Universidad de las Ciencias Informáticas

Facultad 6

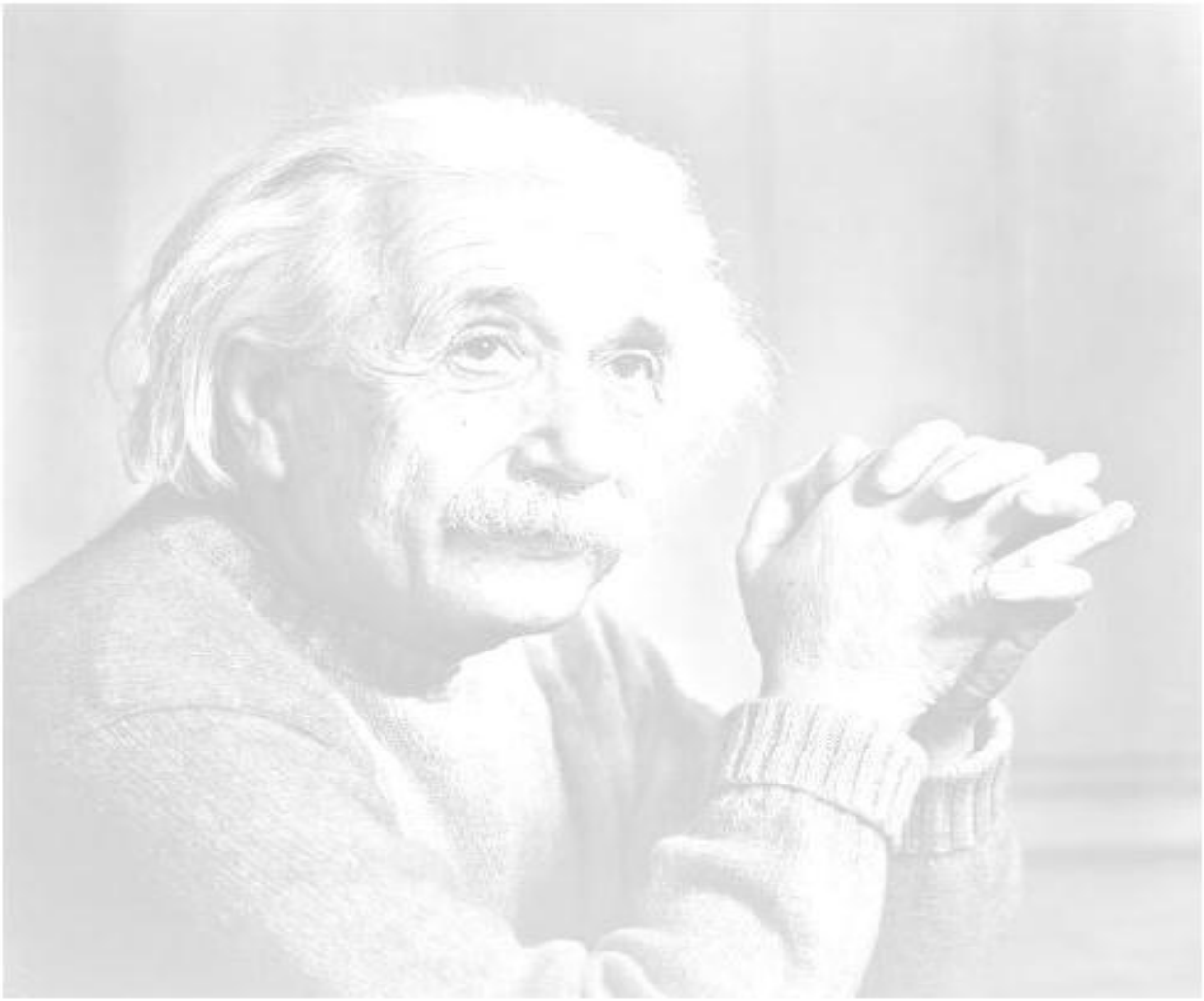
TÍTULO: Desarrollo del diseñador y generador de formularios web a partir de un modelo relacional

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

Autor: Yasniel Montano Rodríguez

Tutor(es): Ing. Héctor Luis Reyes Zaldívar

Ing. Alberto Mendoza Garnache



¿Por qué esta magnífica tecnología científica, que ahorra trabajo y nos hace la vida más fácil, nos aporta tan poca felicidad? La repuesta es está, simplemente: porque aún no hemos aprendido a usarla con tino.

Albert Einstein.

DECLARACIÓN DE AUTORÍA

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

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

Yasniel Montano Rodríguez

_____

Firma del Autor

Héctor Luis Reyes Zaldívar

_____

Firma del Autor

Alberto Mendoza Garnache

_____

Firma del Tutor

DATOS DE CONTACTO

Autor:

Yasniel Montano Rodríguez

Email: ymontano@estudiantes.uci.cu

Tutores:

Héctor Luis Reyes Zaldívar

Ingeniero en Ciencias Informáticas

Email: hlreyes@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Alberto Mendoza Garnache

Ingeniero en Ciencias Informáticas

Email: agarnache@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

AGRADECIMIENTOS

Quiero agradecerle en primer lugar a mi estrella guía, la que con su luz ilumina todos mis caminos, por siempre estar ahí cuando más solos nos sentimos, por hacer de mí por encima de cualquier circunstancia un hombre de bien, de buenos principios y costumbres. Gracias por siempre creer en mí, por toda esta vida de cariño y de enseñanzas, en ningún lugar del mundo encontraré el amor que solo tú me has dado. Gracias a ti mamá por amarme tanto por ser mi razón de ser, si hoy estoy aquí es gracias a ti. Por eso para ti el resultado de tanto estudio durante todo este tiempo, te amo.

A todos mis hermanos en especial a mi Tuti y a Sady por todo su apoyo, dedicación y aliento, por ese cariño incomparable que me llena de vida y me da fuerzas para seguir adelante, hoy le doy gracias a la vida por tenerlas a mi lado, por ser parte de mi inspiración, nada cambiará lo que siento hoy por ustedes.

Para ti son estas sencillas líneas, si hay alguien que tengo que agradecerle en esta vida es a ti, gracias Migue por ser más que cuñado un padre para mí, el que con su cariño y respeto logró relevar el amor de un padre que tanto necesitaba, por ser mi ejemplo a seguir, por amar y defender a mi hermana por encima de cualquier cosa, gracias por todos los regaños, consejos, por la educación, por sembrar valores en mí desde niño que hoy me hacen ser el hombre que soy, agradecer por ser parte de todos los momentos felices de mi vida, gracias por siempre estar ahí, de corazón te digo que no tengo en la vida como pagar todo lo

que has hecho por mí, espero que este sea uno de los frutos que con amor supiste sembrar.

Mi tati no he conocido nadie en la vida que tan solo con una pizca de cariño pueda llenarme de tantas alegrías, a ti Ayrin gracias por todo este tiempo juntos, por tu amor incomparable, por tu apoyo, dedicación, por mimarme como si fuera un niño cuando más mal me sentía, por tantos años de compresión y dulzuras a mi lado, por creer en mi a pesar de lo difícil que fuera el camino, gracias por dejarme ser de ti el hombre de tu vida, para ti la madre de mis hijos gracias por ser parte de mi inspiración, te amo.

A Betty, Alain, Rolito, Sandra y Rolando por tantos regaños, consejos, por siempre estar al tanto de mi queriéndome como un hijo más de ustedes. Gracias por darme la oportunidad de ser parte de esa linda familia, a todos le agradezco por todo los buenos momentos que hemos pasados juntos.

Bueno en general a toda mi familia no menciono nombre para que no se me quede nadie pero si a mi prima Solaris que en sus años de vida siempre deseó ver este momento.

A Tico porque más que un tutor fue un amigo para mí, gracias por su ayuda incondicional cuando más difícil se me hacía terminar, gracias por su apoyo, sus enseñanzas y por todo el momento que compartimos juntos.

A mis amistades son muchos y es una pena que alguien se me quede, pero bueno gracias a todos por ser parte de mi vida, en especial a Ale, Hismel, Jorge, Yunior,

al Yaspa, Yosbel, a Oni, a Rey, al Yoyo y a Juan Carlos Moya. Gracias a todos en general por siempre estar ahí cuando más lo necesitaba, por ser parte de tantos momentos alegres y tristes de mi vida.

A todos los quiero.

DEDICATORIA

Con todo mi amor y cariño a mi Madre, mis hermanas y mi novia.

RESUMEN

En el presente trabajo se pretende crear la primera versión del Diseñador y Generador de formularios web para el Sistema de Información de Gobierno (SIGOB) que pertenece al Centro de Tecnología de Gestión de Datos (DATEC) ubicado dentro de la Universidad de las Ciencias Informáticas (UCI). El software de diseño y generación de formularios web implementado por SIGOB surge como respuesta a la necesidad de agilizar el proceso de captura de datos en el país. Como apoyo a esta necesidad se implementó un componente que diseña y genera formularios web a partir de un modelo relacional. Como base de este proceso, previamente fue realizado un amplio estudio de los conceptos implicados en el negocio y un diseño adecuado a las tecnologías y herramientas propuestas por el grupo de arquitectura de SiGOB. Como resultado se obtuvo un componente que contribuye con la aceleración y humanización del desarrollo de formularios basado en ExtJS mediante la generación automática de código de JavaScript. Durante el desarrollo de este componente se realizaron las pruebas funcionales lo que garantiza su perfecto funcionamiento.

PALABRAS CLAVE

Diseñador, DATEC, formularios web, generador, SIGOB.

INDICE

ÍNDICE DE FÍGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las Tecnologías de Información y las Comunicaciones (TICs) agrupan los elementos y las técnicas utilizadas en el tratamiento de las informaciones, principalmente de informática, internet y telecomunicaciones. Estas garantizan en gran medida los procesos que se desarrollan en la sociedad y en la economía de los diferentes países en el presente siglo, de manera que su uso es cada vez mayor con una importancia creciente. Estas tecnologías se adentran de manera peculiar en áreas muy importantes a nivel empresarial, posibilitando que los procesos de negocio sean más eficientes y rápidos al establecerse para sistemas de software que ayudan a la gestión de información en las empresas.

Los sistemas electrónicos son aquellos programas de desarrollo tecnológico que permiten a los clientes que a través de su equipo de cómputo consuman los servicios que proporcionan las empresas y realice las consultas que brinda la aplicación. Dichos sistemas son creados para mejorar las características productivas de una entidad, de esta manera gestionar y explotar la información de las instituciones para las cuales han sido creadas, así como presentar dentro de sus funciones particulares, un modelo de captura de datos eficiente.

La captura de datos puede tornarse en ocasiones muy compleja, cuando no se dispone de herramientas eficaces que logren gestionar la tarea sin demoras y le proporcionan al usuario de un producto final donde se pueda apreciar lo que en realidad él necesita.

Para realizar dicha tarea son muy populares los formularios, por lo que existen disímiles herramientas encargadas de realizar dentro de sus procesos la captura de datos utilizando los mismos. Para ayudar a que esta captura sea más eficiente y menos compleja, en la actualidad se han venido desarrollando los diseñadores y generadores de formularios web que dotan a los sistemas de nuevas funcionalidades que dan la oportunidad al usuario de crear formularios y generarlos automáticamente.

En la actualidad resulta vital para los sistemas que existen en muchas empresas, contar con componentes que sean capaces de diseñar y generar formularios web. Sin este tipo de tecnologías, la captura de información resultaría difícil para lograr agilizar la productividad en las entidades que estén involucradas en dicho proceso. Después de consultar un conjunto de bibliografías se pudo constatar que en el mundo existen diseñadores y generadores de formularios web, pero estos no responden a

las necesidades de un modelo relacional que pueda ser utilizado por el Sistema de Información de Gobierno (SIGOB) en el país.

En Cuba, los formularios como forma de captura de información son muy utilizados por diferentes sectores económicos y sociales, presentando la cara del sistema e interactuando de forma fácil con usuarios no identificados rigurosamente con la informática.

