiene la particularidad que al diseñar un formulario se crea automáticamente la base de datos, y las herramientas necesarias para almacenar, Administrar y entender la información recogida.

Formspring: esta herramienta según la bibliografía consultada permite generar formularios online, que se puede insertar en los sitios web, cuenta con una serie de planes, siendo algunos de los mismos gratis, además se pueden generar formularios con la utilización de herramientas que permiten realizar diversas tareas como insertar campos personalizados, editores y ordenadores.

Flow: se pudo constatar a través de estudios de bibliografía que esta herramienta permite la generación de formularios web online, es una herramienta sencilla de utilizar ya que no requiere del conocimiento de programación básicos o avanzados del usuario.

QuestionForm: según las opiniones encontradas acerca de este producto se pudo apreciar que es muy utilizado porque crea formularios de manera rápida. Una de sus particularidades es que presenta variedad de idiomas, y a la vez que es creado el formulario tiene la opción de poder enviarlo por correo e incluirlo en el sitio del usuario. Esta herramienta permite monitorizar los resultados, la actividades y las estadísticas por vía SMS, RSS. Tiene la desventaja de tener problemas en la seguridad.

Estas herramientas son eficientes atendiendo a las particularidades que cada una implementa, pero no es posible su utilización en el centro, debido a que el diseño que se realiza a través de ellos no es integrable a las necesidades que presenta DATEC. Estos diseñadores permiten trabajar a partir de un modelo relacional, pero no permiten diseñar los formularios desconectados de la base de datos, lo cual conllevaría a que si en algún momento falla la conexión a la BD, el diseño del formulario se vería interrumpido, además, muchos de los diseñadores investigados poseen licencias comerciales.

1.4.2 Sistemas Diseñadores y Generadores de formularios web en Cuba

En Cuba los formularios como forma de captura de información son muy utilizados por diferentes sectores económicos y sociales de la isla. Presentando la cara del sistema e interactuando de forma fácil con usuarios identificados a nivel básico con la informática. En ocasiones se hace necesario contar en las entidades con formularios que puedan adecuarse a las necesidades específicas de las mismas, que ayuden a la gestión de la información, propiciando la toma de decisiones a nivel operativo táctico y estratégico de las entidades. Hasta el momento no se conoce de una implementación anterior de una aplicación que dentro de sus funcionalidades cuente con un diseñador y generador de formularios web que pueda vincularse con las particularidades de cada entidad.

1.4.3 Diseñadores y Generadores de formularios web en la UCI.

En la Universidad de las Ciencias Informáticas específicamente en el centro DATEC se encuentra implementado el Sistema Integral de Gestión Estadística (SIGE), esta herramienta perteneciente a PADTSI, permite al usuario diseñar formularios estadísticos, ofreciendo una serie de funcionalidades para definir los principales elementos que conforman los mismos. La aplicación contempla el diseño de formularios de nomenclatura cerrada y de nomenclatura abierta, así como la edición de formularios previamente creados y guardados y/o exportados desde la misma aplicación.

Como parte de los productos de la línea ha sido creado el IDE (Ambiente de Desarrollo Integrado) Lycan GENESIS que centra sus objetivos en la creación de aplicaciones enriquecidas para la web (RIA) utilizando las librerías de ExtJS. Lycan posee un diseñador establecido para encuestas, que dentro de sus principales funcionalidades se encuentran la creación de nuevas plantillas de encuestas, permite el redimensionamiento de los componentes de diseño, además permite exportar e importar el diseño de las plantillas creadas como archivo Zip, entre otras funcionalidades que hacen del diseñador una herramienta potente para la realización de formularios.

Aunque el centro DATEC cuenta con estas poderosas herramientas de diseño de formularios, y las mismas poseer principios tales como implementar diccionarios de datos, debido a la forma que fueron concebidos, estos no generan formularios automáticamente.

1.5 Arquitectura de Software

La arquitectura de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos. (10)

La arquitectura destaca decisiones tempranas del diseño que tendrían un profundo impacto en todo el trabajo de ingeniería del software que sigue. Facilita la comunicación entre todas las partes (partícipes) interesadas en el desarrollo del sistema. Constituye un modelo relativamente pequeño e intelectual comprensible de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes. (10)

1.5.1 Arquitectura basada en componentes

Los sistemas de software basado en componentes se basan en principios de una ingeniería de software específica (ISBC). La ingeniería de software basada en componentes ha cambiado la forma en que se desarrollan los sistemas de software. El programador cambia sus objetivos y pasa de programar el software a componer sistemas de software, para su posterior utilización como componentes de software reutilizables.

El objetivo de esta arquitectura es construir aplicaciones complejas mediante ensamblado de componentes que han sido previamente diseñados por otros desarrolladores o por subgrupos del equipo de desarrollo, a fin de ser rehusados en múltiples aplicaciones. En la arquitectura basada en componentes la interfaz constituye el elemento básico de conexión. Cada componente debe describir de forma completa las interfaces que ofrece, así como las interfaces que requiere para su operación. Además, debe operar con independencia de los mecanismos internos que utilice para soportar la funcionalidad de la interfaz. (11)

1.6 Patrones

En la terminología de objetos, el patrón es una descripción de un problema y su solución, que recibe un nombre y puede emplearse en otros contextos. Los patrones no proponen descubrir ni expresar

nuevos principios de la ingeniería del software. Todo lo contrario: intentan codificar los conocimientos y principios ya existentes. (12)

Los patrones de arquitectura describen un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución.

1.6.1 Patrón por Capas

El patrón por capas es similar a al estilo por capas: y es definido según Garlan y Shaw. : como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior.

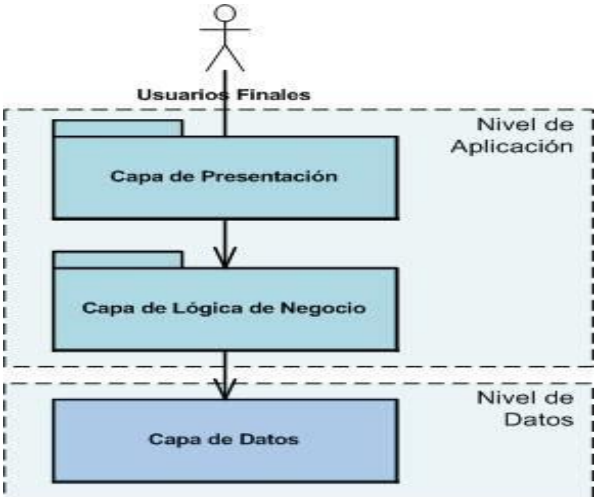


Figura 1: Arquitectura en Capas

El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. El estilo admite muy naturalmente optimizaciones y refinamientos, también proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones y versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas. (11)

1.6.2 Patrones de Caso de uso.

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 mantenible, reusable y entendible. Entre los patrones de casos de uso, se encuentran:

CRUD (Create, Read, Update, Delete) o Gestión de Información: Modela las diferentes operaciones que pueden ser realizadas sobre una entidad de información de un tipo específico, tales como creación, lectura, actualización y eliminación.

Inclusión o extensión concreta: El modelo de extensión consiste en dos casos de uso y una relación ampliada entre ellos. El caso de uso de extensión es concreto; es decir que puede estar instanciado, solo así se amplía el caso de empleo bajo.

1.6.3 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Además de permitir describir el sistema con el suficiente detalle para ser implementado.

Los patrones de diseño tienen como ventaja que proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Además están basados en la recopilación del conocimiento de los expertos en desarrollo de software por lo que es una experiencia real, probada, que funciona y ayuda a no cometer los mismos errores.

Entre los patrones más empleados se encuentra **GRASP** (las siglas pertenecen a las palabras del inglés, General Responsibility Assignment Software Patterns) en el que se destacan 5 patrones fundamentales:

- **Experto:** Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. El patrón

Experto asigna responsabilidades a las clases que tienen la información necesaria para cumplir con la responsabilidad. (12)

- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. Lo que define este patrón es que una instancia de un objeto la tiene que crear el objeto que tiene la información para ello. (12)

- **Alta cohesión:** Mantiene la complejidad dentro de los límites manejables, es decir asigna una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuan 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. (12)

- **Controlador:** es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Asigna las responsabilidades de capturar los eventos del sistema a las clases. (12)

Dentro de los patrones de diseño podemos encontrar patrones **Gof (gang of four o pandilla de los cuatros)** dentro de los cuales se encuentran:

- **Patrones de creación:**

  - **Constructor:** separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones. (12)
  - **Método de la fábrica:** este patrón define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Además permite que una clase delegue en sus subclases la creación de objetos. (12)
  - **Prototipo**: especifica los tipos de objetos a crear por medio de una instancia de un prototipo, y crear nuevos objetos copiando este mismo prototipo. (12)
  - **Singleton**: Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. (12)

- **Patrones estructurales**

  - **Adaptador:** convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Además permiten que cooperen clases que de otra manera no podrían hacerlo, pues tendrían interfaces incompatibles. (12)
  - **Puente**: desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente. (12)
  - **Compuesto:** combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos. (12)
  - **Decorador:** añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender funcionalidades. (12)
  - **Fachada:** proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define la interfaz de alto nivel que hace que sea más fácil de utilizar el subsistema. (12)

- **Patrones de comportamiento**

  - **Iterador:** proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna. (12)
  - **Mediador:** define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente y permite variar la interacción entre ellos de forma independiente. (12)
  - **Observador:** define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan todos los objetos automáticamente. (12)
  - **Estado:** permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Tal parece que cambia la clase del objeto. (12)

1.7 Metodología de desarrollo de software

El desarrollo de software pude ser un reto para los desarrolladores, pues crear un software en el menor tiempo posible y con calidad puede resultar difícil, sino se cuenta con algún proceso que ayude a agilizar el desarrollo de los mismos.

Desde hace algún tiempo se vienen utilizando las metodologías, ya que imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente.