En el país no se conoce de una implementación anterior de un sistema que dentro de sus funcionalidades cuente con módulos para el diseño y generación de formularios web que pueda vincularse con las particularidades de cada entidad. Por lo que en muchas ocasiones se hace difícil utilizar software que no sean de libre acceso.

La Universidad de las Ciencias Informáticas cuenta con varios centros vinculados directamente a la producción de software. En la facultad 6 se encuentra el Centro de Tecnología de Gestión de Datos, dicho centro está constituido por cuatro líneas fundamentales de producción, y a cada una de ellas se le asignan un conjunto de proyectos para desarrollar, basado en los principios de la calidad total y mejora. Una de estas es la línea de Soluciones Integrales en la cual se encuentra en desarrollo un proyecto de gran importancia para el país, SIGOB, concebido como activo de software reutilizable de la plataforma tecnológica especializada del centro, por ser el componente que se utiliza para la creación y configuración de otras herramientas con las características del cliente. Como producto destaca la integración en él del Paquete para la Ayuda de Toma de Decisiones (PATDSI), el cual contiene el Sistema Integral de Gestión Estadística (SIGE), que en el contexto de SIGOB asume la función de captura de información; y el Sistema de Encuestas, con un objetivo similar al de SIGE.

Como se ha mencionado anteriormente, para el SIGOB se hace necesario integrar todo tipo de procesos que se estén desarrollando. La confección de un módulo para el diseño y generación de formularios web responde a la no existencia en Cuba de una herramienta que se integre a las necesidades de DATEC.

El estudio de la problemática anterior desencadena el siguiente **problema científico:** Las insuficiencias en el diseño y generación de formularios web está afectando el proceso de captura de datos en el Sistema de Información de Gobierno.

Para llevar a cabo la investigación se ha definido como **objeto de estudio**: Los componentes de software para el diseño y generación de formularios web.

Enmarcando el objeto de estudio se define como campo de acción: Los componentes de software para el diseño y generación de formularios web a partir de un modelo relacional.

El objetivo general de la presente investigación es: Desarrollar un diseñador y generador de formularios web a partir de un modelo relacional para el Sistema de Información de Gobierno.

Para alcanzar el objetivo trazado se definen los siguientes **objetivos específicos**:

- Realizar el marco teórico de la investigación.
- Realizar análisis y diseño del componente diseñador y generador de formularios web a partir de un modelo relacional para SIGOB.
- Implementar el componente diseñador y generador de formularios web a partir de un modelo relacional para SIGOB.
- Realizar pruebas a nivel de desarrollador al componente diseñador y generador de formularios web a partir de un modelo relacional para el SIGOB.

Se definen las siguientes tareas de investigación para dar cumplimiento a los objetivos específicos planteados anteriormente:

- Investigación de las tecnologías actuales de diseñadores y generadores de formularios web.
- Análisis de los principales componentes de los formularios web.
- Definición de los requisitos funcionales y no funcionales del componente a implementar.
- Diseño del componente diseñador y generador de formularios web en su versión 1.0.
- Implementación del componente diseñador y generador de formularios web.
- Realización de las pruebas a nivel de desarrollador al componente obtenido.

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

Capítulo 1: Fundamentación teórica. Se enuncian las principales definiciones relacionados con la captura de datos a través de diseñadores y generadores de formularios web. Se realiza un estudio de algunas soluciones existentes, concluyendo con la selección del mecanismo más indicado para desarrollar el componente.

Capítulo 2: Análisis y diseño. En este capítulo se hace una descripción del sistema a automatizar, en el cual se mostrará un diagrama de casos de uso del sistema para su mejor entendimiento. Se

definirán los actores del sistema, así como los requisitos funcionales que el sistema debe poseer y los requisitos no funcionales mínimos que debe tener la computadora 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 para cada caso de uso, así como los diagramas de secuencia por escenario de los mismos.

Capítulo 3: Implementación y Pruebas. En este capítulo se realizan las descripciones de las clases y componentes que se generan en la implementación del sistema, mostrándose el modelo de implementación, despliegue y se brinda una solución a los requisitos identificados. Se realizan pruebas a nivel de desarrollador para garantizar la calidad del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

Este capítulo muestra una panorámica de los diseñadores y generadores de formularios web referentes al proceso de captura de datos, así como la filosofía de trabajo que establecen los mismos y sus principales ventajas, además una detallada descripción de la metodología, herramientas y tecnologías utilizadas para una mejor comprensión y dar solución al problema científico planteado.

2.1 Conceptos básicos relacionados con los diseñadores y generadores de formularios web

Las aplicaciones web son muy utilizadas por usuarios, por las facilidades que brindan, una de ella es que con solo tener el nombre de la página que se desea usar puedes obtener servicios de varios tipos.

Según el grupo de Ingeniería de Software de la Universidad de Sevilla: una aplicación web, es una herramienta informática distribuida cuya interfaz de usuario es accesible desde un cliente web, normalmente un navegador web. (1)

Los servicios que una aplicación web puede brindar son implementados en la mayoría de los casos a través de **formularios** que como se ha visto anteriormente son utilizados para poder establecer la captura de información con más precisión. Según los autores Noel Luis Núñez Camallea y Ronald Coutin Abalo quienes definen formulario en su libro Diccionario de Informática 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. (1)

En la mayoría de las aplicaciones, los formularios forman parte de la interfaz, permitiéndoles a los usuarios introducir datos. Por la importancia que representan los formularios y el uso que se les puede dar, es necesario el diseño de los mismos, ya que la mala concepción de esta función puede situarse como una barrera para la iteración e inducir al usuario a cometer errores potenciados por la frustración en el desempeño de sus tareas.

El diseño es el proceso previo de configuración mental en la búsqueda de una solución en cualquier campo. Lo que implica que no cualquiera pueda realizar el diseño. Los diseñadores son las personas que profesionalmente se dedican a realizar esta tarea.

El concepto de diseño se aplica a diferentes acepciones en la informática, está el caso del **diseño web** que es el arte de planificar, diseñar e implementar sitios web. No se define como un diseño tradicional, sino como un diseño que tiene dentro de sus conceptos la navegabilidad, usabilidad, arquitectura de la información e interactividad. El diseño web puede ser re-definido como una práctica social especializada que consiste en el procesamiento racional, intuitivo y fáctico de una serie de variables objetivas y subjetivas por medio del cual los hombres intervienen operativamente sobre la realidad material, natural y artificial.

En el diseño web, se aprecia el diseño de formularios y para hacer la tarea más rápido existen sistemas diseñadores de formularios, que según los autores Noel Luis Núñez Camallea y Ronald Coutin Abalo quienes definen diseñador de formulario en su libro Diccionario de Informática como ***"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 del usuario"***, así como los generadores de formularios, que permiten crear los formularios automáticamente sin tener que ser pensado por un usuario, en qué lugar se desean los componentes que integran los mismos.

El modelo relacional permite representar la información del mundo real de una manera intuitiva, introduciendo conceptos cotidianos y fáciles de entender por cualquier inexperto. De esta manera, mantiene información sobre las propias características de la base de datos, que facilitan las modificaciones, disminuyendo los problemas ocasionados en las aplicaciones ya desarrolladas. Se trata de un modelo lógico que establece una estructura sobre los datos, aunque posteriormente éstos puedan ser almacenados de múltiples formas.

Un diccionario de datos es un conjunto de metadatos que contiene las características lógicas de los datos que se van a utilizar en el sistema que se programa, incluyendo nombre, descripción, alias, contenido y organización.

Estos diccionarios se desarrollan durante el análisis de flujo de datos y ayuda a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño del proyecto.

Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que

participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.

En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos.

Contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.

Razones para su utilización:

1- Para manejar los detalles en sistemas muy grandes, ya que tienen enormes cantidades de datos, aun en los sistemas más chicos hay gran cantidad de datos. Los sistemas al sufrir cambios continuos, es muy difícil manejar todos los detalles. Por eso se registra la información, ya sea sobre hoja de papel o usando procesadores de texto. Los analistas más organizados usan el diccionario de datos automatizados diseñados específicamente para el análisis y diseño de software.

2- Para asignarle un solo significado a cada uno de los elementos y actividades del sistema. Los diccionarios de datos proporcionan asistencia para asegurar significados comunes para los elementos y actividades del sistema y registrando detalles adicionales relacionados con el flujo de datos en el sistema, de tal manera que todo pueda localizarse con rapidez.

3- Para documentar las características del sistema, incluyendo partes o componentes así como los aspectos que los distinguen. También es necesario saber bajo qué circunstancias se lleva a cabo cada proceso y con qué frecuencia ocurren. Produciendo una comprensión más completa. Una vez que las características están articuladas y registradas, todos los participantes en el proyecto tendrán una fuente común de información con respecto al sistema.

4- Para facilitar el análisis de los detalles con la finalidad de evaluar las características y determinar donde efectuar cambios en el sistema. Determina si son necesarias nuevas características o si están en orden los cambios de cualquier tipo. Se abordan las características:

- Naturaleza de las transacciones: las actividades de la empresa que se llevan a cabo mientras se emplea el sistema.
- Preguntas: solicitudes para la recuperación o procesamiento de información para generar una respuesta específica.
- Archivos y bases de datos: detalles de las transacciones y registros maestros que son de interés para la organización.
- Capacidad del sistema: habilidad del sistema para aceptar, procesar y almacenar transacciones y datos

5- Localizar errores y omisiones en el sistema, detectan dificultades, y las presentan en un informe. Aun en los manuales, se revelan errores.

1.2 Sistemas diseñadores y generadores de formularios web en el mundo

El análisis de los sistemas diseñadores y generadores de formularios web que existen facilita exponencialmente cuál de estas herramientas a utilizar para la realización de esta investigación o cuáles de los sistemas se adecuan a las necesidades presentadas por los clientes. Algunos de estos sistemas son utilizados por disímiles usuarios, ya sean empresariales, estatales o usuarios corrientes, dado por las particularidades que implementan y los servicios que brindan.

A continuación se enunciarán algunas de las herramientas de diseño de formularios web existentes a nivel mundial:

- ✓ **Form1 Builder:** Puede ser utilizado en cualquier navegador de Internet y permite construir fácilmente formularios web seguros. Entre sus características se incluyen: 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 web de varias páginas. Memoria de formulario para mantener la información y asegurar que el contenido ingresado nunca se perderá. Form1 (el código de formulario generado) corre en su servidor web. Para poder correr Form1 su servidor debe soportar el lenguaje de programación PHP. La mayoría de los servidores web soportan PHP. Plataforma en la que trabaja, Windows.
- ✓ **DesignExpert:** El software de diseño de formularios DesignExpert permite a usuarios de formularios escaneables diseñar formularios para reproducción de lectura por marcas ópticas (OMR) o aplicaciones de proceso de formularios por imágenes. Dicho software logra crear y personalizar formularios que se ajusten a necesidades específicas e imprimir los formularios en

el sitio del usuario o en el de la empresa productora del mismo. DesignExpert presenta diferentes funciones para la creación de formularios.

- ✓ **FormLogix:** Aplicación que 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:** Herramienta 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, gestionar y entender la información recogida.
- ✓ **Formspring:** Herramienta que permite generar de formularios online, que se puede insertar en los sitios web. Cuenta con una serie de planes, siendo algunos de los mismos gratis. Con Formspring se pueden generar formularios con la utilización de herramientas que permiten realizar diversas tareas como insertar campos personalizados, editores y ordenadores.
- ✓ **Flow:** Permite la generación de formularios web online, es una herramienta sencilla de utilizar ya que no requiere del conocimiento de programación del usuario.
- ✓ **QuestionForm:** Crea formularios profesionales de manera rápida. 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 de seguridad.

Estas herramientas son muy eficientes atendiendo a las particularidades que cada una implementa en sus funcionalidades, pero no son utilizadas por DATEC ya que el diseño de los formularios no es integrable a las necesidades del SIGOB, su trabajo se hace a partir de un modelo relacional pero no permite el diseño de los formularios de forma desconectada a la base de datos, lo cual haría más rápido y eficiente el proceso de captura de datos, además que muchos de estos sistemas antes estudiados son privativos. Motivo por el cual no se usarán estas herramientas ya que no dan respuestas al problema planteado.

1.3 Sistemas diseñadores y generadores de formularios web en la UCI

Según el sitio web Kumbia PHP Framework cualquier sistema para agilizar y automatizar el proceso de captura de datos debe tener un diseñador y generador de formularios web que de acuerdo a sus necesidades presente ventajas en cuanto a:

- ✓ Realización de la mayor parte del trabajo
- ✓ Generación de interfaces
- ✓ Validación de datos e integridad
- ✓ Flujo de entrada de datos
- ✓ Presentación de información
- ✓ Adaptación a necesidades específicas
- ✓ Producción de resultados más rápidos.

El proyecto SIGE presenta un diseñador de formularios relacionados con estadísticas, esta herramienta permite al usuario diseñar tablas estadísticas a partir de indicadores y aspectos, ofreciendo una serie de funcionalidades para definir los elementos que conforman los mismos. La aplicación contempla el diseño de formularios web de nomenclatura cerrada y de nomenclatura abierta, así como la edición de formularios previamente creados y guardados y/o exportados desde la misma aplicación.

1.4 Metodología para desarrollo de software

El desarrollo de software puede ser un reto para los desarrolladores, pues crear un software en el menor tiempo posible y con calidad puede ser casi imposible, si no se cuenta con algún proceso que ayude a agilizar el desarrollo de software.

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.

Apoyando lo antes mencionado según Grady (2000), plantea que un proceso de desarrollo de software define quién 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. (2)

Dentro de las metodologías de desarrollo de software se encuentra Open UP seleccionada por el grupo de arquitectura de Soluciones Integrales por ser metodología ágil, con iteraciones cortas y con un enfoque centrado al cliente.

1.4.1 Open up

Dentro de las metodologías existentes se ha decidido utilizar Open Up debido a que fue aprobada por la línea de Soluciones Integrales para el desarrollo de sus productos, esta permite mantener la filosofía de RUP la cual es una de las más empleadas actualmente en la universidad.

Open UP/Basic es un Framework de procesos de desarrollo de software de código abierto. Es un proceso modelo y extensible, dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

Open UP/Basic se centra en articular la arquitectura para facilitar la colaboración técnica, reducir el riesgo y minimizar el sobreesfuerzo de desarrollo. Procura un equilibrio entre las necesidades de los involucrados con los resultados del proyecto y los costos técnicos, con el fin de maximizar el valor delos involucrados y las guías del proceso de desarrollo. (3)Teniendo en cuenta lo antes planteado se realiza un ciclo de vida interactivo que mitiga el riesgo a tiempo y ofrece demostrar resultados en curso al cliente del proyecto.

1.4.2 Beneficios en el uso del Open Up

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

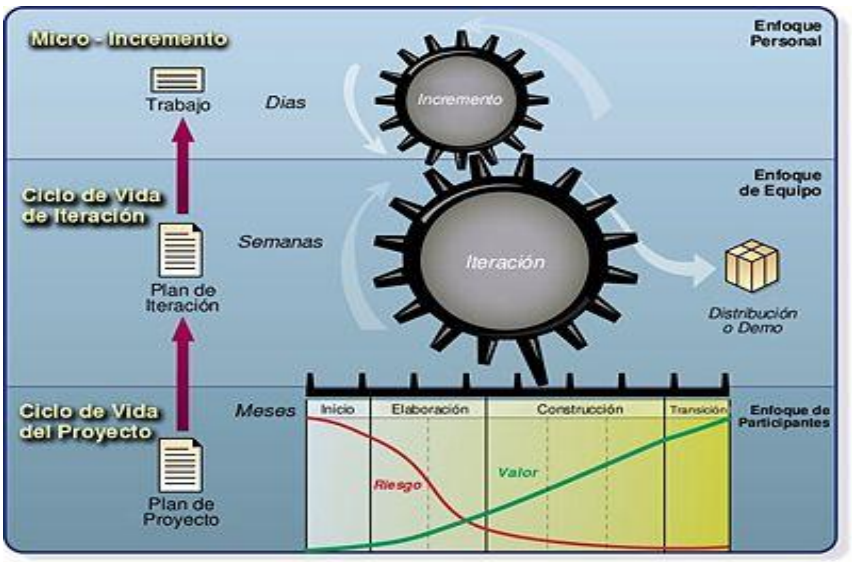


Figura 1: Ciclo de vida Open Up.

Fases de Open UP

1. Concepción

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

2. Elaboración

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

3. Construcción

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

4. Transición

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

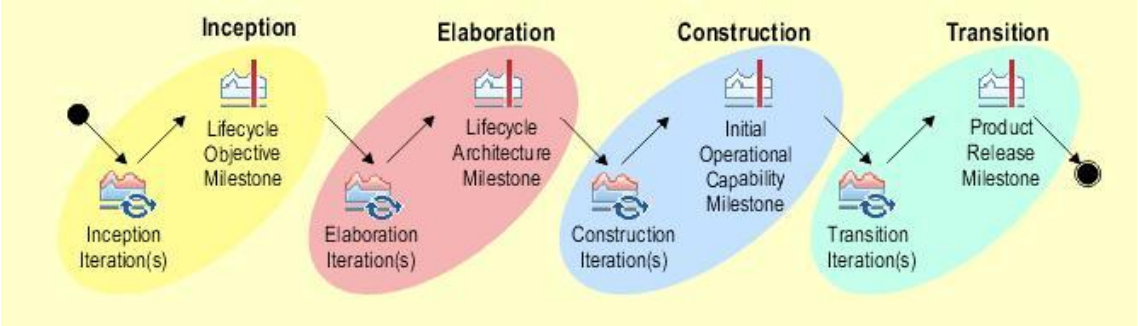


Figura 2: Iteraciones de Open Up.

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.

- **Desarrollador**

El grupo de arquitectura de DATEC define que el desarrollador es el responsable del avance de una parte del sistema, incluyendo el diseño que se ajusta a la arquitectura y la posible creación de un prototipo de la interfaz de usuario.

1.5 Principales patrones de arquitectura, casos de uso y diseño

Concepto de patrón

Según **Christopher Alexander** cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma

forma. Cada uno es una regla de tres partes, que expresa una relación entre un contexto, un problema y una solución. (4)

1.5.1 Patrones de arquitectura

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. Algunos patrones de arquitectura son:

1. Capas

2. Tubería-filtros

3. Pizarra

4. Bróker

5. Modelo-Vista-Controlador

6. Presentación-Abstracción-Control

7. Reflexión

8. Microkernel.

De acuerdo con la definición y la función que realizan estos patrones se ha decidido utilizar en el diseño del módulo el siguiente: Modelo-Vista-Controlador (MVC): en ocasiones se le define más bien como un patrón de diseño o como práctica recurrente, y en estos términos es referido en el marco de la estrategia arquitectónica de Microsoft. (5) El patrón MVC separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes: - Modelo: el modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). - Vista: Maneja la visualización de la información. - Controlador: Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

1.5.2 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 requerimientos del sistema, de una forma más eficiente. Los siguientes patrones son empleados en la solución:

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

Concordancia Reuso: Modela la reutilización del modelo que consiste en tres casos de uso. La primera sub-secuencia, llamada Común, modela la secuencia de las acciones que debe aparecer en múltiples casos de empleo en el modelo. Otros casos de empleo modelan los usos del sistema que comparten la sub-secuencia común de acciones. Obviamente, habrá al menos dos de ellos.

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

1.5.3 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 empleados se encuentra **GRASP** (las siglas pertenecen a las palabras del inglés, General Responsibility y 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.

- ✓ **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.
- ✓ **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.
- ✓ **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.

Estos patrones facilitan la localización de los diferentes objetos del sistema, así como la comunicación y el entendimiento entre desarrolladores y diseñadores. En este caso están los 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.
  - **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.
  - **Prototipo**: especifica los tipos de objetos a crear por medio de una instancia de un prototipo, y crear nuevos objetos copiando este mismo prototipo.

- **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
- **Puente**: desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.
- **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.
- **Decorador:** añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender funcionalidades.
- **Fachada:** proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que sea más fácil de utilizar el subsistema.

➢ **Patrones de comportamiento**

- **Cadena de Responsabilidad:** evita acoplar el emisor de una petición a su receptor, al dar más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada para un objeto.
- **Iterador:** proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
- **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.
- **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.
- **Estado:** permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Tal parece que cambia la clase del objeto.

1.6 Ambiente de Desarrollo

1.6.1 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 del Grupo de Gestión de Objetos, en

inglés Object Management Group (OMG), y no solo es la forma mundial de representar la estructura de las aplicaciones, comportamiento y arquitectura sino que también que representa procesos de negocio y estructuras de datos. (6) 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

Según Larman en su libro plantea que UML presenta ventajas en cuanto a:

- ✓ 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. (7)

1.6.2 Herramienta CASE

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, de las cuales hablaremos de Visual Paradigm por ser la herramienta a utilizar en la solución de la investigación, ya que ha sido definida así por la línea Soluciones Integrales.

- ✓ **Visual Paradigm for UML 6.1**

Es una herramienta CASE, desarrollada por la compañía Visual Paradigm International, que utiliza UML como lenguaje de modelado. Presenta las siguientes características:

- Soporta el ciclo de vida completo del software: análisis, diseño, implementación y despliegue.
- Permite la captura de requisitos, el dibujo de diagramas UML, la realización de ingeniería inversa y generación de código PHP.
- Se integra con las siguientes herramientas:

  - Eclipse/IBM Web Sphere
  - Builder
  - Net Beans IDE
  - Oracle JDeveloper
  - BEA Weblogic

- Está disponible en varias ediciones, cada una destinada a necesidades específicas: Enterprise, Professional, Community, Standard, Modeler y Personal. (8)

1.7 Tecnologías

JavaScript

Este lenguaje 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 utilizado en la realización de páginas Web. La verdadera fuerza de JavaScript es la compatibilidad con los distintos navegadores. Además de que es interpretado por todos los navegadores modernos.

Las principales características de JavaScript como lenguaje son:

- ✓ Interpretado
- ✓ Estructurado: provee todos los recursos de un lenguaje estructurado como C (funciones, ciclo por conteo y por condición, condicionales, condicionales múltiples, y demás estructuras).
- ✓ Tipado Dinámico: como la mayoría de los lenguajes scripts los tipos de las variables están asociados al valor que contiene en un momento dado.

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

Como plantea Armando Robert Lobo en su tesis (2009), en principio JavaScript se tomó como un lenguaje torpe para desarrollos pesados, fundamentalmente porque cada navegador implementa diferentes características del estándar, por todo esto se relegó su utilidad a simples efectos visuales y a la validación de formularios. Tal realidad parece que está cambiando rápidamente a tal punto que ya se habla de un paradigma en el uso de JavaScript conocido como JavaScript No Obstructivo y aunque no se dispone de una definición concreta se plantean principios a seguir a la hora de desarrollar aplicaciones serias en lo que a portabilidad refiere. La aparición de tecnologías como AJAX vienen a complementar junto al acceso al DOM que desde hace tiempo provee el lenguaje, las posibilidades de desarrollo de aplicaciones enriquecidas para la web, de esta forma se da el salto de un modelo sincrónico a disponer ahora, con la utilización de AJAX, de aplicaciones asíncronas, que por sus características llevan la interacción con la web a otra dimensión.

Para la implementación del diseñador y generador de formularios web se empleó este lenguaje, ya que es propuesto por la línea de Soluciones Integrales, la línea para la capa de presentación de sus productos define utilizar ExtJS como framework de desarrollo y este el lenguaje que soporta es JavaScript.