Un proceso de desarrollo de software define quien está haciendo qué, cuándo y dónde para alcanzar un determinado objetivo; en la ingeniería el objetivo es construir un producto de software o mejorar uno existente con calidad. (13)

Dentro de las metodologías de desarrollo de software existentes se encuentra **OpenUp**, proceso de desarrollo basado en Rational Unified Process (RUP), desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad, basándose en los principios de adaptación, importancia a los involucrados e interesados en los resultados del proyecto, colaboración, valor a la iteración, y calidad continua. OpenUp/Basic permite un abordaje ágil al proceso de desarrollo de software con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles y tareas. OpenUp/Basic es un proceso interactivo de desarrollo de software simplificado, completo y extensible. (14)

Esta metodología es apropiada para proyectos de bajos recursos, debido a que permite disminuir las probabilidades de fracaso en los proyectos e incrementar las probabilidades de éxito. Permite detectar errores en fases tempranas a través de un ciclo iterativo. Una de sus principales ventajas está dada por la poca elaboración de documentación, diagramas e iteraciones requeridos en la metodología RUP. Dicha metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

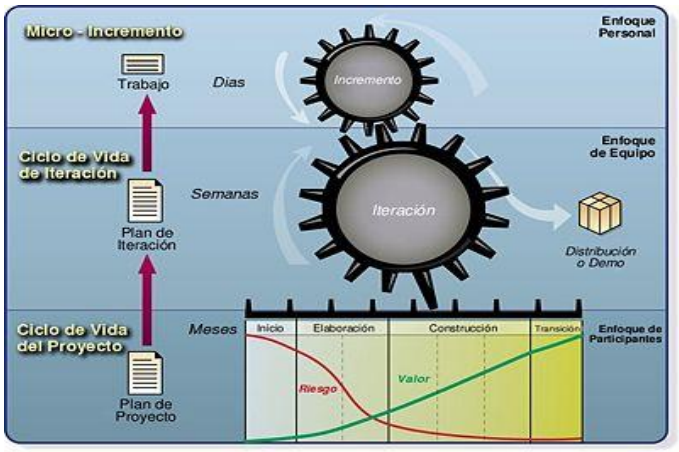


Figura 2 Ciclo de vida de OpenUP

Fases del OpenUP

1. Concepción

Concepción es la primera de las cuatro fases del ciclo de vida de OpenUP, en ella se realiza el entendimiento de los objetivos del proyecto, de esta manera se obtiene suficiente información para confirmar lo que el proyecto debe hacer. El objetivo de ésta fase es capturar las necesidades de los *stakeholders* (clientes) en los objetivos del ciclo de vida para el proyecto. (15)

2. Elaboración

Es la segunda de las cuatro fases del ciclo de vida del OpenUP donde se tratan los riesgos significativos para la arquitectura. El propósito de esta fase es establecer la base para la elaboración de la arquitectura del sistema. (15)

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 basándose en la arquitectura definida. (15)

4. Transición

Es la última fase, cuyo propósito es asegurar que el sistema es entregado a los usuarios, y evalúa la funcionalidad y performance del último entregable de la fase de construcción. (16)

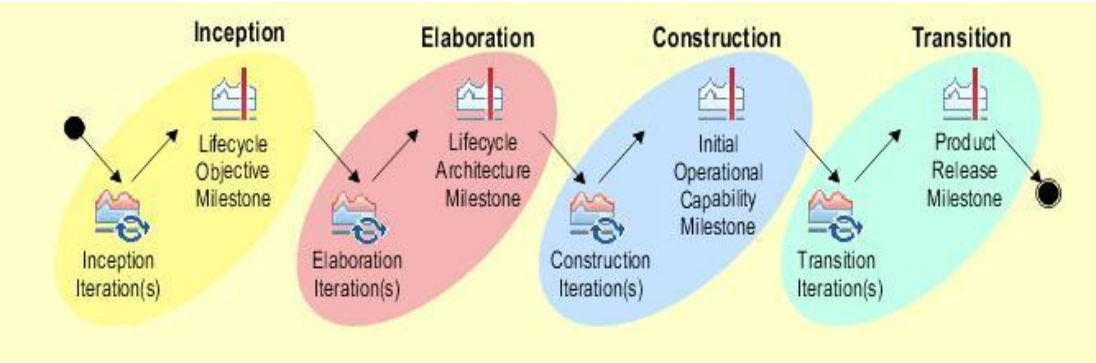


Figura 3 Iteraciones de OpenUP

Roles en la solución

- **Analista**

La persona en este cargo representa al cliente y las preocupaciones de los usuarios finales mediante la recopilación de los grupos interesados para entender el problema a resolver y por la captura y fijación de prioridades para los requisitos. (17)

- **Desarrollador**

Es responsable del desarrollo de una parte del sistema, incluyendo el diseño que se ajuste a la arquitectura, posible creación de un prototipo de la interfaz de usuario y, a continuación, la aplicación, la unidad de pruebas, y la integración de los componentes que forman parte de la solución. (17)

1.8 Herramientas y tecnologías a emplear

1.8.1 UML 2.0

El Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Es la especificación de OMG (Grupo de gestión de objetos, en inglés Object Management Group) y no solo es la forma mundial de representar la estructura de las aplicaciones, comportamiento y arquitectura sino que también representa procesos de negocio y estructura de datos. (18)

Permite el modelado de sistemas con tecnología orientada a objetos y no es considerado un proceso. Este lenguaje tiene elementos que conforman su vocabulario. En los diagramas se relacionan estos conjuntos de elementos y son los encargados de visualizar un sistema desde diferentes perspectivas.

Ventajas

- Concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actuales y futuras.
- Ampliamente utilizado por la industria desde su adopción por OMG.
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.

- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario. (12)

1.8.2 Herramienta CASE (Computer Aided Software Engineering)

Las herramientas CASE son muy utilizadas en la actualidad porque son aplicaciones que ayudan a aumentar la productividad del desarrollo de software reduciendo costes, tiempo y dinero de los sistemas en confección. El uso de las herramientas CASE presenta para el usuario que las usa varias ventajas entre las que se encuentra que ayuda a mejorar la calidad del software.

En el mundo hoy existen varias herramientas CASE, entre ellas se encuentra el Visual Paradigm, herramienta que será utilizada para la confección del producto.

Visual Paradigm for UML 8.0

Es una herramienta CASE, desarrollada por la compañía Visual Paradigm International, que utiliza UML para el modelado de los diagramas, está diseñado para desarrollar software a partir de la programación orientada a objetos Es una herramienta que soporta el ciclo de vida completo del software, permite la captura de requisitos y la realización de ingeniería inversa. Una sus mejores características está dada por la integración a varios IDE de desarrollo, como Eclipse/IBM WebShpere, NetBeans, Oracle Jdeveloper, entre otros.
El Visual Paradimg, es una herramienta de modelado visual
Además es soportado por varios Sistemas operativos entre los que se encuentra Ubuntu, Windows y Mac OS Está disponible en varias ediciones, cada una destinada a necesidades específicas: Enterprise, Professional, Community, Standard, Modeler y Personal.

1.8.3 Tecnologías de desarrollo del lado del cliente.

JavaScript

JavaScript se implementó por primera vez en la versión beta de Netscape Navigator 2.0 en junio de 1995, por la empresa Netscape Communications Corporation. Es un lenguaje de programación

interpretado, lo que quiere decir que no requiere de compilación, es muy utilizado en la realización de páginas web dinámicas. No es un lenguaje orientado a objetos pero debido a que es basado en prototipos se puede aplicar conceptos de la programación orientada a objetos, es estructurado y presenta un tipado dinámico, su potencia radica en la compatibilidad con los distintos navegadores ya sean estos modernos o no. Este lenguaje se ejecuta al lado del cliente, lo que evita la sobrecarga del servidor por las peticiones concurrentes que pueden tener respuestas localmente. Se encuentra vinculado a las tecnologías AJAX para enviar y recibir peticiones al servidor

AJAX

AJAX (Asyncrhonous JavaScript + XML) aparece por primera vez, publicado en un artículo por Jesse James Garret el 18 de febrero de 2005. Hasta ese entonces no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación web que está surgiendo. AJAX la definen como una tecnología y en realidad se trata de varias tecnologías independientes que se unen para formar este nuevo término. Entre las tecnologías que la forman se encuentran, XHTML, CSS, JSON, DOM, JavaScript y XMLHttpRequest. El uso de AJAX permite mejorar la interacción del usuario con la aplicación, evitando las recargas constantes de páginas, lo cual propicia que la comunicación con el usuario no sea interrumpida.

1.8.4 Lenguaje de programación al lado del servidor

PHP 5.2

PHP (Hyper Preprocessor) es lenguaje interpretado de alto nivel introducido en páginas HTML y ejecutado en el servidor. Este lenguaje de programación permite crear páginas web dinámicas y la creación de aplicaciones gráficas independientes del navegador.

PHP presenta interfaces para una gran cantidad de sistemas de base de datos diferentes, 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.

1.8.5 Marco de desarrollo para la aplicación web

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 que ayuda a la rápida confección de sitios web. Los frameworks desde su creación han sido muy utilizados por los programadores y diseñadores porque los provee de una mejor estructura y organización en sus proyectos. Para las aplicaciones desarrolladas en PHP existe una cantidad considerable de frameworks libres de mucha calidad y renombre que pueden ser utilizados entre los que se pueden encontrar a Zend Framework, y Symfony entre muchos otros. En esta investigación se hablará del marco de desarrollo ExtJS, debido a su uso en la implementación del diseñador y generador de formularios web referidos en el documento.

ExtJS 3.2

ExtJS es una librería de JavaScript para el desarrollo de aplicaciones enriquecidas para la web (RIA) haciendo un uso extensivo de AJAX. Su potencia reside en la colección de componentes para el diseño de interfaces de usuario (GUIs). En sus inicios ExtJS fue creado como una extensión de Yahoo User Interface (YUI) otro framework bastante similar. En el momento que comenzó a emplearse este marco de 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, siendo el framework aceptado por numerosos usuarios.

1.8.6 Herramienta de Programación

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. Tiene buen soporte de Portal en Perl para Automatización de Web (Webapps) (war, jsp y Servlets) y actualmente cuenta con el plugin *Portal Pack* para la realización de Portlets. Como parte las características generación automática de código, refactorización inteligente, establece el formato del código, soporta librerías de terceros.

Estas características presentes en el IDE lo convierten en uno de los más potentes en la actualidad. El IDE cuenta con mejoras que permite el desarrollo en PHP, JavaScript, CSS, HTML, XHTML, java entre otras tecnologías. Lo que hace de este IDE idóneo para la implementación es que permite la interacción con las librerías de ExtJS y la generación automática de códigos para los lenguajes que soporta. Por estas razones se utilizará la herramienta para la confección del Diseñador y generador de formularios en su versión 2.0

1.9 Conclusiones

En el presente capítulo se explicaron los fundamentos teóricos de los diseñadores y generadores de formularios web, los cuales son herramientas muy utilizadas por la ayuda que brindan en la confección de los formularios. Para dar solución a la problemática planteada se decidió hacer uso de la metodología de desarrollo OpenUP, el lenguaje de modelado visual UML 2.0, la herramienta CASE Visual Paradigm 8.0 para la confección de los diagramas y como lenguaje de programación, JavaScript. Como framework de desarrollo ExtJS y para la implementación, el uso del IDE NetBeans.