XML

Son las siglas de Extensible Markup Language o Lenguaje de Marcación Extensible desarrollado actualmente por el W3C. Es utilizado para el intercambio de datos entre diferentes plataformas, pero de forma general engloba un conjunto de tecnologías desarrolladas sobre este para el tratamiento de datos, siendo su principal característica la de describir la información que contiene. El desarrollo de XML comenzó en 1996 en un grupo de trabajo creado por el W3C estando la primera versión lista en 1998, desciende de SGML Standard Generalized Markup Language o Lenguaje General de Marcas (ISO 8879:1986 SGML) quien a su vez fue desarrollado en la década del 60 por Charles Goldfarb, Edward Mosher and Raymond Lorie de IBM. Tiene innumerables usos desde la transmisión de noticias vía RSS hasta la transmisión de información científica como es el caso de CML Chemical Markup Language. En el marco de la descripción de la arquitectura adquiere importancia como formato para la exportación de información a través de mecanismos de serialización.

JavaScript Object Notation (JSON)

JSON es una notación muy simple para definir objetos en JavaScript, representa una alternativa al intercambio de información respecto a XML al resultar más ligero, a ello se suma que es más fácil su interpretación al poder ser decodificado directamente en un objeto.

Por todo lo antes planteados es que se decide utilizar esta tecnología en el desarrollo de la aplicación ya que facilita el trabajo con objetos en JavaScript.

1.8 Marco de desarrollo para la aplicación web

Los marcos o framework de desarrollo web son muy utilizados porque representan una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación, es decir representa una aplicación incompleta que es configurable a la que se le puede añadir piezas concretas para la construcción de una aplicación con nuevas características. Un framework puede incluir programas de soporte, librerías de código, códigos script, u otras aplicaciones que faciliten el desarrollo. Para las aplicaciones desarrolladas en PHP existe una cantidad considerable de frameworks libres de mucha calidad y renombre que pueden ser utilizados entre los que se pueden encontrar a Zend Framework, y Symfony entre muchos otros. En esta investigación se hablará del marco de desarrollo ExtJS, debido a su uso en la implementación de los componentes referidos en el documento.

ExtJS 3.2

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

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

La liberación final de la versión 3.3 de ExtJS se realizó el 11 de octubre de 2010. Agregando dentro de sus componentes elementos como Pivot Grid, Action Column y nuevos componentes para el manejo de Calendarios.

1.9 Entorno de Desarrollo

NetBeans IDE 6.8.

El NetBeans es un entorno de desarrollo integrado (IDE), por sus siglas en inglés (Itegrated Development Environment) es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso (9). Tiene buen soporte de Portal en Perl para Automatización de Web (Webapps) (war, jsp y Servlets) y además en la actualidad cuenta con el plugin *Portal Pack* para la realización de Portlets.

Características:

- ✓ Generación automática de código.
- ✓ Utilidades de búsqueda y reemplazo muy intuitivos, amigables y sobre todo realmente útiles para el programador.
- ✓ Coloración de código.
- ✓ Refactorización inteligente.
- ✓ Diseñadores de interfaces gráficas para Swing y JSF.
- ✓ Asistentes para escritura de código rápido de acceso a datos basados en JPA (Java Persistence API).
- ✓ Asistentes para escritura de servicios Web.
- ✓ Soporte para múltiples máquinas virtuales.
- ✓ Soporte para múltiples servidores contenedores J2EE.
- ✓ Soporte para librerías de terceros.
- ✓ También es importante recordar la variedad de plugins (componentes de Software conectables a NetBeans) disponibles para incrementar las funcionalidades del IDE.

¿Por qué NetBeans IDE?

Las características que presenta el NetBeans como IDE open source y multiplataforma lo convierte en uno de los más potentes en la actualidad. En los últimos tiempos el IDE cuenta con mejoras que permite poder desarrollar en PHP, JavaScript, CSS, HTML, XHTML, entre otras tecnologías. Permitiendo además 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 componente diseñador y generador de formularios.

1.10 Conclusiones

En el presente capítulo se explicaron los fundamentos teóricos de los diseñadores y generadores de formularios, los cuales forman parte de disimiles herramientas como funcionalidades de las mismas. Además se puede llegar a la conclusión de que los componentes a desarrollar tendrán una ventaja sobre los sistemas vistos en la investigación puesto que representará algunos conceptos del modelo relacional, ayudando a que el sistema de información del gobierno cuente con componentes ágiles de utilizar y con características. Se seleccionó para dar solución a la problemática planteada, la metodología de desarrollo Open UP, el lenguaje de modelado visual UML, la herramienta Case Visual Paradigm, el lenguaje de programación JavaScript y Ajax. Como principal framework de desarrollo ExtJS, como IDE de desarrollo NetBeans, y finalmente como gestor de base de datos PostgreSQL.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

Introducción

En este capítulo se realizará una descripción del sistema a automatizar, mostrándose los actores que interactúan con el sistema, los requisitos mínimos que debe tener el sistema de cómputo donde se vaya a instalar la aplicación, así como los requisitos funcionales necesarios que el componente debe poseer. Se mostrará el modelo de dominio del sistema, el diagrama de casos de uso del sistema para la mejor comprensión de las funcionalidades del componente. Además de los principales elementos del diseño del sistema.**Modelo de dominio del sistema**

El modelo conceptual permite representar y explicar los conceptos significativos en un dominio del problema. Modelo que comunica a los interesados cuales son los términos importantes y como se relacionan entre sí. El modelo de dominio de la presente investigación se muestra en la figura 3.Los pasos principales para su realización son.

- ✓ Identificar las clases conceptuales.
- ✓ Representarlas en diagrama de clases.
- ✓ Identificar sus relaciones.

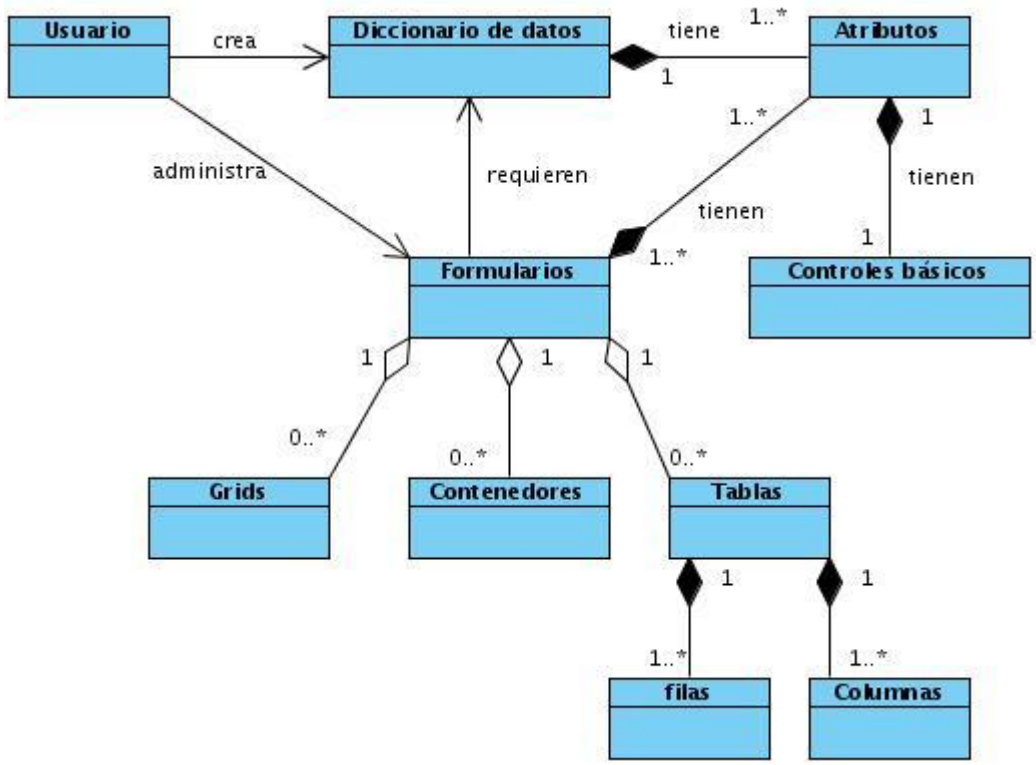


Figura 3: Modelo de Dominio.

Para una mejor compresión de este diagrama a continuación se describen las clases conceptuales representadas en el mismo:

Usuario

Entidad que administra los formularios.

Diccionario de datos

Esta entidad es creada y administrada por el Usuario, es la que encargada de facilitar y manejar el modelo relacional de la aplicación lo que permite el diseño de los formularios web de forma desconectada a la base de datos.

Formularios

Esta entidad se encarga de trabajar con los formularios que son administrados por el usuario.

Clase Atributos

Representa el contenido de una base de datos para el trabajo desconectado de la misma.

Controles básicos

En esta entidad se selecciona los controles que el usuario desea para el diseño y generación de formularios web, dándole doble click al control básico.

2.3 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 requerimientos no funcionales de este sistema a automatizar son:

Usabilidad

RNF 1 Facilidad de uso

- ✓ El sistema será fácil de manejar por cualquier persona, incluso aquellas con pocos conocimientos informáticos.
- ✓ El sistema debe ser intuitivo y tener un alto nivel de usabilidad permitiendo que usuarios sin mucho conocimiento informático, en aproximadamente 15 días puedan explotarlo al 100%.

Fiabilidad

RNF 2 Disponibilidad

El sistema debe poder permanecer constantemente funcionando excepto cuando sea necesario reiniciarlo o detenerlo por tareas de mantenimiento o cambio de configuración.

RNF 2.1 Disponibilidad de Servicios

- ✓ Debido a que el sistema mantendrá en todo momento una cola de eventos, es necesario que este permanezca en línea constantemente para atender las solicitudes desde un usuario directamente.
- ✓ El sistema permanecerá fuera de servicio sólo en caso que se estén realizando tareas de

mantenimiento o de configuración.

RNF 3 No deben ocurrir errores críticos

Entre los errores críticos están: pérdida total de la información, o inhabilitación para el uso de ciertas partes del funcionamiento del sistema.

Interfaces

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

Hardware

- ✓ Se debe contar con 256 MB de memoria RAM como mínimo, aunque lo ideal sería 512 MB.
- ✓ Procesadores Pentium IV.

2.4 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 requerimientos funcionales del sistema a automatizar son:

RF 1 Adicionar entidad

RF 2 Eliminar entidad

RF 3 Adicionar atributo

RF 4 Eliminar atributo

RF 5 Adicionar contenedor

RF 6 Adicionar control básico

RF 7 Editar propiedades del contenedor

RF 8 Editar propiedades del control básico

RF 9 Asociar atributo al control básico

RF 10 Asociar entidad al contenedor

RF 11 Adicionar área de trabajo

RF 12 Generar formulario.

Actores del sistema

Los actores del sistema son aquellos individuos u otros sistemas externos que interactúan con este. El actor establece un rol en el cual varios individuos pueden jugar ese papel.

Tabla 1: Descripción del actor del sistema.

Actor	Descripción
Diseñador	Persona encargada de realizar los formularios

2.5 Diagrama de casos de usos del Sistema

El artefacto fundamental en este flujo de trabajo es el diagrama de caso de uso del sistema que contiene actores y casos de uso. Este artefacto constituye un modelo de las funciones deseadas para el sistema y su entorno, además sirve como contrato entre el cliente y los desarrolladores. Se utiliza como entrada esencial para las actividades de análisis, diseño, implementación y prueba.

2.6 Aplicación de patrones de casos de uso

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Para la obtención de los casos de uso se emplearon los patrones de casos de uso mencionados en el capítulo anterior y que se explica cómo se evidencian los mismos en el diagrama de casos de uso del sistema.

El patrón CRUD propone formar un caso de uso a partir de los requisitos funcionales relacionados con las acciones de adicionar y eliminar una determinada información, como se evidencia en los casos de uso Administrar diccionario de datos, el cual comprende las funcionalidades adicionar y eliminar entidades. En el caso de uso Administrar área de trabajo que contiene los requisitos adicionar área de

trabajo, salvar y generar formulario. Además en el caso de uso administrar propiedades se recogen las funcionalidades de editar las propiedades del contenedor y el control básico.