2 ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En este capítulo se realiza una descripción de la aplicación, mostrándose el actor que interactúa con el sistema, los requisitos mínimos que debe tener el cómputo donde se instale la aplicación, así como los requisitos funcionales que el componente debe poseer. Se muestra el modelo de dominio del sistema, el diagrama de casos de uso del sistema para la mejor comprensión de la integración de la nuevas funcionalidades del componente diseñador y generador de formularios web. Además de los principales elementos del diseño del sistema.

2.4 Modelo de dominio del sistema

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema representando los objetos existentes o eventos que suceden en el entorno en el que se trabaja. (13)

Muchos objetos del dominio pueden obtenerse de una especificación de requisitos o mediante entrevista a expertos del dominio. Entre los prinipales objetivos de este modelo se encuentra la comprensión y descripción de las clases más importantes dentro del sistema. La metodología usada propone realizar el modelo de dominio cuando no existen procesos definidos en el negocio.

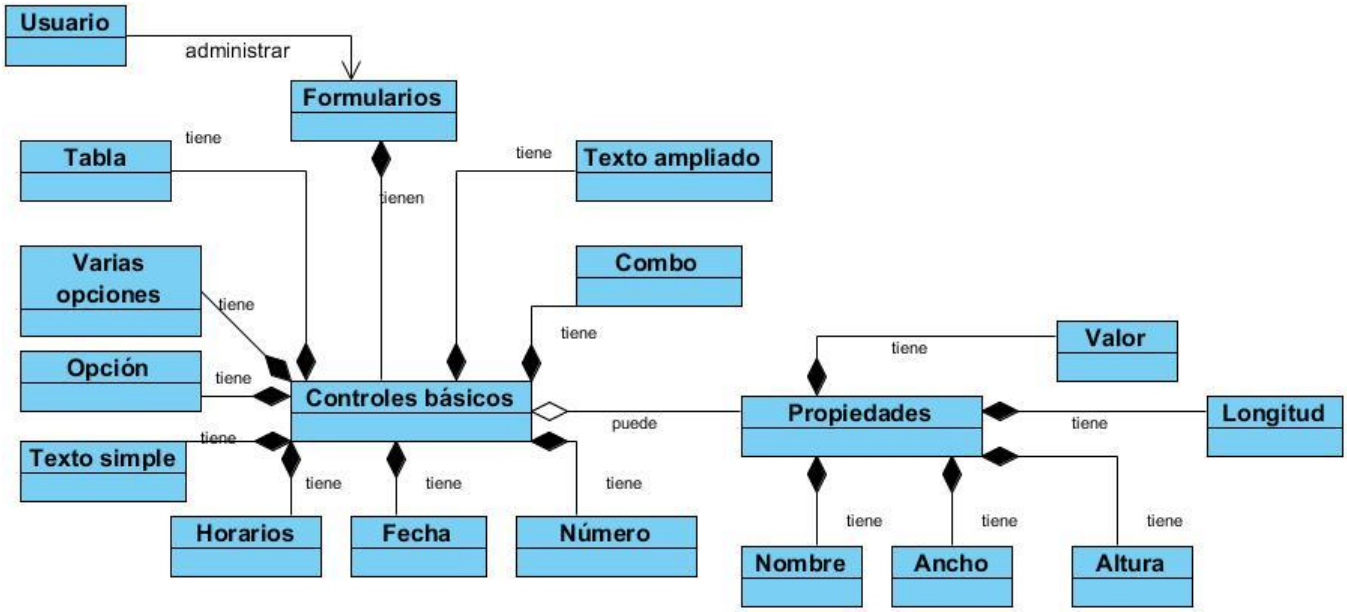


Figura 4 Modelo de dominio del sistema

2.4.1 Clases Conceptuales del Dominio

Clase Usuario

Representa a la persona encargada de realizar los formularios, es quien los crea, elimina, importa, exporta y genera. Además adiciona en los Dicionarios de datos las entidades y atributos, así como los controles básicos que hacen posible la confección de los formularios.

Clase Formulario

Es la clase encargada de realizar las diferentes acciones sobre los formularios, como crear, elimininar, exportar y generarlos desde el sistema. Estas acciones se realizan después de haber creado el Diccionario de datos, compuesto por entidades, a las cuales se les hace corresponder contenedores como paneles, ventanas, etc, además está compuesto por atributos que se les hace asociar controles básicos (textfield, checkbox, radiobutton, grid, textarea, textnumber, etc) dependiendo del tipo de dato que se especifique en la adición de los atributos.

Clase Controles básicos

Representa los principales elementos que componen a los formularios, estos puedes ser radiobutton, checkbox, textfield, grid, entre otros. Esta clase es la encargada de administrarlos, permitiendo que se realicen varias acciones sobre ellos, entre las que se encuentra la asociación de atributos a estos componentes y la eliminación de los controles básicos.

Clase Tabla/Grid

Representa uno de los widgets[3] de los fomularios, es una tabla dinámica con la que se puede capturar más datos que con una tabla normal.

Clase Varias opciones/Checkbox

Representa uno de los widgets de captura de datos, con el que se puede marcar varias respuestas u opciones para una pregunta.

Clase Opción/RadioButton

Representa un elemento del formualrio que acepta solo una opción a realizar, es decir, se utiliza cuando se tienen varias opciones y se necesita de una sola respuesta.

---

[3] Es la combinación de las palabras window (ventana) y gadgets (dispositivos)

Clase Horarios/TimeField

Representa un campo de formularios que establecen algunos lenguajes para mostrar un conjunto de horarios y se debe escoger uno de ellos en la captura de datos.

Clase Texto simple/TextField

Representa un campo de texto básico que almacena datos compuestos tales como números, caracteres en mayúscula o minúscula, además de caracteres especiales como el @, *, entre otros.

Clase Número/NumberField

Representa un tipo de campo que establecen algunos lenguajes para aceptar entrada de datos numéricos.

Clase Texto ampliado/TextArea

Son un tipo de campo que permite la entrada de varias líneas de textos.

Clase Fecha/Date

Representa un campo que permite almacenar una fecha.

Clase Propiedades

Es la clase encargada de administrar las proppiedades de los formularios y los controles básicos, permitiendo renombrar o redimensionar controles básicos y formularios.

Clase Valor

Representa un atributo de propiedad que define un valor por defecto.

Clase Longitud

Representa un atributo de propiedades que define el largo de un elemento del formulario, el cual puede ser cambiado por el usuario si desea.

Clase Altura

Representa un atributo de propiedades que define la altura de un elemento del formulario, el cual puede ser cambiado por el usuario si desea.

Clase Ancho

Representa un atributo de propiedades que define el ancho de un elemento del formulario, el cual puede ser cambiado por el usuario si desea

Clase Nombre

Representa un atributo de propiedades que define el nombre de un elemento del formulario, el cual puede ser cambiado por el usuario si desea.

2.5 Requisitos

2.5.1 Requisitos no funcionales

Los requisitos no funcionales describen las propiedades o cualidades que el sistema debe tener. Debe pensarse como propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. Los requisitos no funcionales de este sistema a automatizar son:

Usabilidad

RNF 1 Facilidad de uso

El sistema será fácil de manipular por cualquier persona, incluyendo aquellos que no cuenten con mucho conocimientos informáticos.
El sistema debe ser intuitivo y tener un alto nivel de usabilidad permitiendo que usuarios con nivel básico en la informática, en aproximadamente 30 días puedan explotarlo al máximo.

Fiabilidad

RNF 2 Disponibilidad

El sistema debe estar funcionando en todo momento, solo se detendrá en caso de fallo eléctrico, se reinicie la pc en la que se encuentra montado por motivos de configuración o cambios significativos en la aplicación.

RNF 3 Disponibilidad de servicios

El sistema debe mantener el servicio en todo momento, permanecerá inactivo o fuera de servicio en caso que se estén realizando tareas de mantenimiento o de configuración.

Restricciones de diseño e implementación

Lenguaje de programación JavaScript.

NetBeans 7.1 como IDE de desarrollo.

Visual Paradigm como herramienta CASE.

Lenguaje de modelado UML 2.0.

Framework ExtJS 3.2.

PHP 5.2

Interfaz

Interfaces de usuario

La interfaz de usuario debe ser amigable y fácil de entender, para evitar que pueda perderse en la navegación del sistema.

2.5.2 Requisitos funcionales

Los requisitos funcionales describen lo que el sistema debe hacer, representa capacidades o condiciones que el sistema debe poseer, independientemente de las cualidades o propiedades con las que se relacione. Los requisitos funcionales del sistema a automatizar son:

RF1.1 Exportar formulario como archivo js

RF1.2 Generar formulario

RF 2.1 Asociar control básico

RF 2.2 Adicionar control básico

RF 2.3 Eliminar control básico

RF3.1 Editar propiedades

2.6 Modelo de Casos de Uso del Sistema (CUS)

Un modelo es una abstracción del sistema, especificando el sistema modelado desde un cierto punto de vista y en un determinado nivel de abstracción. Una vista, es por ejemplo, una vista de especificación o una vista de diseño del sistema. (18)

En el modelo de casos de uso, el sistema contiene casos de uso y los actores que inicializan estos casos de uso. Los actores del sistema son aquellos individuos u otros sistemas externos que interactúan con el software. El actor establece un rol en el cual varios individuos pueden jugar ese papel. Para el desarrollo de los componentes se identificaron los actores siguientes.

Actor	Descripción
Diseñador	Persona encargada de realizar las diferentes acciones sobre los componentes de un formulario en el Diseñador y generador de

	formularios.

Tabla 1: Descripción Actores del Sistema.

2.6.1 Definición de los casos de usos del sistema.

Luego de haber descrito los actores del sistema se realiza el proceso de identificación de los casos de usos, con ayuda de los requisitos funcionales del sistema, los casos de uso identificados son:

Casos de uso base

CU Administrar formulario.

Casos de uso extendidos e incluidos

CU Administrar controles básicos.

CU Editar propiedades.

2.6.2 Diagrama de casos de uso del sistema

El diagrama de casos de uso representa la interacción entre el cliente (Actor) y el sistema que se encuentra en desarrollo, además del orden en que los elementos (casos de uso) interactúan entre sí.

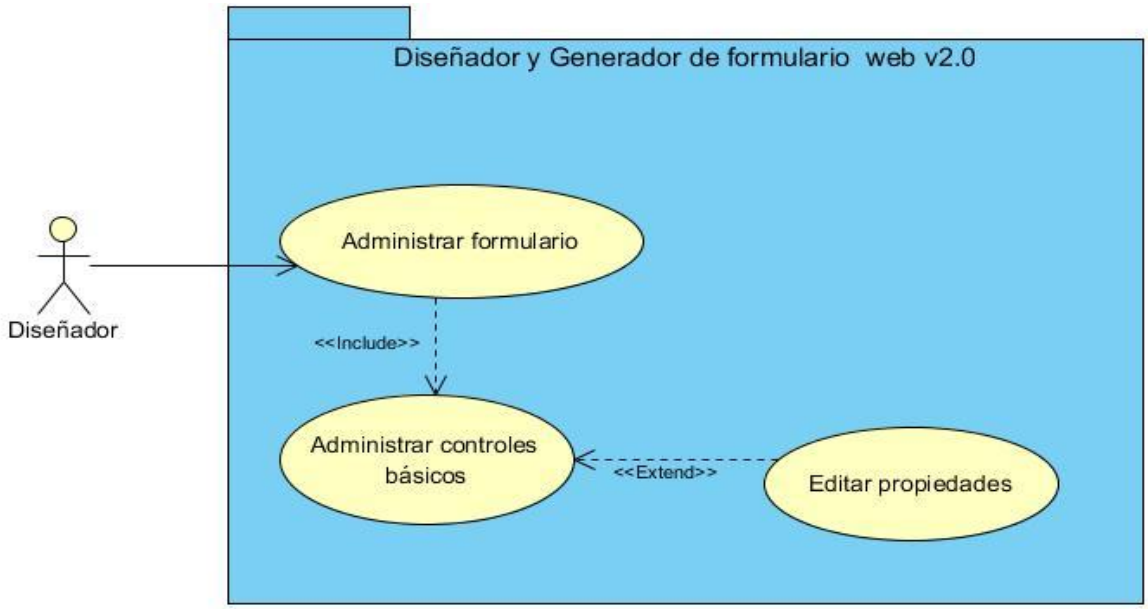


Figura 5 Diagrama de Casos de Uso del Sistema

2.6.3 Descripción del caso de uso Administrar formulario

Caso de Uso:	**Administrar formulario**
Actores:	**Diseñador**
Resumen:	El caso de uso se inicia cuando el diseñador decide realizar algunas de estas acciones. **Generar formulario:** el diseñador necesita trabajar más rápido en la confección del formulario y decide generarlo. **Exportar formulario como archivo js:** el diseñador después de haber creado el formulario, decide guardar el código como archivo js.
Precondiciones	Entidades y atributos creadas Formulario creado
Referencias	RF 1.1, RF 1.2,
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador elige la opción que desea realizar: • Generar formulario • Exportar formulario como archivo js	2. El sistema muestra la interfaz en correspondencia de la opción seleccionada. • Generar formulario. Ir a la sección ¨Generar formulario¨. • Exportar formulario como archivo js. Ir a la sección ¨Exportar formulario¨.

Sección "Generar formulario"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema genera el formulario. 2. El sistema envía un mensaje de confirmación del formulario generado. Terminando así el CU.
Sección "Exportar formulario como archivo js"	
Acción del Actor	**Acción del Sistema**
	1. El sistema muestra la ventana del navegador para realizar la descarga o abrir el archivo.
2. El diseñador decide entre salvar el archivo js o abrirlo en el navegador	
	3. El sistema guarda el archivo en la carpeta descarga de la pc cliente o abre el archivo en el navegador. Finalizando así el CU.
Flujos Alternos	
2a. El diseñador cancela la operación	**2b**. El sistema muestra la interfaz anterior.
Prototipo de Interfaz	
Pos condiciones	Formulario generado exportado y/o guardado o importado.

Tabla 2: Descripción del CU Administrar formulario

2.6.4 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 mantenible, reusable y entendible. Estos patrones tienen un enfoque hacia el diseño, y las técnicas son utilizadas en modelos de alta calidad

Para la realización del diagrama de casos de uso del sistema fueron utilizados el patrón CRUD parcial, extensión e inclusión Concreta.

El patrón **CRUD** es definido a partir de agrupar determinados requisitos funcionales, que representan las acciones de adicionar y eliminar determinada información, en este caso es evidenciado en los casos de uso administrar formulario y administrar controles básicos.

El patrón extensión concreta se evidencia en la posible inicialización del caso de uso editar propiedades de los controles básicos, a partir de que el diseñador haya creado un formulario en el área de trabajo, lo cual quiere decir que si el diseñador lo desea puede eliminar el formulario y los elementos que lo componen.

Otro de los patrones utilizados fue inclusión concreta, debido a que conformar un formulario no es solo dejar en el área de trabajo una ventana, sino que se hace necesario contar con elementos dentro de él, en este caso los controles básicos representan una parte importante, ya que sin ellos el formulario sería un contenedor vacío y no representaría absolutamente nada.

2.7 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.

2.7.1 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.

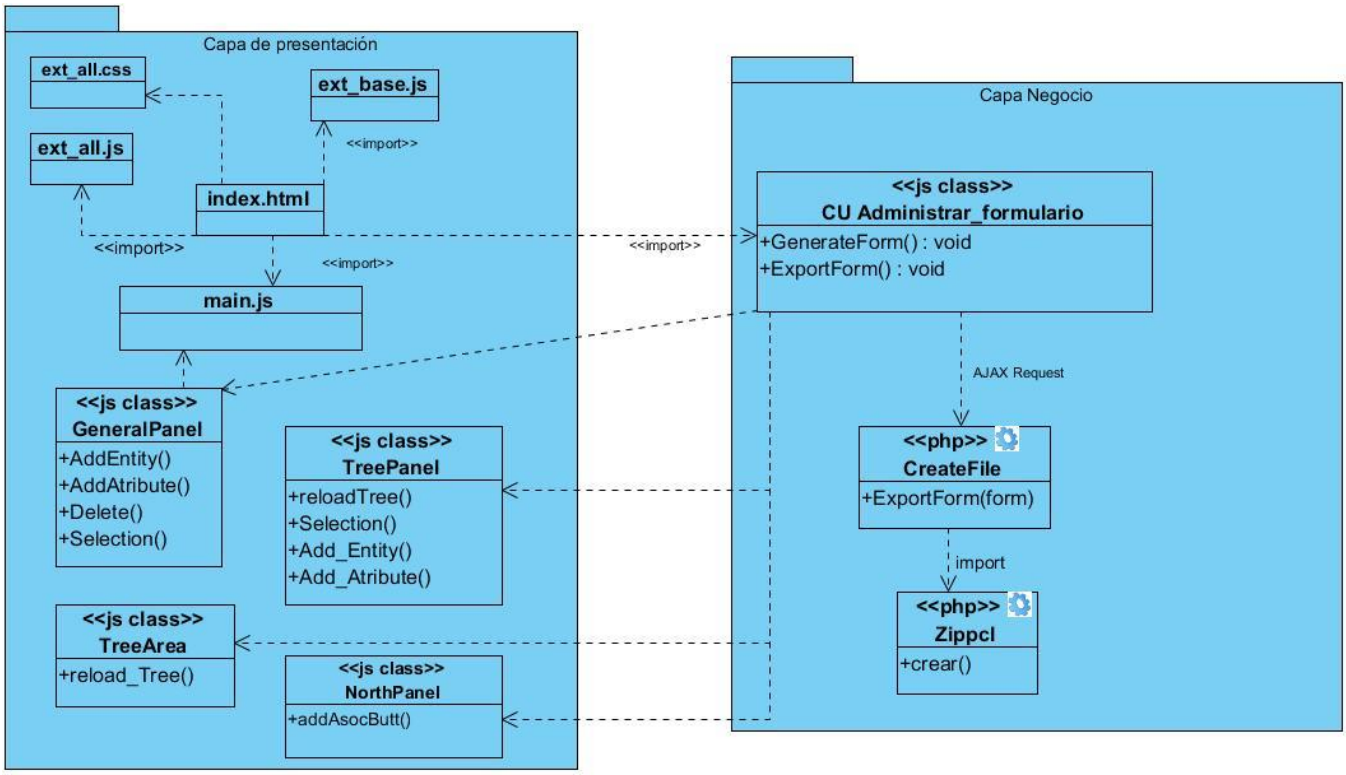


Figura 6 Clases de Diseño del CU Administrar formulario

Principales Clases

General Panel: este componente representa toda la interfaz principal, aquí es donde se crean los componentes de las clases. GeneralPanel hace la función de observador esperando a que los demás componentes lancen sus eventos y notifica de la actualización a los interesados.

TreePanel: este componente es el encargado de mostrar el árbol representado por los Diccionarios de datos, las entidades como subnodos del diccionario de datos, y los atributos como hijos de las entidades.

TreeArea: este componente es el que contiene al TreePanel y la principal función que realiza es la de recargar el árbol y todas las funciones que se realizan en el TreePanel.

Administrar formularios: esta clase es parte del negocio, es la encargada de realizar las diferentes acciones sobre los formularios, como generar el formulario, que se crea en el área de trabajo, exportar el formulario creado como archivo js, estas funciones serán enviadas a realizar desde el NorthPanel de la aplicación.

CreaFile: es la clase php que captura la información enviada a través de AJAX y crea un archivo zip para ser descargado por el navegador.

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, el diagrama de secuencia contiene los detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario además de los mensajes entre los objetos.

El diagrama de secuencia 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.