La posibilidad de administrar propiedades durante el proceso de diseño del formulario evidencia el uso del patrón extensión concreta. Mientras que el patrón concordancia reuso se refleja a la hora de diseñar un formulario y administrar el diccionario de datos, donde siempre se deben ejecutar la funcionalidad comprendida en el caso de uso asociar controles básicos.

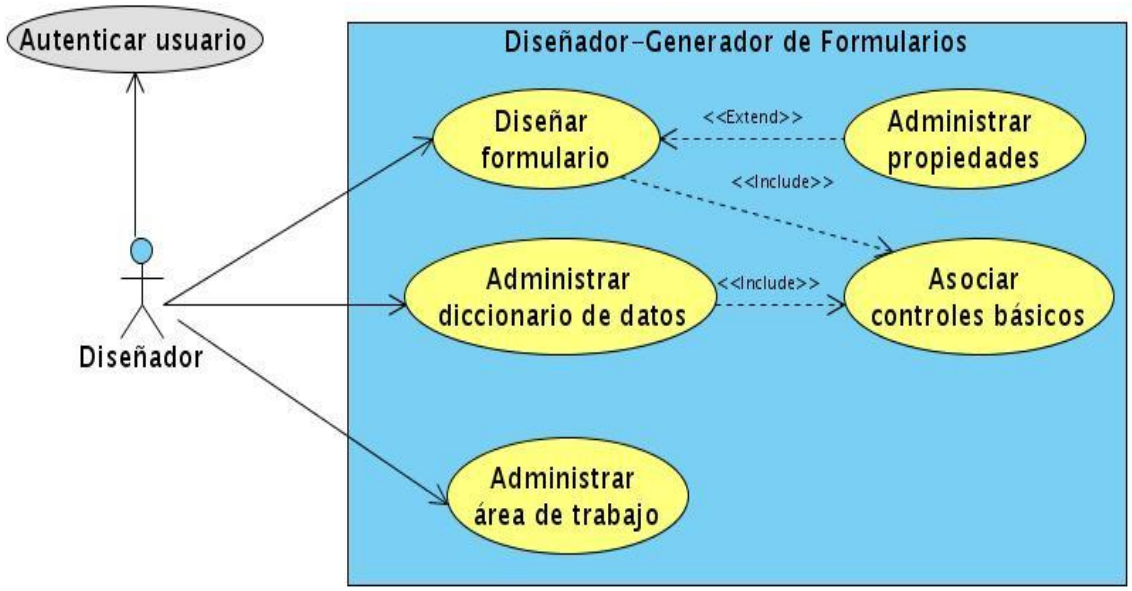


Figura 4: Diagrama de Caso de Uso del Sistema.

2.7 Descripción de los casos de usos del sistema

Descripción del caso de uso Administrar diccionario de datos

En este caso de uso el diseñador puede realizar operaciones sobre los diccionarios de datos, como adicionar y eliminar entidades y atributos, importarlos y exportarlos.

Tabla 2: Descripción del CUS Administrar diccionario de datos.

Caso de Uso:	Administrar diccionario de datos.
Actores:	Diseñador

Resumen:	El caso de uso se inicia cuando el diseñador decide realizar alguna de estas funcionalidades. **Adicionar entidad al diccionario de datos**, el actor decide comenzar a diseñar los formularios, y crea el nuevo diccionario agregándole una nueva entidad, terminando así el CU. **Eliminar entidad del diccionario de datos**, el actor decide borrar una entidad en diccionario de datos, terminando así el CU. **Adicionar atributos al diccionario de datos,** el actor decide comenzar a diseñar los formularios, y crea el nuevo diccionario agregándole un nuevo atributo a la entidad seleccionada en el diccionario de datos, terminando así el CU. **Eliminar atributo del diccionario de datos**, el actor decide borrar un atributo de la entidad seleccionada en el diccionario de datos, terminando así el CU.
Precondiciones:	El diseñador debe estar autenticado en el sistema.
Referencias	RF1, RF2, RF3, RF4
Prioridad	Crítico

Descripción del caso de uso Diseñar formularios

En este caso de uso el diseñador puede realizar operaciones sobre como adicionar contenedor y adicionar controles básicos.

Tabla 3: Descripción del CUS Diseñar formularios.

Caso de Uso:	Diseñar formulario
Actores:	Diseñador

Resumen:	El caso de uso se inicia cuando el diseñador decide realizar alguna de estas funcionalidades. **Adicionar contenedor**, el actor decide comenzar a diseñar los formularios, y selecciona un nuevo contenedor y lo adiciona en el área de trabajo, terminando así el CU. **Adicionar control básico**, el actor decide comenzar a diseñar los formularios, selecciona un nuevo control básico y se lo adiciona al contenedor seleccionado que se encuentra en el área de trabajo, terminando así el CU.
Precondiciones:	El diseñador debe estar autenticado en el sistema.
Referencias	RF 5, RF 6
Prioridad	Crítico

Descripción del caso de uso Administrar propiedades

Este caso de uso le permite al diseñador realizar operaciones sobre los formularios como editar propiedades del contenedor y de los controles básicos.

Tabla 4: Descripción del CUS Administrar propiedades.

Caso de Uso:	Administrar propiedades
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el diseñador decide realizar alguna de estas funcionalidades. **Editar propiedades del contenedor**, el diseñador decide tener el control de las propiedades del contenedor seleccionado y de esta forma poder modificar sus parámetros, finalizando así el CU. **Editar propiedades de los controles básicos**, el diseñador decide tener el control de las propiedades del control básico seleccionado dentro del contenedor y de esta forma poder modificar sus parámetros, finalizando así el CU.
Precondiciones:	El diseñador debe estar autenticado en el sistema.

Referencias	RF 7, RF 8.
Prioridad	Secundario

Descripción del caso de uso Asociar controles básicos

En este caso de uso el diseñador puede realizar operaciones como asociar atributo al control básico, y asociar entidad al contenedor.

Tabla 5: Descripción del CUS Asociar controles básicos.

Caso de Uso:	Asociar controles básicos
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el diseñador decide realizar alguna de estas funcionalidades. **Asociar entidad al contenedor**, el diseñador decide relacionarle al contenedor los datos que posee una entidad en el diccionario de datos, finalizando así el CU. **Asociar atributo al control básico**, el diseñador decide relacionarle al control básico seleccionado en el contenedor, los datos que posee un atributo de la entidad que fue relacionada con ese mismo contenedor al que pertenece el control básico, finalizando así el CU.
Precondiciones:	El diseñador debe estar autenticado en el sistema, debe haber creado un diccionario de datos y al menos diseñado algún formularios
Referencias	RF 9, RF 10
Prioridad	Crítico

Descripción del caso de uso Administrar área de trabajo.

En este caso de uso el diseñador puede realizar operaciones sobre el área de trabajo, como adicionar área de trabajo, salvar formulario y generar formulario.

Tabla 6: Descripción del CUS Administrar área de trabajo.

Caso de Uso:	Administrar área de trabajo
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el diseñador decide realizar alguna de estas funcionalidades. **Adicionar área de trabajo**, el diseñador decide relacionarle al contenedor los datos que posee una entidad en el diccionario de datos, finalizando así el CU. **Generar formulario**, el diseñador decide generar el formulario antes diseñado, terminando así el CU.
Precondiciones:	El diseñador debe estar autenticado en el sistema.
Referencias	RF 11, RF 12.
Prioridad	Crítico

2.8 Descripción de los patrones de diseño aplicados en el sistema.

2.8.1 Patrones GRASP

Alta Cohesión: cada una de las clases realiza una labor única.

Bajo Acoplamiento: las clases tienen pocas dependencias entre ellas. En este caso la clase General Panel es la que se comunica con cada una de las clases existentes.

Creador: para acceder a los distintos componentes y trabajar sobre ellos, se crean objetos de ellas, siendo las mismas clases quienes lo crean.

2.8.2 Patrones GoF

Observador: el sistema será construido basado en una arquitectura orientada a componentes. Entre dichos componentes existirá una estrecha relación de colaboración por lo que debe garantizarse su comunicación, manteniendo a la vez un bajo acoplamiento. Buscando dicho objetivo se ha propuesto el uso de este patrón, el cual propicia la interacción de los elementos a través del lanzamiento y captura de eventos.

A continuación se muestra la aplicación de este patrón en la realización del caso de uso Diseñar formularios:

El sistema está compuesto por 5 componentes: GeneralPanel, NorthPanel, TreeArea, TabWorkArea y GridPanel. GeneralPanel es el responsable de crear las instancias del resto de los elementos, por lo que además tiene la responsabilidad de gestionar la comunicación entre ellos (Fig. 5).

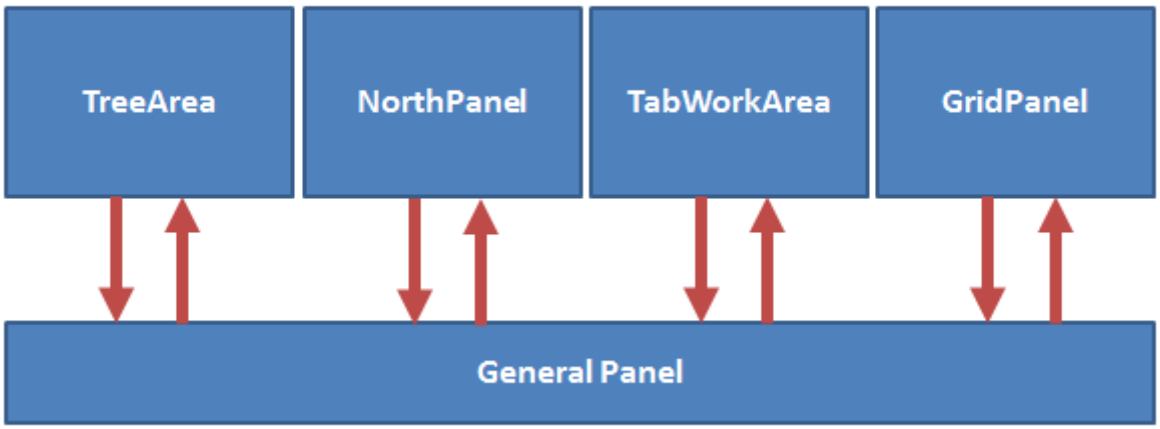


Figura 5: Relación entre componentes.

Al seleccionarse una entidad en el Diccionario de Datos, el componente TreeArea lanza el evento 'nodeclick' que es capturado en GeneralPanel, para ser almacenada en una variable. Seguidamente debe ser presionado el botón 'Diseñar', ubicado en NorthPanel, que propicia el lanzamiento del evento 'asociar' el cual es capturado igualmente en GeneralPanel. En ese momento se muestra una ventana para obtener datos relacionados con el contenedor que se va a diseñar. Una vez obtenida la información es lanzado el evento 'newcontenedor' y este capturado por GeneralPanel, el cual ejecuta la función 'addContenedor' del componente TabWorkArea, encargada de dibujar el contenedor asociado a la entidad seleccionada. Además se recargan los elementos de GridPanel mediante las funciones correspondientes.

Con la implementación de este mecanismo se garantiza la integración y funcionamiento como un todo de los componentes manejados por GeneralPanel, sin que exista una relación directa entre ellos. Es decir, un elemento lanza un evento y un observador lo captura para ejecutar una función determinada, que puede ser, como en este ejemplo, ordenar a otros elementos a realizar una acción.

Prototipo: JavaScript es un lenguaje prototipado, con esta característica se logra simular la orientación a objetos. Con el mismo no se pueden crear clases ni instanciar objetos, pero si se puede lograr un comportamiento similar aplicando el patrón Prototipo. En este caso se declaran funciones (clases) y mediante el operador 'new' se crean los prototipos o copias (instanciación) de estas funciones, obteniéndose así nuevas funciones (objetos) con el mismo comportamiento pero con estados diferentes.

En el desarrollo de la aplicación este patrón estará presente en el momento en que GeneralPanel crea las diferentes instancias de los componentes que el deberá gestionar.

2.9 Diagrama de clases del diseño

Este tipo de diagramas describe gráficamente las clases de software en una aplicación. Estos presentan clases con sus atributos, métodos y sus dependencias. Durante el diseño se tuvo en cuenta los elementos propuestos en la arquitectura definida para los productos de la línea Soluciones Integrales, con el objetivo de mantener la estandarización en el equipo de desarrollo.