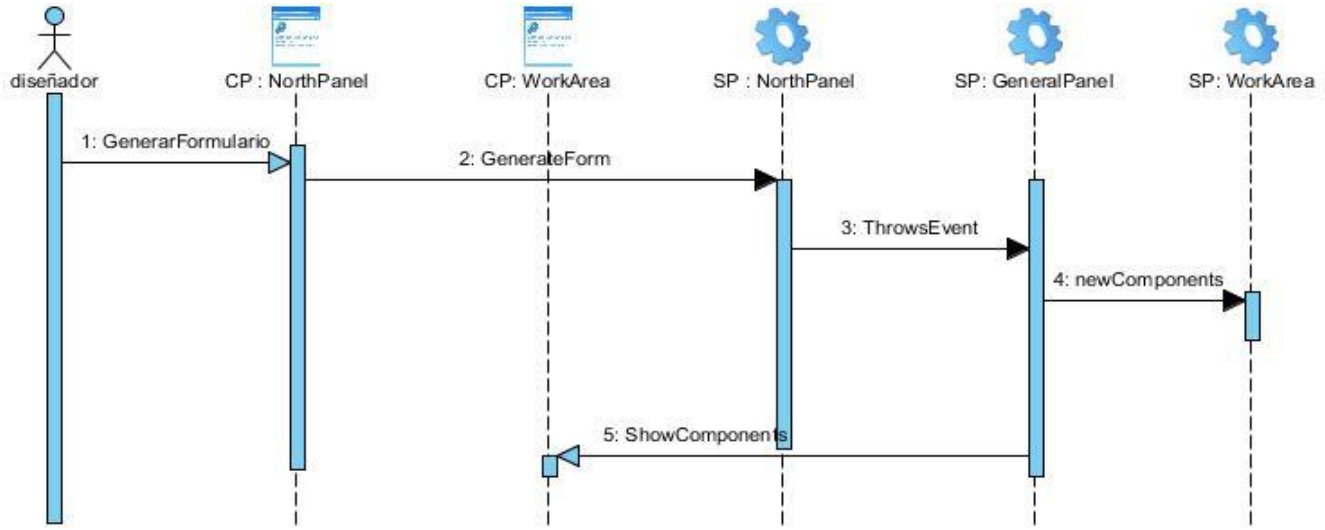


Figura 7 Diagrama de secuencia Generar formulario

El diseñador decide generar el formulario, y para realizar esta acción previamente tiene que haber seleccionado la entidad para la cual creará el formulario, luego la ClienPage NorthPanel le informa a la SP NorthPanel de la acción que se quiere realizar, esta última lanza el evento “generateform” a la GeneralPanel que se encarga de llamar a los componentes necesarios para agregar en el WorkArea, siendo los mismos los campos del formulario.

2.7.2 Patrones arquitectónicos utilizados

La arquitectura definida para los productos del centro, es la arquitectura basada en componentes; esta arquitectura identifica a los elementos principales como componentes del software y centra su atención en la integración de varios de ellos para conformar el sistema.

Para la implementación de la aplicación se define la arquitectura por capas. Este patrón genera una organización en cierta jerarquía permitiendo que la capa inferior provea a la capa superior de los servicios que necesite. Para la aplicación se hace uso de la arquitectura de 2 capas.

Capa de presentación:

Esta capa realiza el manejo de las interfaces y la interacción con los usuarios. Es la cara del sistema y su principal deber es lograr una buena comunicación entre el sistema y agentes externos, para ello debe ser completamente funcional.

Capa de negocio (capa lógica)

Esta capa la forman elementos que dentro del sistema se encargan de la automatización de los procesos de negocio realizado por los usuarios. Establece comunicación con la capa superior (capa de presentación), para recibir las solicitudes que son enviadas, y presentar los resultados a la capa superior.

2.7.3 Patrones de diseño utilizados

GASP (General Responsibility Assignment Software Patterns**)**

- **Alta Cohesión**: En la aplicación cada una de las clases se encuentra enfocada en lo que deben hacer, ya que les han sido asignados los métodos a las clases de forma completa y relacionada, lo que les ayuda a realizar una labor única.

- **Bajo Acoplamiento**: En la aplicación las clases tienen pocas dependencias entre ellas, por lo que existirá menos influencia en los cambios cuando estos se produzcan. En este caso la clase General Panel es la que se comunica con cada una de las clases existentes.
- **Creador:** En la aplicación para acceder a los distintos componentes y obtener las funciones que implementan, se crean instancias de ellos. La ventaja que presenta es que mantiene un bajo acoplamiento entre las clases de la aplicación.

GOF (Pandilla de los cuatro)

- **Observer** (Observador): Este patrón está presente en la implementación, en la cual la clase GeneralPanel es la encargada de notificar los eventos lanzados por las demás clases.

- **Singleton** (Instancia única): En la aplicación cada clase posee una instancia de ellas, lo que permite el trabajo con las funciones definidas en ellas en otras clases.

2.7.4 Modelo de Despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Este modelo es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre dichos elementos. Cada Hardware es representado como un nodo; un nodo es un elemento donde se ejecutan los componentes, entre ellos existen relaciones que son representados como medios de comunicación tales como HTTP, TCP/IP, etc.

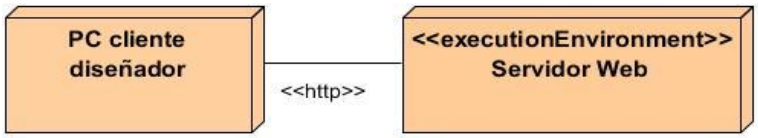


Figura 8 Diagrama de despliegue

2.8 Conclusiones

En este capítulo se definió el principal actor que interactúa con el sistema. Se mostró el modelo de dominio, para entender el sistema, se identificaron las funcionalidades que la aplicación debe poseer, así como las propiedades que el cómputo donde se instalará el sistema debe tener, representando los requisitos funcionales y no funcionales del sistema. Se realizó la descripción del CU exportar formulario representado en el diagrama de caso de uso. Se identificó el patrón arquitectónico a seguir para la implementación, se mostró el modelo de diseño, con sus respectivas clases de diseño y los diagramas de secuencia de una sección de la descripción del caso de uso exportar formulario.

3 IMPLEMENTACIÓN Y PRUEBA

Introducción

Este 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 unas series de código fuente de la construcción de la herramienta. Se valida la solución propuesta a través de pruebas que se realizan al código nombradas Caja Blanca.

3.4 Diagramas de Componentes

El diagrama de componente 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.

La siguiente figura es la representación del diagrama de componentes utilizado en la investigación, el diagrama está compuesto por archivos de extensión js, que responden al funcionamiento del lado del cliente con el uso de ExtJS, del lado del servidor, algunos archivos php necesarios para la crear el archivo zip. Más adelante se realiza la descripción de cada uno de los archivos representados.

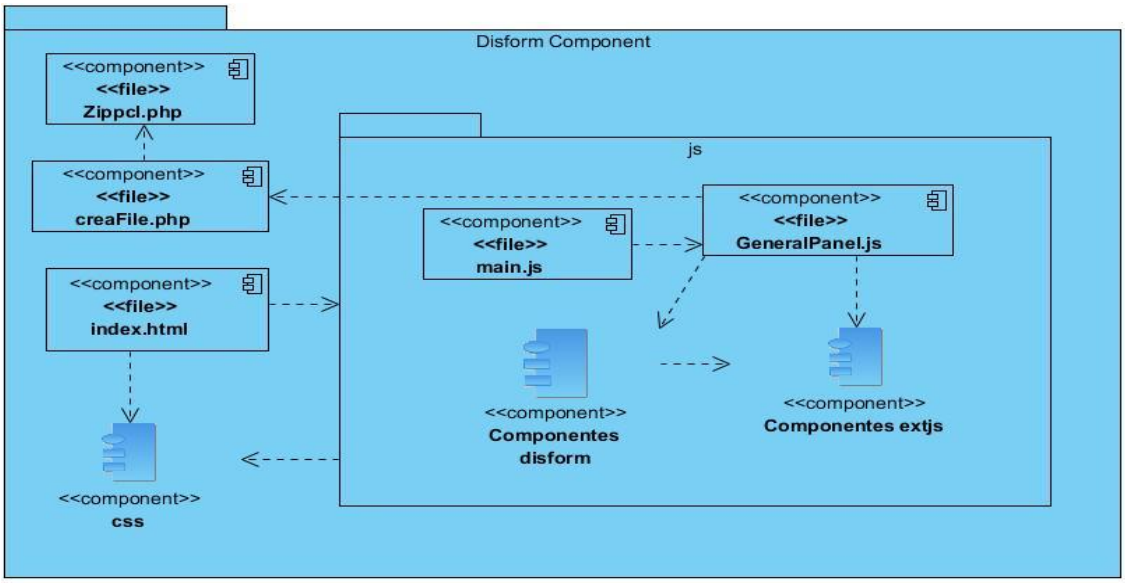


Figura 9 Diagrama de Componente.

Componente	Descripción
Zippcl.php	Es una clase php que permite realizar varias acciones con archivos zip, como crearlos y eliminarlos.
creaFile.php	Es una archivo php que obtiene el código enviado vía POST por la GeneralPanel y utiliza la clase Zippcl.php para crear el archivo zip del parámetro recibido.
index.html	Es la página principal que permite la visualización del diseñador y generador de formularios web.
css	Es un componente que permite la visualización de los iconos del diseñador, así como el color azul que presenta la interfaz.
main.js	Es la clase js encargada de inicializar el diseñador y generador de formularios web.
GeneralPanel	Es la clase encargada de ejecutar los eventos que son enviados desde las clases que integran el diseñador y generador de formularios web, estás son TabWorkArea, TreePanel, TreeArea, NorthPanel y GridPanel
Componentes disform	Estos componentes son las distintas clases que integran el diseñador y generador de formularios: TabWorkArea, TreePanel, TreeArea, NorthPanel, GridPanel, WindAdd y WindAddAtrib.
Componentes ExtJS	Son los diferentes componentes de ExtJS que fueron utilizados para la realización de la aplicación como los son: Panel, Window, Grid, TabPanel, Wiewport

Tabla 3: Descripción de los componentes definidos en la implementación.

3.5 Código fuente

3.5.1 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 tuvo en cuenta las siguientes reglas:

Estilo de indentación: es una regla que permite delimitar a través de llaves los bloques de códigos lógicos, lo que hace que dicho código sea más legible a los ojos de otros programadores. En la aplicación la indentación va a estar dado por el formato que define el IDE NetBeans, En caso de que una sentencia sea muy larga, se debe cortar la línea y comenzar en la siguiente, después de una coma u operadores lógicos (AND, OR).

Líneas en blanco: en ocasiones se hace necesario establecer visibilidad en el código fuente para mayor organización, las líneas en blanco ayudan a garantizar la estructuración del código, separando las sentencias que pueden pertenecer a clases, bloques, comentarios, etc. Para la implementación de la aplicación se ha decido utilizar esta regla.

Uso del SWITCH-CASE en lugar de sentencias IF-ELSE repetitivas: este estilo permite que dada una condición se puedan obtener respuestas dependiendo de los casos que estén involucrados, lo que provoca que el código sea óptimo. Para obtener la optimización del código de la aplicación se ha decidido utilizar el uso de switch-CASE.

Comentarios: los comentarios ayudan a entender al programador las secciones de código implementado y disminuye el esfuerzo del análisis. En ocasiones se hace necesario explicar que es lo que realiza una función, ya que por intervalos de tiempo el programador olvida que fue lo que pensó en un determinado momento.

Ejemplo de códigos de la aplicación

```
new Ext.form.TextField({
    id: 'tf',
    fieldLabel: 'Nombre',
    allowBlank : false,
    listeners:{
        //habilitar boton si hay algo escrito en el TextField
        'render': function(c){
            c.getEl().on('keyup', function(){
                Ext.getCmp('acept').setDisabled(false);

            },c);

        }
    },
    maxLength: 255,
    maxLengthText:'Este campo permite 255 caracteres.',
    blankText : 'Este campo es obligatorio.\n '

},
```

Figura 10 Código del evento asociar controles básicos

La figura anterior muestra un fragmento de código del evento asociar, capturado por la clase GeneralPanel (clase principal) que relaciona las entidades o atributos del TreePanel con un componente de los formularios, en este caso se está relacionando una entidad con el tipo de ventana que el usuario desea y para lograr hacer está asociación es necesario que se escriba en el textil, lo cual sería el nombre del contenedor a escoger, de esta manera se lanza la acción de activar el botón "Aceptar" para logar realizar dicha funcionalidad. En este fragmento se hace uso de los estilos de indentación, comentarios y línea en blanco.

```
switch(myChildNodes[j].atttype.toString()){
    case "1":

        this.componentes[this.posActual].controles.push({
            type: "textfield",
            fieldLabel: myComp.text,
            width: '40%',
            draggable: true,
            height: '15%',
            datalength: myComp.maxlenght,
            name: myComp.text,
            id: myComp.text+j
        });
        this.gp.reloadCombo(this.componentes[this.posActual]);

        this.twa.addTextField(myComp);

        break;

    case "2":

        this.componentes[this.posActual].controles.push({
            type: "textareafield",
            fieldLabel: myComp.text,
            width: '40%',
            height: '15%',
            name: myComp.text,
            id: myComp.text+j
        });
        this.gp.reloadCombo(this.componentes[this.posActual]);
```

Figura 11 Ejemplo de código del evento generar formulario

Este fragmento de código es parte del evento “generateform”, que es el encargado de generar el formulario después de haber creado la entidad y los diferentes atributos pertenecientes a la entidad ya asociados previamente con los componentes del formulario, para realizar esta función es necesario la utilización del SWITCH-CASE para la simplificación del código, ya que la funcionalidad depende de lo que el usuario elija como tipo de datos en la asociación previa, luego dichos componentes se crean en el área de trabajo y la propiedades de estos elementos son vistas en el GridPanel de propiedades.

3.6 Pruebas

Las pruebas del software es un elemento importante dentro del ciclo de vida de un producto, es sin lugar a dudas una de las formas más eficientes de garantizar la calidad del producto ya terminado. Las pruebas de software representan una revisión de las especificaciones del diseño y de la codificación, es una actividad en el cual el componente es ejecutado bajo ciertas condiciones específicas, los resultados observados son registrados y se realiza una evaluación del componente. Como parte de sus principales objetivos se encuentra

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta el momento.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces. (10)

Dentro de los tipos de pruebas que existentes se encuentran las pruebas de caja blanca, estas pruebas son un método que utiliza el diseño procedimental para obtener los casos de pruebas, los métodos que emplea analizan diferentes partes del producto y se complementan entre sí para garantizar la calidad del software. Formando parte de los tipos de prueba se encuentra las pruebas de caja negra, que se pueden realizar conociendo las funciones específicas para la que ha sido creado el software, de esta manera se buscan errores en el producto y demuestra si cada función es totalmente operativa o no. En la investigación se hace uso de las pruebas de caja blanca debido a que es la ideal para probar que el código no tiene errores, y es la indicada debido a que no existen entradas de datos del usuario para validar por pruebas de caja negra.

3.7 Diseño de Casos de Pruebas de Caja Blanca aplicados

A continuación se modela la prueba de camino básico aplicada a uno de los métodos o acciones de la clase GeneralPanel.js. Para realizar la prueba se tuvieron en cuenta una serie de pasos lógicos que define este tipo de prueba de caja blanca.

Pasos

1. Se construye el grafo a partir del código fuente del método a probar
2. Se determina la complejidad ciclomática del grafo V (G). La cual se puede calculada de varias formas.
   - V(G)= A-N +2, siendo A el número de aristas del grafo y N el número de nodos
   - V (G)= P+1, siendo P los nodos predicados (ponderados) del grafo.
   - Matriz de grafo.
   - Cantidad de regiones que componen el grafo.
3. Se construye una secuencia de caminos independientes en el grafo G.
4. Se preparan casos de pruebas por los caminos hallados en el paso anterior.
5. Se determinan los datos a proporcionar como entrada para ejecutar el camino hallado.

```
191         //generar formulario
192         this.np.on('generateform', function(){
193             var n = 0;
194             if(!this.selection.isLeaf())
195             {
196                 for (var i = 0; i < this.treedata.length; i++) {
197                     //obteniendo los nodos de la entidad seleccionada
198                     if(this.treedata[i].text == this.selection.text){
199                         var myChildNodes = this.datos.entidades[i].attributes;
200                         var myComp;
201
202                      //Agregando elementos al area de diseño
203                         var obj = {
204                             text: 'Formulario Generado',
205                             type: "panel"
206                         }
207
208                          //creando contenedor
209                         if(!this.componentes[this.posActual] || !this.componentes[this.posActual].cor
210                             var tipes='nodos\n \t';
211
212                             var ct = self.twa.addContenedor(obj);
213
214                             if(this.componentes[this.posActual]){
215                                 this.componentes[this.posActual].controles.push({
216                                     type: obj.type,
217                                     fieldLabel: obj.text,
218                                     layout:'absolute',
219                                     width: ct.getWidth(),
220                                     height: ct.getHeight(),
221                                     id: ct.getId()
```

```
222                                 });
223                             }
224                             else {
225                                 this.componentes.push({
226                                     type: obj.type,
227                                     title: obj.text,
228                                     width: ct.getWidth(),
229                                     height: ct.getHeight(),
230                                     id: this.selection.attributes.entityid,
231                                     controles: [{
232                                             type: obj.type,
233                                             fieldLabel: obj.text,
234                                             width: ct.getWidth(),
235                                             height: ct.getHeight(),
236                                             id: ct.getId()
237                                         }]
238                                 });
239                             }
240                             Ext.getCmp('botonGen').enable();
241                             this.gp.reloadCombo(this.componentes[this.posActual]);
242
243                             //adicionando componentes al contenedor
244                             for (var j = 0; j < myChildNodes.length; j++) {
245
```

```
245
246                     var Today = new Date();
247                     var ms = Today.getMilliseconds();
248
249                      myComp ={
250                          text: myChildNodes[j].attname,
251                          maxLenght: 30,
252                          draggable: true,
253                          id: myChildNodes[j].attname+j+ms
254                      };
255
256                      switch(myChildNodes[j].atttype.toString()){
257                          case "1":
258                              this.componentes[this.posActual].controles.push({
259                                  type: "textfield",
260                                  fieldLabel: myComp.text,
261                                  width: 150,
262                                  draggable: true,
263                                  height: 22,
264                                  datalength: myComp.maxLenght,
265                                  name: myComp.text,
266                                  id: myComp.id
267                              });
268                              this.gp.reloadCombo(this.componentes[this.posActual]);
269                              this.twa.addTextField(myComp);
270                              break;
```

```
271
272                         case "2":
273                             this.componentes[this.posActual].controles.push({
274                                 type: "textareafield",
275                                 fieldLabel: myComp.text,
276                                 width: 150,
277                                 height: 60,
278                                 name: myComp.text,
279                                 id: myComp.id
280                             });
281                             this.gp.reloadCombo(this.componentes[this.posActual]);
282                             this.twa.addTextArea(myComp);
283                             break;
284
285                         case "3":
286                              this.componentes[this.posActual].controles.push({
287                                 type: "radio",
288                                 fieldLabel: myComp.text,
289                                 width: 588,
290                                 height: '15%',
291                                 name: myComp.text,
292                                 id: myComp.id
293                             });
294                             this.gp.reloadCombo(this.componentes[this.posActual]);
295                             this.twa.addRadio(myComp);
296                             break;
```

```
297
298                      case "4":
299                          this.componentes[this.posActual].controles.push({
300                              type: "numberfield",
301                              fieldLabel: myComp.text,
302                              width: 150,
303                              height: 22,
304                              name: myComp.text,
305                              id: myComp.id
306                          });
307                          this.gp.reloadCombo(this.componentes[this.posActual]);
308                          this.twa.addNumberField(myComp);
309                          break;
310
311                      case "5":
312                          this.componentes[this.posActual].controles.push({
313                              type: "numberfield",
314                              fieldLabel: myComp.text,
315                              width: 150,
316                              height: 22,
317                              name: myComp.text,
318                              id: myComp.id
319                          });
320                          this.gp.reloadCombo(this.componentes[this.posActual]);
321                          this.twa.addNumberField(myComp);
322                          break;
```

```
323
324                                 case "6":
325                                     this.componentes[this.posActual].controles.push({
326                                         type: "datefield",
327                                         fieldLabel: myComp.text,
328                                         width: 150,
329                                         height: 22,
330                                         name: myComp.text,
331                                         id: myComp.id
332                                     });
333                                     this.gp.reloadCombo(this.componentes[this.posActual]);
334                                     this.twa.addDateField(myComp);
335                                     break;
336
337                                 case "7":
338                                     this.componentes[this.posActual].controles.push({
339                                         type: "checkboxgroup",
340                                         fieldLabel: myComp.text,
341                                         width: 588,
342                                         height: '15%',
343                                         name: myComp.text,
344                                         id: myComp.id
345                                     });
346                                     this.gp.reloadCombo(this.componentes[this.posActual]);
347                                     this.twa.addCheckBox(myComp);
348                                     break;
349
```

```
349
350                                 case "8":
351                                     this.componentes[this.posActual].controles.push({
352                                         type: "grid",
353                                         fieldLabel: myComp.text,
354                                         width: 150,
355                                         height: '15%',
356                                         name: myComp.text,
357                                         id: myComp.id
358                                     });
359                                     this.gp.reloadCombo(this.componentes[this.posActual]);
360                                     this.twa.addGrid(myComp.text);
361                                     break;
362
363                                 case "9":
364                                     this.componentes[this.posActual].controles.push({
365                                         type: "timefield",
366                                         fieldLabel: myComp.text,
367                                         width: 150,
368                                         height: 22,
369                                         name: myComp.text,
370                                         id: myComp.id
371                                     });
372                                     this.gp.reloadCombo(this.componentes[this.posActual]);
373                                     this.twa.addTimeField(myComp);
374                                     break;
```

```
375
376                                 case "10":
377                                     this.componentes[this.posActual].controles.push({
378                                         type: "combo",
379                                         fieldLabel: myComp.text,
380                                         width: 150,
381                                         height: 22,
382                                         name: myComp.text,
383                                         id: myComp.id
384                                     });
385                                     this.gp.reloadCombo(this.componentes[this.posActual]);
386                                     this.twa.addCombo(myComp);
387                                     break;
388                                 }
389                             }
390                             Ext.Msg.alert('Tipos de datos', tipes);
391                         }
392                         else{
393                             Ext.Msg.show({
394                                 title:'Error',
395                                 msg: 'Ya existe un contenedor asociado a esta &aacute;rea de tra
396                                 icon: Ext.MessageBox.ERROR,
397                                 buttons: Ext.MessageBox.OK
398                             });
399                         }
400
401                         Ext.MessageBox.show({
402                             title: 'Informaci&oacute;n',
403                             msg: 'Formulario generado correctamente.',
404                             icon: Ext.MessageBox.INFO,
405                                 buttons: Ext.MessageBox.OK
406                             });
407                             break;
408                         }
409
410                     }
411                 }
412                 else{
413                     Ext.Msg.alert('Seleccione una entidad');
414                 }
415
416                 },this)
```