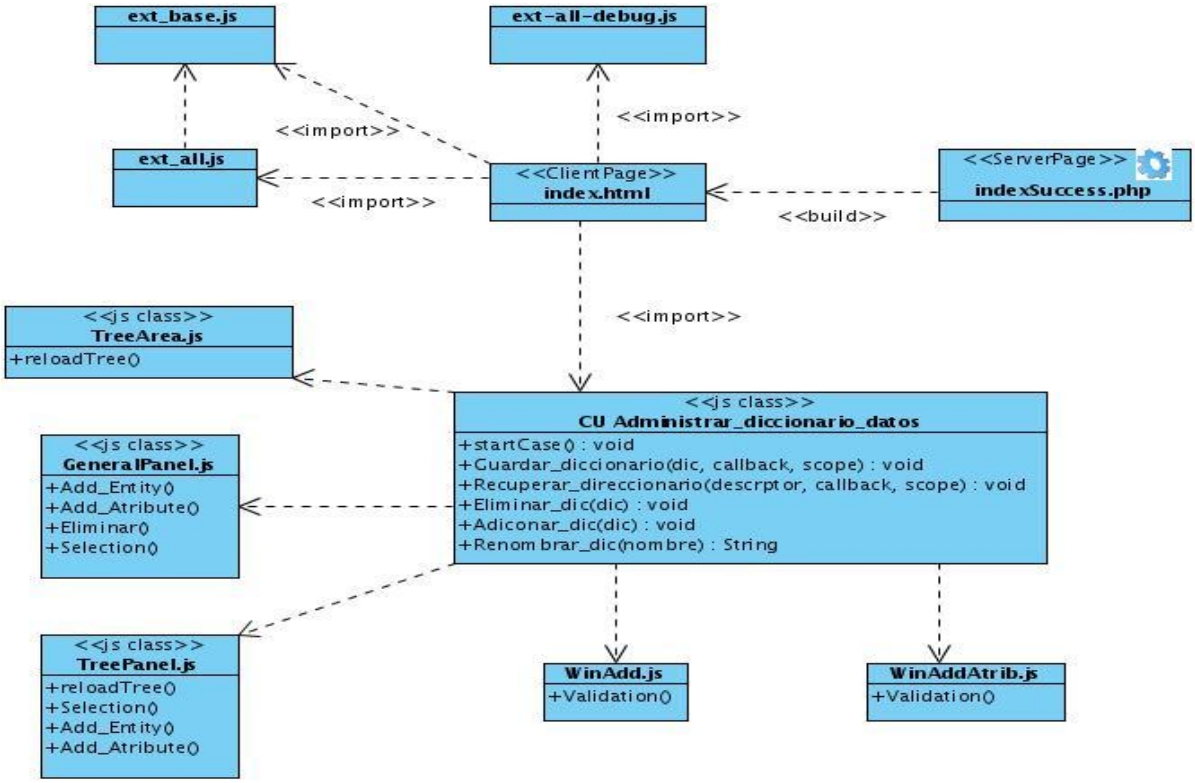


Figura 6: Diagrama de Clases del Diseño.

Principales clases:

GeneralPanel: es el componente que representa toda la interfaz principal, donde se distinguen cada uno de los componentes que lo conforman.

TreePanel: es el componente que se mostrará en parte izquierda de la aplicación y se encargará de mostrar todo el modelo relacional conformado por entidades y atributos.

TreeArea: es el componente que contiene a TreePanel en el lado izquierdo de la aplicación, su principal función es recargar el árbol y todas las funciones que se realizan en TreePanel.

WinAdd: es el componente que se muestra cuando el usuario desea adicionar una nueva entidad.

WinAddAtrib: es el componente que se muestra cuando el usuario desea adicionar una nueva entidad

2.10 Diagramas de interacción del diseño

Secuencia

Los diagramas de interacción muestran las relaciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas, por lo que son empleados para modelar aspectos dinámicos del sistema. Los diagramas de secuencia y de colaboración son dos tipos de diagramas de interacción que son semánticamente equivalentes pero sin embargo los diagramas de colaboración destacan el orden estructural de los objetos que interactúan y los de secuencia le incorporan además el orden temporal de los mensajes. Para la realización de los casos de uso del diseño es más factible el empleo de los diagramas de secuencia ya que representan con más claridad el flujo de las acciones que debe realizar el sistema.

Adicionar área de trabajo

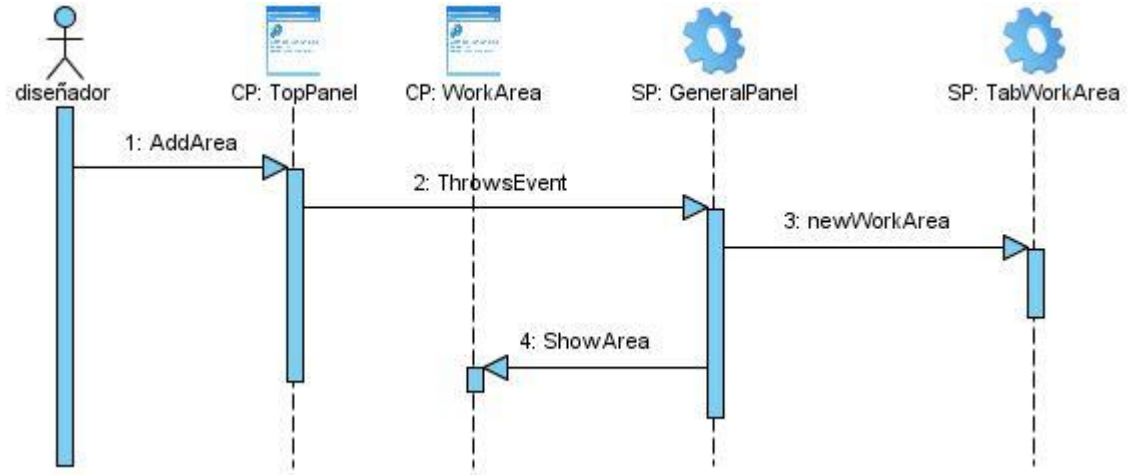


Figura 7: Diagrama de Secuencia Adicionar área de trabajo.

El diseñador adiciona un área de trabajo para diseñar, la client page TopPanel es la encargada de lanzar el evento hacia la Server Page (SP) GeneralPanel gestiona y ordena a la SP TabWorkArea que cree una nueva área de trabajo, la cual es mostrada en la client page TabWorkArea al usuario.

Adicionar entidad al diccionario de datos

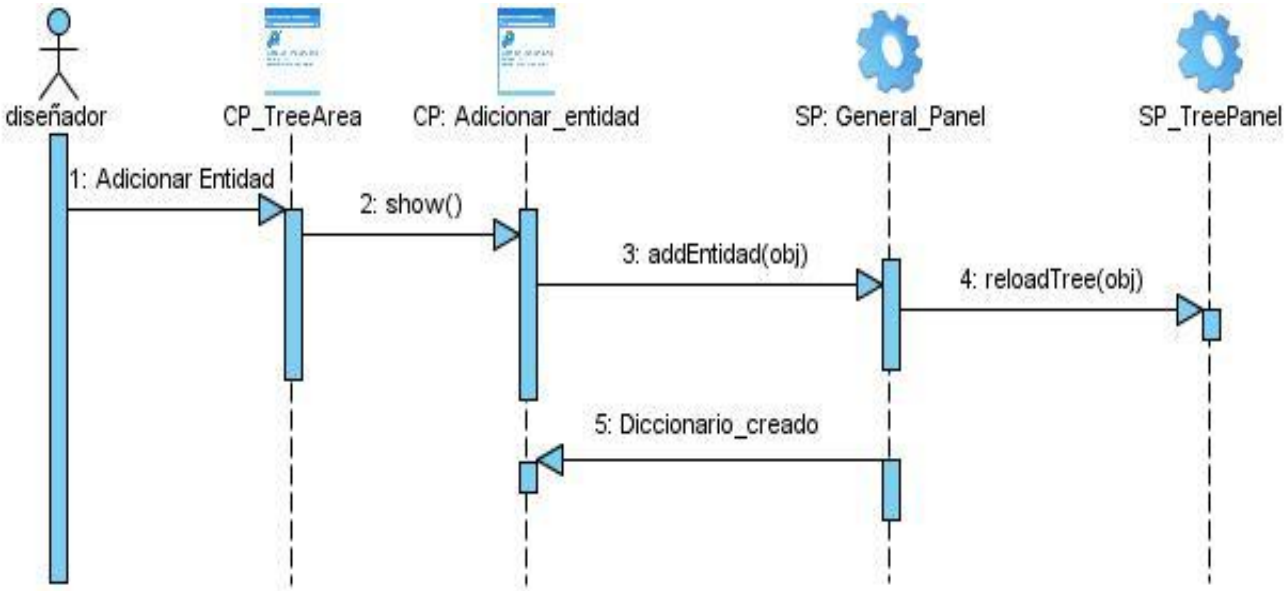


Figura 8: Diagrama de Secuencia Adicionar entidad al diccionario de datos.

En el modelado representado se describe como el diseñador adiciona una entidad al diccionario de datos, este selecciona la opción de adicionar entidad en la CP_TreeArea, después de realizada esta acción se muestra la CP_Adicionar_entidad pidiéndole la información al usuario de esa entidad, SP_Genera_Panel escucha el evento lanzado y manda a ejecutar la opción de reloadTree() de la SP_TreePanel creando así la entidad en el diccionario de datos.

2.11 Diagrama de Despliegue

El Modelo de Despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de computo. Se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. Consiste o está compuesto por los siguientes elementos:

- Nodos: elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.
- Dispositivos: nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.
- Conectores: expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.
- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar. Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como HTTP, USB, TCP/IP, etc.
- La funcionalidad (los procesos) de un nodo se define por los componentes que se distribuyen sobre ese nodo.

Figura 9: Modelado de despliegue del sistema.

2.12 Conclusiones

- ✓ La construcción del modelo de dominio permitió representar y explicar los conceptos o términos significativos del dominio del problema en cuestión, así como la relación entre ellos.
- ✓ Se identificaron 12 requisitos funcionales con los cuales se llegó a una definición formal de lo que el sistema debe cumplir.
- ✓ Se especificaron las propiedades que debe tener el producto final a partir de los requisitos no funcionales.
- ✓ La aplicación de los patrones de casos de uso permitió agrupar las funcionalidades identificadas en 5 casos de uso, las relaciones entre los casos de uso y estructurar el diagrama de casos de uso del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En este capítulo se hace un estudio del diseño del sistema, se representa la realización de casos de uso del diseño a través de los diagramas de clases del diseño, de interacción y diagrama de despliegue. Se realiza una breve descripción de la implementación del sistema, así como fracciones de códigos de las principales clases y métodos que se implementaron, además del modelo de implementación del sistema.

3.1 Diagrama de Componentes

Modelan la vista estática de un sistema. Estos representan las organizaciones y relaciones de dependencia entre componentes software, pudiendo mostrarse las interfaces que estos soporten.

En este tipo de diagramas se tienen en consideración los requisitos relacionados con la habilidad de desarrollo, la gestión del software, la reutilización y las limitaciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

Para la realización de estos diagramas se cuenta con diferentes tipos de componentes, como son:

- ✓ **Document:** Componente que representa un documento.
- ✓ **File:** Componente que representa un documento que contiene código fuente o datos.
- ✓ **Executable:** Componente que se puede ejecutar en un nodo.
- ✓ **Table:** Componente que representa una tabla de una base de datos.
- ✓ **Library:** Especifica una biblioteca de objetos estática o dinámica.

A continuación en la **Figura 8** se muestra el diagrama de componente de la aplicación desarrollada:

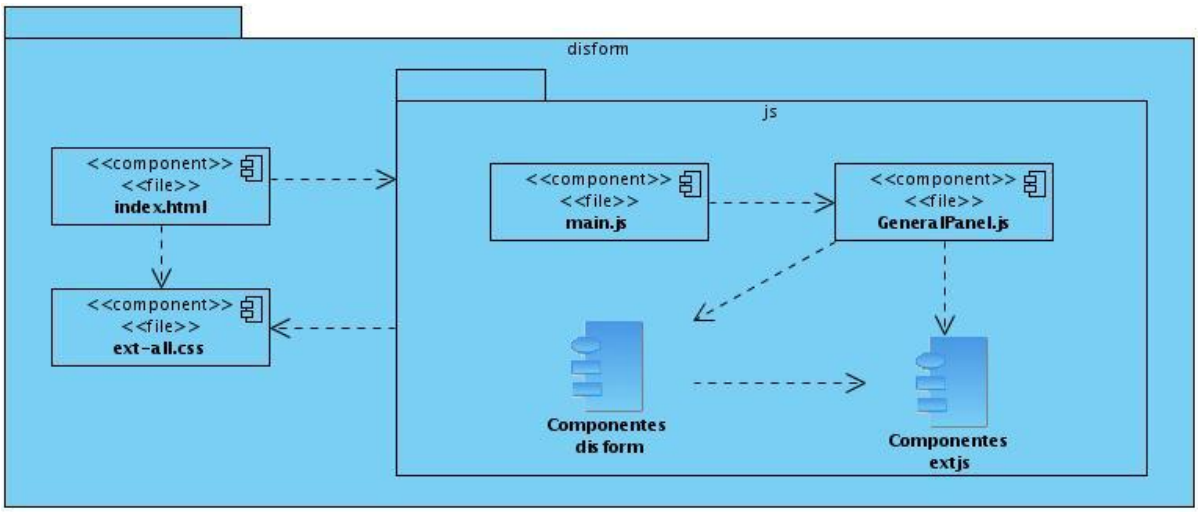


Figura 10: Modelo de implementación.

Descripción de los componentes del sistema

Como se puede observar la aplicación cuenta con cuatro componentes principales, Main, Index, GeneralPanel yext-all.css los cuales estarán desplegados en un servidor web.

Paquete disform

Agrupa las clases asociadas en la implementación del Diseñador y generador de formularios web. Index, ext-all.css y el paquete Js que contiene a GeneralPanel, Main, además de los paquetes de componentes disform y ExtJS.

El GeneralPanel es el componente controlador de cualquier función que se desarrolle en la aplicación, este posee las instancias por regiones de los elementos que se renderizan en el navegador.

El componente Index es el que contiene todas las direcciones de componentes que se muestran en la página web.

El ext-all.css es uno de los archivos que conforman la librería de ExtJS y es utilizado por todos los componentes de la aplicación.

3.2 Fragmento de código de la clase principal

```
this.ta.on('addatribute', function(){
    this.wat = new WinAddAtrib();
    this.wat.show();
    this.wat.on('accept', function(obj){
        for( i= 0; i< this.treedata.length; i++){
            if(this.treedata[i].text == this.selection.text){
                for( j=0; j< this.treedata[i].children.length; j++) {
                    if(this.treedata[i].children[j].text == obj.text) {
                        Ext.MessageBox.show({
                            title: 'Error',
                            msg: 'Existe un atributo con el mismo nombre.',
                            icon: Ext.MessageBox.ERROR,
                            buttons: Ext.MessageBox.OK
                        });
                        return;
                    }
                }
                this.wat.close();
                this.treedata[i].children.push(obj);
                this.datos.entidades[i].attributes.push({
                    attname: obj.text,
                    atttype: obj.datatype,
                    attlength: obj.datalength
                });
            }
        }
        this.ta.reloadTree(this.treedata)
    },this);

},this);
```

Figura 11: Fragmento de código de GeneralPanel.

La figura anterior muestra un segmento del código de la clase que controla cualquier tipo de función y evento lanzado en la aplicación. Este fragmento representa como fue implementada la funcionalidad adicionar un nuevo atributo al diccionario de datos. Primeramente se muestra en pantalla una ventana exigiéndole al usuario la entrada de datos como el nombre del atributo y tipo de dato, después se verifica si ya existe ese elemento y la entidad a la que pertenece. Después de haber hecho estos pasos se agrega el atributo a la entidad correspondiente finalizando de esta manera la operación.

3.3 Pruebas de Software

Debido a la imposibilidad de trabajar de forma perfecta, el avance del software ha de ir acompañado de una acción que garantice la calidad. Las pruebas de software consisten en la ejecución de un programa bajo condiciones y requisitos especificados con la intención de descubrir errores y garantizar

la calidad del sistema. Los objetivos en esta etapa difieren de una iteración a otra. Por lo que se define una estructura de la prueba de acuerdo a los objetivos de la iteración.

Nivel de prueba

La Prueba es empleada para distintos objetivos, en diferentes escenarios o niveles de trabajo. Para garantizar que esta aplicación funcione de forma satisfactoria se aplican las Pruebas de Desarrollador, ya que son diseñadas por el equipo de desarrollo.

Técnicas de Pruebas

Para este nivel de prueba en la aplicación Diseñador y generador de formularios web se aplica la técnica de Pruebas de Funcionalidad, este tipo de técnicas concentra su atención en la validación de las funciones, métodos, servicios y casos de uso. Además asegura que el sistema sea utilizado solamente por los usuarios deseados.

Tipos de Prueba

Para garantizar la calidad del software se utiliza en la aplicación las Pruebas Funcionales, el objetivo fundamental es probar que el Diseñador y generador de formularios web, cumpla con las funciones específicas para los cuales han sido creados.

Métodos de Prueba

En la etapa de prueba existen dos métodos fundamentales Caja Negra y Caja Blanca. La prueba de caja negra es la seleccionada para garantizar la calidad de esta aplicación, se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Además los casos de prueba pretenden demostrar que las funciones del software son operativas. Se ejecuta cada caso de uso, usando datos válidos e inválidos, para verificar que se aplique para cada regla del negocio y que los resultados esperados ocurran cuando se introduzcan datos válidos.

Para el desarrollo de los casos de prueba de caja negra en la aplicación, se utiliza la Técnica Partición de Equivalencia, esta permite reconocer los valores válidos e inválidos de las entradas existentes en el Diseñador y generador de formularios web. Además divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.

Casos de Prueba

Los casos de pruebas son el conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo determinado, además deben de comprobar si el producto cumple con los requisitos del usuario.

Los casos de pruebas, para cada uno de los casos de uso planteados, a través de la matriz de datos, para comprobar que el Diseñador y generador de formularios web funcione de forma correctamente, se presentan a continuación:

- ✓ V: indica válido
- ✓ I: indica inválido
- ✓ NA: que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante
- ✓ V: representan valores de entrada de datos.
- ✓ Las variables V1, V2, V3, V4 que se representan en la siguiente tabla, significan los valores de entrada de datos para los casos de pruebas.

Tabla 7: Descripción de las variables para el caso de prueba Administrar diccionario de datos.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	-Nombre de la entidad.	-Campo de texto	No	-Cadena de caracteres, nombre de la entidad que se va a adicionar.
V2	-Nombre del atributo.	-Campo de texto	No	-Campo que admite solo caracteres, nombre del atributo que se va adicionar.
V3	-Tipo de dato.	- ComboBox	No	-Campo que tiene definido por defecto los tipos de datos de un atributo, este campo es obligatorio.
V4	-Longitud.	- Campo de texto	Si	-Campo que solo admite números, el campo es obligatorio según el tipo de dato seleccionado.

Tabla 8: Matriz de datos para el caso de uso Administrar diccionario de datos.

Escenario	Descripción	V1	V2	V3	V4	Respuesta del sistema	Flujo central
SC1.1. Adicionar entidad, con datos válidos.	En este escenario se adiciona una nueva entidad al diccionario de datos, a partir de los datos entrados por el usuario.	V	NA	NA	NA	La salida será una nueva entidad adicionada al diccionario de datos, a partir de los datos entrados por el usuario.	1. El usuario debe adicionar una nueva entidad, y de esta forma comienza a construir el modelo relacional o diccionario de datos. 2. El usuario debe adicionar un nuevo atributo a la entidad seleccionada. 3. Si desea eliminar alguna entidad del diccionario de datos, la selecciona y da click en el botón eliminar. 4. Si desea eliminar algún atributo del diccionario de datos, lo selecciona y da click en el botón eliminar.
		String					
SC1.2. Adicionar entidad, con datos inválidos.		I	NA	NA	NA	No se aceptan campos vacíos, el usuario no debe entrar ni números y caracteres especiales.	
		símbolos					
		I	NA	NA	NA		
		[ ]					
		I	NA	NA	NA		
		int					
SC1.3. Adicionar atributos con datos válidos.		NA	V	V	V	La salida será un nuevo atributo adicionado en la entidad seleccionada en el diccionario de datos, a partir de los datos entrados por el usuario.	
		NA	String	String	int		

SC1.4. Adicionar atributos con datos inválidos.		NA	I	I	I	El usuario siempre debe seleccionar un elemento del campo tipo de datos, además de no permitir que se agreguen números.	
			símbolos	[ ]	String		
		NA	I	NA	I		
			[ ]		símbolos		
		NA	I	NA	I		
			int		int		
SC1.5. Eliminar entidad con datos válidos.		NA	NA	NA	NA	Salida, eliminación de la entidad seleccionada en el diccionario de datos.	
SC1.6. Elimina atributos con datos inválidos.		NA	NA	NA	NA	Salida, eliminación del atributo seleccionado en el diccionario de datos	
		NA	NA	NA	NA		

Tabla 9: Descripción de variables para el caso de prueba Diseñar formularios.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	-Nombre del contenedor.	-Campo de texto	No	-Cadena de caracteres, nombre del contenedor que se va a diseñar.
V2	-Nombre del control básico.	-Campo de texto	No	-Cadena de caracteres, nombre del control básico que se va a diseñar
V3	-Seleccionar contenedor.	-Radio	No	-Opción obligatoria, adiciona un nuevo contenedor al área de

Tabla 10: Matriz de datos para el caso de uso Diseñar formulario.

Escenario	Descripción	V1	V2	V3	Respuesta del sistema	Flujo central
SC2.1. Diseñar contenedor con datos válidos.	En este escenario se adiciona un nuevo contenedor al área de trabajo, a partir de los datos entrados por el usuario.	V	NA	V	La salida será un nuevo contenedor adicionado al área de trabajo, a partir de los datos entrados por el usuario.	1. El usuario debe adicionar un nuevo contenedor, y de esta forma comienza a diseñar el formulario. 2. El usuario debe adicionar un nuevo control básico al contenedor de la entidad al cual pertenece.
		String				
SC2.2. Diseñar contenedor con datos inválidos.		I	NA	NA	No se aceptan campos vacíos, el usuario no debe entrar números, caracteres especiales y siempre debe seleccionar una de las opciones.	
		símbolos				
		I	NA	NA		
		[ ]				
		I	NA	NA		
		int				
SC2.3. Diseñar controles básicos con datos válidos.		NA	V	NA	La salida será un nuevo control básico adicionado en el contenedor correspondiente, a partir de los datos entrados por el usuario.	
			String			
SC2.4. Diseñar controles básicos con datos inválidos.		NA	I	NA	No se aceptan campos vacíos, el usuario no debe entrar números, caracteres especiales.	
			símbolos			
		NA	I	NA		
			[ ]			
		NA	I	NA		

			int			

Después de realizadas las pruebas de Caja Negra, se comprobó el correcto funcionamiento del Diseñador y generador de formularios web y la validación sus campos, comprobando que solo se acepten los caracteres válidos para los mismos. Cada conflicto detectado en la realización de la aplicación fue registrado en la planilla de No Conformidades. En la siguiente tabla se muestra un resumen de los conflictos encontrados.

Fecha	Versión	Casos de Prueba	Cant. de No Conformidad	Cant. de No Conformidad PD	Cant. De No Conformidad RA
23/05/2011	1.0	CU- Administrar diccionario de datos	3	-	3
25/05/2011	1.1	CU-Diseñar formularios	2	-	2

Tabla 11: Resumen de las no conformidades detectadas.