Figura 12 Código del evento Generar formulario

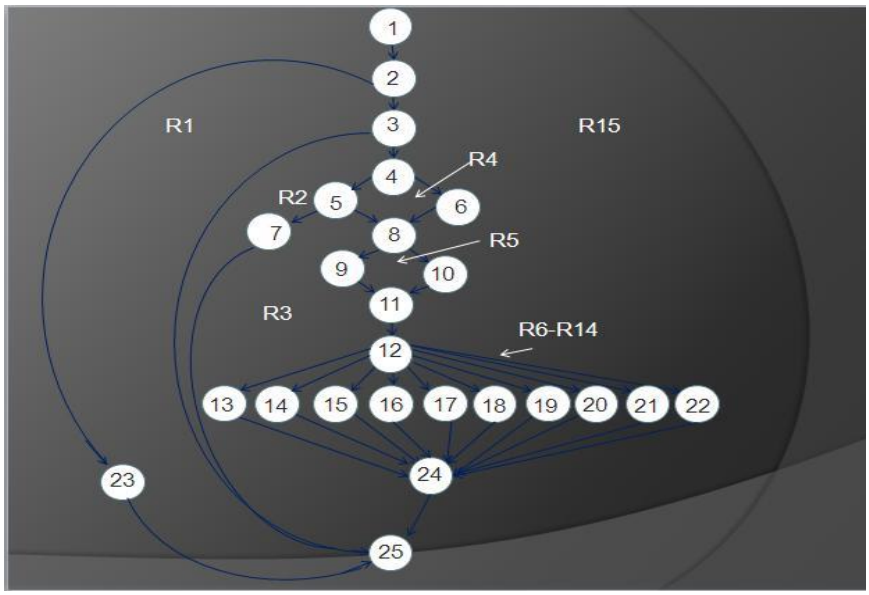


Figura 13 Grafo de Flujo

Fórmula	Valores	Resultado
V(G)= A-N+2	38-25+2	15
V(G)= P+1	14+1	15
V(G)= cantidad de regiones	R(i)	15

Tabla 4 Complejidad ciclomática

Matriz del grafo.

Con la matriz se obtienen lo nodos predicados del grafo, sumando el número de conexiones de un nodo hacia otros nodos del grafo, la sumatoria de cada resultado se le resta 1, y de esta forma se obtienen la complejidad ciclomática del grafo.

Conjunto básico de caminos linealmente independientes.

El valor que se obtiene de V (G) es el número de caminos lineales independientes de la estructura. En el caso del código en cuestión obtenemos 15 caminos independientes.

1-2-23-25

1-2-3-25

1-2-3-4-5-7-25

1-2-3-4-5-8-9-11-12-13-24-25

1-2-3-4-5-8-9-11-12-14-24-25

1-2-3-4-6-8-10-11-12-13-24-25

1-2-3-4-6-8-10-11-12-14-24-25

1-2-3-4-6-8-10-11-12-15-24-25

1-2-3-4-6-8-10-11-12-16-24-25

1-2-3-4-6-8-10-11-12-17-24-25

1-2-3-4-6-8-10-11-12-18-24-25

1-2-3-4-6-8-10-11-12-19-24-25

1-2-3-4-6-8-10-11-12-20-24-25

1-2-3-4-6-8-10-11-12-21-24-25

1-2-3-4-6-8-10-11-12-22-24-25

Caso de prueba del camino 1

Valor (k) seleccionado= nodo hoja (true)

Resultados esperados: no se genera formulario y se envía una alerta al usuario.

Caso de prueba del camino 2

Valor (k) seleccionado= nodo hoja (false) y el texto de la selección no coincide con el texto que se encuentra en el árbol.

Resultados esperados: no generar formulario.

Caso de prueba del camino 3

Valor (k) seleccionado= nodo hoja (false), el texto de la selección coincide con el texto que se encuentra en el árbol, y busca en el listado de componentes, si se encuentra en esa lista, en caso de ser verdadero

Resultados esperados: mostrar ventana de error.

Caso de prueba del camino 4

Valor (k) seleccionado= nodo hoja (false), el texto de la selección coincide con el texto que se encuentra en el árbol, y busca en el listado de componentes, si no se encuentra en esa lista, se habilita botón generar formulario, se elige el textfield.

Resultados esperados: formulario generado con un textfield como elemento del mismo.

Casos de pruebas del camino 5 al camino 15

En los caminos siguientes al cuarto ocurre similar, se elige otro elemento del formulario o todos los elementos que están definidos para generar el formulario.

Resultados esperados: formulario generado.

3.8 Comparación entre la v1.0 de la v2.0 del diseñador y generador de formularios web.

Con la primera versión del Disform es posible diseñar el formulario con los componentes textarea, textfield, integer y radiobutton, además de editar las propiedades de los mismos y mostrar el código del diseño realizado, aunque el componente textarea no era mostrado en la visualización del código, los contenedores del formulario no se podía editar la propiedad título y el radiobutton, no podía cambiar el de tamaño por el fieldset que lo contiene.

En la nueva versión del Disform 2.0, se puede diseñar los formularios con nuevos componentes, editar las propiedades de todos los elementos definidos, generar el código de los mismos, así como exportar el código como un archivo js, generar el formulario después de creada la entidad con sus atributos, eliminando pasos como el asociar atributos que se utiliza en el diseñar, lo cual trae como ventaja que se realice el diseño del formulario con mayor rapidez. Además se valida que el usuario elija las opciones indicadas habilitando e inhabilitando los botones de las diferentes acciones a realizar, de esta forma se asegura que el usuario no cometa errores.

En la siguiente tabla se muestran las características de cada una de las versiones.

Funcionalidades	Versión 1	Versión 2
Adicionar entidad	x	x
Eliminar entidad	x	x
Adicionar atributo	x	x
Eliminar atributos	x	x
Diseñar formulario	x	x
Generar formulario		x
Exportar formulario como archivo js		x
Adicionar y eliminar Checkbox		x

Adicionar y eliminar Radiobutton		x
Adicionar y eliminar TextField	x	x
Adicionar y eliminar Date		x
Adicionar y eliminar TimeFlied		x
Adicionar y eliminar NumberFiled	x	x
Adicionar y eliminar Grid		x
Adicionar y eliminar TextArea	x	x
Adicionar y eliminar Combobox		x
Generar código	x (con problemas)	x

Tabla 5: Tabla comparativa entre las versiones del Disform

3.9 Conclusiones

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 los estándares de programación para el código fuente de la aplicación. Se realizaron las pruebas de caja blanca, camino básico para validar los métodos implementados, en el documento se muestra el código del generar formulario en el cual se obtuvieron 15 caminos lineales independientes y se realizan los casos de pruebas pertinentes. Se realiza una comparación entre las dos versiones del diseñador y generador de formularios web como forma de apreciar mejor los cambios ocurridos de una versión a otra.

CONCLUSIONES GENERALES

- Se identificaron las funcionalidades a realizar a través de los requisitos funcionales de la aplicación web. Se realizó el diseño utilizando patrones que contribuyen a facilitar el proceso de implementación.
- Se desarrolló las funcionalidades de la aplicación web que garantizan la generación de formularios automáticamente.
- Se realizó las pruebas de caja blanca al código fuente de la aplicación web.

RECOMENDACIONES

- Se implemente importar formulario, que han sido exportados.
- Que se realice el enlace del diseñador con la base de datos para que el modelo relacional se evidencie.

Se continúe perfeccionando el diseñador y generador de formulario en próximas versiones.

REFERENCIAS BIBLIOGRÁFICAS

1Suite Monica Miranda*Suite Monica Miranda*http://monica-miranda.suite101.net/que-son-las-nuevas-tecnologias-a18755

2Delta Desinghttp://www.ddelta.com.ar/solutions/appWeb.aspx

3Wiziq*Wiziq*http://www.wiziq.com/tutorial/107690-Captura-de-datos-presentacion

4Deperu.com*Deperu.com*http://www.deperu.com/abc/diseno/269/el-diseno

5Techtastico*Techtastico*http://techtastico.com/post/%C2%BFque-es-el-diseno-web/

6*Diccionario de la Informática*Ciudad de La HabanaCientífico-Técnica2005

7bloggerandreinabloggerhttp://bloggerandreinablogger.blogspot.com/

8Kumbia php frameworkScribd*Scribd*Scribd.INChttp://es.scribd.com/doc/71448152/177/Ventajas-Generadores-de-Formularios

9alzado.org*alzado.org*http://www.alzado.org/articulo.php?id_art?=221

10*Ingeniería del Software*2005

11*Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*Bueno Aires2004versión 1.0

12*UML y Patrones*La HabanaFélix Varela2004Tomo I y II

13*El Proceso Unificado de Desarrollo de Software. 2000.*2000

14CBASQA-Desarrollo de Software, SQA.*CBASQA-Desarrollo de Software, SQA.*http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/

15ECURED*ECURED*http://www.ecured.cu/index.php/Patrones_de_Casos_de_Uso

16Ecured*Ecured*http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodologica.html

17*Ayuda OpenUp.*

18 Object Management Group. http://www.uml.org.

BIBLIOGRAFÍA

Albornoz L., Francis G. Diseño de formularios por computadora. *Diseño de formularios por computadora.* [En línea] http://www.monografias.com.

Ayuda OpenUp. 2006.

Carlos Indriago, Enmanuel González. Wiziq. *Wiziq.* [En línea] http://www.wiziq.com/tutorial/107690-Captura-de-datos-presentacion.

Carlos Reynoso, Nicolás Kicillof. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Bueno Aires : s.n., 2004. versión 1.0.

CBASQA-Desarrollo de Software, SQA. [En línea] [Citado el: 20 de noviembre de 2010.]. CBASQA-Desarrollo de Software, SQA. *CBASQA-Desarrollo de Software, SQA.* [En línea] 02 de septiembre de 2008. [Citado el: 18 de abril de 2012.] http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.

Cháves, Carlos Alberto García. Mailxmail. [En línea] 13 de 04 de 2005. [Citado el: 07 de mayo de 2012.] http://www.mailxmail.com/curso-diseno-base-datos-relacionales/sistemas-bases-datos.

Delta Desing. [En línea] [Citado el: 05 de 05 de 2012.] http://www.ddelta.com.ar/solutions/appWeb.aspx.

Deperu.com. *Deperu.com.* [En línea] [Citado el: 05 de 05 de 2012.] http://www.deperu.com/abc/diseno/269/el-diseno.

Diseñadores Web Profesionales | Diseño Gráfico y web, Tecnología e Internet | ProfesionalNe. *Diseñadores Web Profesionales | Diseño Gráfico y web, Tecnología e Internet | ProfesionalNe.* [En línea] 26 de 03 de 2009. [Citado el: 10 de 04 de 2012.] http://profesionalnet.wordpress.com/category/aplicaciones-ria/.

ECURED. *ECURED.* [En línea] [Citado el: 13 de 05 de 2012.] http://www.ecured.cu/index.php/Patrones_de_Casos_de_Uso.

Ecured. *Ecured.* [En línea] [Citado el: 17 de abril de 2012.] http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodologica.html.

formularios, Control de. bloggerandreinablogger. [En línea] 16 de abril de 2009. [Citado el: 5 de 05 de 2012.] http://bloggerandreinablogger.blogspot.com/.

Bibliografía

Grady, Booch, Ivar, Jacobsony James, Rumbaugh . *El Proceso Unificado de Desarrollo de Software. 2000.* 2000.

Group., Object Management. Object Management Group. . [En línea] 1997. [Citado el: 04 de diciembre de 2011.] http://www.uml.org.

http://jsr.germinus.com/inicio. [En línea]

IDESA. *IDESA.* [En línea] [Citado el: 04 de diciembre de 2011.] http://www.idensa.com/. Inteligencia de Negocio.

Ivar Jacobson, Grady Booch, James Rumbaugh. El Procesos Unificado de Desarrollo de Software.

Junoy, Josep M. alzado.org. *alzado.org.* [En línea] 22 de diciembre de 2004. [Citado el: 07 de mayo de 2012.] http://www.alzado.org/articulo.php?id_art?=221.

Kiccillof, Carlos Reynoso y Nicolás. *Estilos y patrones en la Estrategía de Arquitectura de Microsoft.* BUenos Aires, Argentina : Universidad de Buenos Aires : s.n., 2004., 2004.

Kumbia php framework. Scribd. *Scribd.* [En línea] Scribd.INC, 2009. [Citado el: 18 de abril de 2012.] http://es.scribd.com/doc/71448152/177/Ventajas-Generadores-de-Formularios.

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

Leopordo, Carlos. Techtastico. *Techtastico.* [En línea] 22 de 03 de 2006. [Citado el: 10 de 04 de 2012.] http://techtastico.com/post/%C2%BFque-es-el-diseno-web/.

Lobo, Armando Robert. *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* Ciudad Habana : s.n., 2009. Tesis.

Mabel Navarro Bermúdez, Yissel Rodríguez. *BioSyS: Implementación del Módulo de Simulación.* Ciudad de La Habana : Universidad de la Ciencias Informáticas., 2008.

Microsoft. office.microsoft.com. */office.microsoft.com.* [En línea] [Citado el: 20 de 05 de 2012.] http://office.microsoft.com/es-mx/training/los-formularios-desempenan-una-util-funcion-RZ001027863.aspx?section=3 .

ANEXOS

Anexo 1:

Directrices de usabilidad para el diseño de formularios.

Shneiderman proporciona la siguiente lista de directrices de diseño:

- Dar un título al formulario que exprese claramente su función.
- Las instrucciones han de ser breves y comprensibles.
- Hacer grupos lógicos de campos y separarlos con blancos. Por ejemplo: nombre, primer apellido y segundo apellido es un grupo lógico.
- Aspecto ordenado, alineando los campos y las etiquetas.
- Las etiquetas de los campos deben usar terminología familiar.
- Ser consistente en el uso de los términos, es decir, usar siempre las mismas palabras para los mismos conceptos.
- El tamaño visible del campo debe corresponderse con la longitud del contenido que ha de introducir el usuario.
- Permitir el movimiento del cursor por medio del teclado y no solo con el mouse.
- Permitir que el usuario pueda corregir con libertad los caracteres que ha introducido en los campos.
- En donde sea posible, impedir que el usuario introduzca valores incorrectos. Por ejemplo, impedir que introduzca caracteres alfabéticos en campos que solo admiten valores numéricos.
- Si introduce valores incorrectos, indicar en un mensaje cuales son los correctos.
- Avisar cuanto antes al usuario si ha introducido valores incorrectos. Si es posible no esperar a que haya rellenado el formulario totalmente.
- Marcar claramente los campos opcionales.
- Si es posible, colocar explicaciones o listas de los valores válidos al lado de los campos
- Dejar clara la acción que debe hacer el usuario al terminar de rellenar el formulario.

Anexos 2

Caso de Uso:	**Administrar controles básicos**
Actores:	**Diseñador**
Resumen:	El caso de usos comienza cuando el diseñador decide realizar algunas de las siguientes acciones. **Asociar control básico a atributo:** el diseñador decide adicionar un control básico como textarea, textfield, checkbox, radiobutton y grid en el área de trabajo. **Eliminar control básico**: el diseñador decide borrar algunos de los controles básicos, como: checkbox, textfield, radiobutton, grid, timefield
Precondiciones:	Haber creado una entidad a la que se le agregarán los atributos necesarios para poder administrar los controles básicos
Referencias	RF 2.1, RF 2.2, CU incluido
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
El diseñador decide realizar alguna de estas acciones: • Asociar control básico a atributo. • Eliminar control básico.	El sistema muestra una de las interfaces en correspondencia a la opción seleccionada. • Asociar control básico a atributo. Ir a la sección “Asociar control básico”. • Eliminar control básico. Ir a la sección “Eliminar control

	básico”.
Sección “Asociar control básico a atributo”	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra la interfaz con los tipos de controles básicos existentes en la aplicación.
2. El diseñador elige el tipo de control básico.	3. El sistema guarda el tipo especificado. Terminando así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Sección “Eliminar control básico”	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema lanza un mensaje de verificación de la eliminación del control básico.
2. El diseñador acepta eliminar el control básico.	3. El sistema realiza la eliminación. Terminando así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

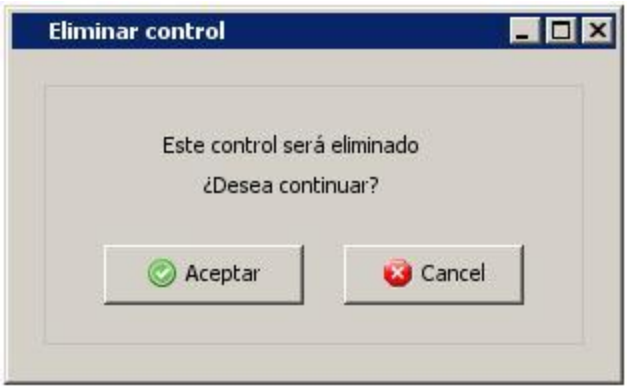
2a. El diseñador cancela la operación.	2b. El sistema muestra la interfaz anterior
Prototipo de Interfaz	
	
Poscondiciones	Control básico eliminado

Tabla 6: Descripción del caso de uso administrar controles básicos

Caso de Uso:	**Editar propiedades**
Actores:	**Diseñador**
Resumen:	El caso de usos comienza cuando el diseñador decide realizar algunas de las siguientes acciones. **Editar propiedades de control básico:** el diseñador decide cambiar las propiedades que trae por defecto el control básico, como checkbox, radiobutton, textfield, textarea, datefiel, timefield, numberfield.
Precondiciones :	Haber creado área de trabajo, entidades y atributos

Referencias	RF 3. CU extendido
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
El diseñador decide realizar alguna de estas acciones: • Editar propiedades de control básico.	El sistema muestra una de las interfaces en correspondencia a la opción seleccionada. • Editar propiedades de control básico. Ir a la sección "Editar propiedades de control básico".
Sección "Editar propiedades de control básico"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra las propiedades del control básico.
2. El diseñador cambia las propiedades.	3. El sistema muestra el control básico con nuevas propiedades. Terminando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

Tabla 7: Descripción del caso de uso Editar propiedades

GLOSARIO DE TÉRMINOS

Asíncrono: adjetivo que califica los eventos que ocurren sin una relación predecible o regular en el tiempo.

AJAX: JavaScript y XML Asíncrono.

Aplicaciones RIA: Aplicaciones Enriquecidas para la web.

Ciclo de vida del software: proceso por el cual deben atravesar todos los productos de software: análisis, diseño, implementación, prueba, despliegue y mantenimiento.

Código fuente: representan instrucciones contenidas en un programa y entendibles por las computadoras.

CU: abreviatura de casos de uso.

Disform: diseñador y generador de formularios web.

Formulario: es una plantilla o página con espacios vacíos que han de ser rellenados con alguna finalidad, por ejemplo datos de contactos.

Framework: es una estructura de soporte definida en la cual se pueden generar otras aplicaciones a partir de ella, puede incluir herramientas de diseño, lenguajes de programación, bibliotecas de clases y compiladores.

Navegador: es una aplicación que permite visualizar información que contiene sitios y páginas web.

PADTSI: Paquete de herramientas de Ayuda para la Toma de Decisiones y Soluciones Integrales.

RNF: Requisitos nos funcionales.

RF: Requisitos funcionales

SIGE: Sistema Integral de Gestión Estadística.

Widgets del formulario: muestran información, por ejemplo cuadro de texto, o cuadro de imagen, a los que también se les llama campo de formulario. De igual forma se les llama widgets del formulario a los controles del formulario.

Modelo relacional: permite representar la información del mundo real, introduciendo conceptos cotidianos y fáciles de entender por cualquier inexperto

Diccionario de datos: contiene datos referentes a los propios datos contenidos en la base de datos incluyendo información acerca de los registros y tipos de campos.

SP: SeverPage (página servidora)