Para más información ver **(Anexo 1) y (Anexo 2).**

Aval

Se obtuvo un documento emitido por el Líder del proyecto Sistema de Información de Gobierno (SiGOB) avalando el aporte del sistema desarrollado para SiGOB y el departamento en general. Dentro de este:

- ✓ La solución de software desarrollada formará parte de la Solución Integral para la gestión de la información de gobierno desarrollada por DATEC bajo el nombre de SIGOB.
- ✓ Se implementaron los requisitos previstos en el alcance del trabajo de diploma, incluidos los artefactos de ingeniería requeridos por DATEC.
- ✓ Se realizaron con el rigor requerido las pruebas funcionales que validan la corrección de la solución, requisito para que la misma forme parte de los activos del Departamento de Integración de Soluciones de DATEC.
- ✓ Contribuye con la aceleración y humanización del desarrollo de formularios basado en ExtJS mediante la generación automática de código de JavaScript.

AVAL a Trabajos de Diplomas

STIÓN DE DATOS

Web: https://portal.datec.prod.uci.cu/ email: datec@uci.cu telef.: 837 3712, 837 3709

La Habana, 15 de junio de 2011
"Año 53 de la Revolución"

De: Ing. Armando Robert Lobo
Líder del proyecto Sistema de Información de Gobierno (SiGOB),
Arquitecto de Sistemas del Departamento de Integración de Soluciones (DATEC)

A: Miembros del Tribunal.

Por medio del presente documento certifico que la investigación realizada por el estudiante Yasniel Montano Rodríguez, tutorada por los Ing. Héctor Luis Reyes e Ing. Alberto Mendoza Garnache tributa al proyecto donde se desarrolla. A continuación se exponen los elementos que amparan la afirmación anterior:

1. La solución software desarrollada formará parte de la Solución Integral para la gestión de información de gobierno desarrollada por DATEC bajo el nombre de SiGOB.

2. Se implementaron los requisitos previstos en el alcance del trabajo de diploma, incluidos los artefactos de ingeniería requeridos por DATEC.

3. Se realizaron con el rigor requerido las pruebas funcionales que validan la corrección de la solución, requisito para que la misma forme parte de los activos del Departamento de Integración de Soluciones de DATEC.

4. Contribuye con la aceleración y humanización del desarrollo de formularios basado en ExtJS mediante la generación automática de código JavaScript.

Finalmente, se solicita adjuntar este documento al expediente de Tesis del estudiante. Para que así conste, firmo la presente, en un ejemplar, en la Universidad de las Ciencias Informáticas a los 15 días del mes de junio del año 2011.

Ing. Armando Robert Lobo
Líder del proyecto

UCI Universidad de las Ciencias Informáticas
Decanato Facultad 6

3.4 Conclusiones

- ✓ El uso de los patrones, ha facilitado el diseño de las clases permitiendo que se obtengan clases modulares, flexibles y reutilizables.
- ✓ La realización del diagrama de despliegue propició una visión de cómo está distribuido el sistema físicamente.
- ✓ Se estructuró el modelo de implementación a partir de los resultados del diseño.
- ✓ Se le realizaron pruebas de caja negra al sistema detectándose 5 no conformidades en los 5 casos de prueba que se le aplicaron. Las no conformidades encontradas fueron registradas y solucionadas para un mejor funcionamiento del software.

CONCLUSIONES

- ✓ A partir de un estudio de los aspectos principales sobre el diseño y generación de formularios web, se identificaron los elementos importantes del diseño a partir de un modelo relacional.
- ✓ A partir de los requisitos funcionales identificados se diseñó el componente diseñador y generador de formularios web a partir de un modelo relacional.
- ✓ Se implementó en el sistema un diseñador y generador de formularios web a partir de un modelo relacional para SIGOB.
- ✓ Se realizó la evaluación de las funcionalidades del sistema haciendo uso del método de prueba Caja Negra, garantizando la calidad de las funcionalidades del sistema.
- ✓ Como resultado de esta investigación se obtuvo un componente que permite agilizar el proceso de captura de datos a través del diseño y generación de formularios web.

RECOMENDACIONES

- ✓ Incorporar la comunicación entre el Diseñador y Generador de formularios web y el servidor para lograr:

    - Salvar los formularios web que fueron diseñados y el diccionario de datos que fue construido.
    - Implementar las funcionalidades importar y exportar un diccionario de datos.

- ✓ Incorporar la funcionalidad de asociar los elementos del diccionario de datos a los controles básicos de diseño utilizando las funcionalidades de arrastre de componente.

BIBLIOGRAFÍA

Arquitectura de Software Dirigida por Modelos.pdf.

B., Eduardo Abedrapo. Principales aspectos del Sistema Nacional de Inversiones de Chile.

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

Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. [En línea]

Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. [En línea]

Daniele, Marcela. Teoría 11: El Arte de Modelar UML. 2007.

Desarrollo de Software Dirigido por Modelos, conceptos teóricos y su aplicación práctica. Pons, Dra. Claudia y Giandini, Dra. Roxana . 2007.

Desarrollo de Software Dirigido por Modelos. manrubia Díez, Jorge y Cueva Lovelle, Juan Manuel . Diciembre 2006, Vol. Memoria Segundo Semestre. FICYT.

Desarrollo de Software Dirigido por Modelos-manrubia Díez, Jorge y Cueva Lovelle, Juan Manuel. Desarrollo de Software Dirigido por Modelos. Diciembre 2006.

dsadada. http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf. http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf. [En línea]

Eric Freeman, Elisabeth Freeman. Head First Design Patterns.

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Pattens-Elements of Reusable Object Oriented Software.

Fowler, Martin. Pattern of Enterprises Architecture.

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

http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf. [En línea]

http://www.exa.unicen.edu.ar/catedras/modysim/teoria/casos_de_uso_a.pdf. [En línea]

http://www.fi.unju.edu.ar/materias/materia/IS1/document/Clases_Teoricas_2010/Introduccion-2010.pdf. [En línea]

http://www.mitecnologico.com/Main/ModelosDeProcesoDeSoftware. [En línea]

http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html. [En línea]

http://www.upedu.org/process/artifact/ar_desmd.htm.

Ingeniería, Facultad de Informática-Universidad Politécnica de Madrid-Unidad Doc de. Patrones del “Grang of Four ”. España-Madrid : s.n.

Jaque, Miguel. http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio. [En línea] Diciembre del 2007.

Jaque, Miguel. http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio. [En línea] Diciembre del 2007.

Kiccillof, Carlos Reynoso y Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. s.l. : Universidad de Buenos Aires, 2004.

l.Jacobson, G.Booch, J.Rumbauggh. El Proceso Unificado de Desarrollo. s.l. : Addison Wesley, 2000.

L.Nahuel, L.Ocaranza, M.Pinasco. Herramientas de soporte al proceso de desarrollo dirigido por modelos . La Plata, Buenos Aires, Argentina : s.n.

La-Diversidad-de-Los-Objetos. http://es.scribd.com/doc/25171397/1-La-Diversidad-de-Los-Objetos-1-1-Caracteristicas. [En línea]

La-Diversidad-de-Los-Objetos-1-1-Caracteristicas. http://es.scribd.com/doc/25171397/1-La-Diversidad-de-Los-Objetos-1-1-Caracteristicas. [En línea]

Larman, Craig. UML y Patrones. . 1999.

Linet Lores Sánchez, Diana Monné Roque. Trabajo de Diploma: Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica (Junio 2009). junio 2009.

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

Marciniak, J.J. Process Models in Software Engineering. s.l. : 2nd Edition, John Wiley and Sons, New York, December 2001.

Marleysis López Duque, Raidel Oceguera Ravelo. Tesis: Herramienta en Matlab para la obtención de información de la base de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano. La Habana, Cuba : s.n., Junio, 2010.

MDA y el papel de los modelos en el proceso de desarrolo de software. Quintero, Juan Bernardo y Anaya, Raquel. Número 8, Colombia, Medelli : s.n., Diciembre 2007. ISSN 1794-1237.

MDA. http://es.scribd.com/doc/47014355/MDA. [En línea]

Modelos, Arquitectura de Software Dirigida por. Arquitectura de Software Dirigida por Modelos.pdf.

ModelosDeProcesoDeSoftware. http://www.mitecnologico.com/Main/ModelosDeProcesoDeSoftware. [En línea]

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

Object Management Group. [En línea] [Citado el: Diciembre 4, 2010.] http://www.uml.org.

Orellanos, Norah Velazco. Estadísticas de Inversión Extranjera Directa en los Países de la Comunidad Andina.

Ortiz, Kadir Hector. http://www.eumed.net/libros/2009c/583/Representacion%20del%20Modelo%20de%20Objetos%20de%20Dominio.htm. [En línea]

Perfiles UML y Desarrollo Dirigido por Modelos: Desafíos y Soluciones para Utilizar UML como Lenguaje de Modelado. Giachetti, Giovanni, Marín, Beatríz y Pastor, Oscar. Valencia, España : SISTEDES, 2008. ISSN 1988–3455.

PergaminoVirtual. [En línea] [Citado el: 26 de febrero de 2011.] http://www.pergaminovirtual.com/definicion/Formulario.html.

Prueba., Disciplina de. http://eva.uci.cu, Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. http://eva.uci.cu, Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. [En línea]

PruebaCasoDePrueba. http://www.mitecnologico.com/Main/PruebaCasoDePruebaDefectoFallaErrorVerificacionValidacion. http://www.mitecnologico.com/Main/PruebaCasoDePruebaDefectoFallaErrorVerificacionValidacion. [En línea]

Pruebas_Funcionales.pdf. http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf. http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf. [En línea]

requirements. http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf. [En línea]

Secretaria de Economia . [En línea] http://www.economia.gob.mx/swb/work/models/economia/Resource/516/1/images/EstadInverMexicoUE.pdf.

sesame. sesame. [En línea] [Citado el: 2 28, 2011.] http://www.openrdf.org/doc/sesame/users/.

Systems, Por Popkin Software and. Modelado de Sistemas com UML.

TiposPruebasSoftware. http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf. [En línea].

tutorial2. http://www.adrformacion.com/cursos/metod5s/leccion1/tutorial2.html. [En línea]

UCI. Entorno Virtual de Aprendizaje curso de Historia de la Informatica. [En línea] http://eva.uci.cu/mod/resource/view.php?id=9287&subdir=/Temas_Generales.

visual-paradigm. http://www.visual-paradigm.com/product/vpsuite/. [En línea]

www.mastermagazine.info/termino/7006.php. http://www.mastermagazine.info/termino/7006.php. [En línea]

www.slideshare.net/guest83f0d26/mda-2596889. http://www.slideshare.net/guest83f0d26/mda-2596889. [En línea]

TRABAJOS CITADOS

1. PergaminoVirtual. [En línea] [Citado el: 26 de febrero de 2011.] http://www.pergaminovirtual.com/definicion/Formulario.html.
2. **Grady, Booch, Ivar, Jacobsony James, Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* 2000.
3. CBASQA-Desarrollo de Software, SQA. [En línea] [Citado el: 20 de Noviembre de 2010.] http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.
4. [En línea] [Citado el: 22 de Diciembre de 2010.] http://msdn.microsoft.com/es-es/library/bb972242.aspx.
5. **Kiccillof, Carlos Reynoso y Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : Universidad de Buenos Aires, 2004.
6. Object Management Group. [En línea] [Citado el: 4 de Diciembre de 2010.] http://www.uml.org..
7. **Larman, Craig.** *UML y Patrones. .* 1999.
8. New Releases and Feature Enhancements of Visual Paradigm Products. [En línea] [Citado el: 4 de Diciembre de 2010.] http://www.visual-paradim/product/news/#VPUML50.
9. **Mabel Navarro Bermúdez, Yissel Rodríguez.** *BioSyS: Implementación del Módulo de Simulación.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

ANEXOS

Anexo 1: Tabla de No conformidades para el caso de prueba Administrar diccionario de datos

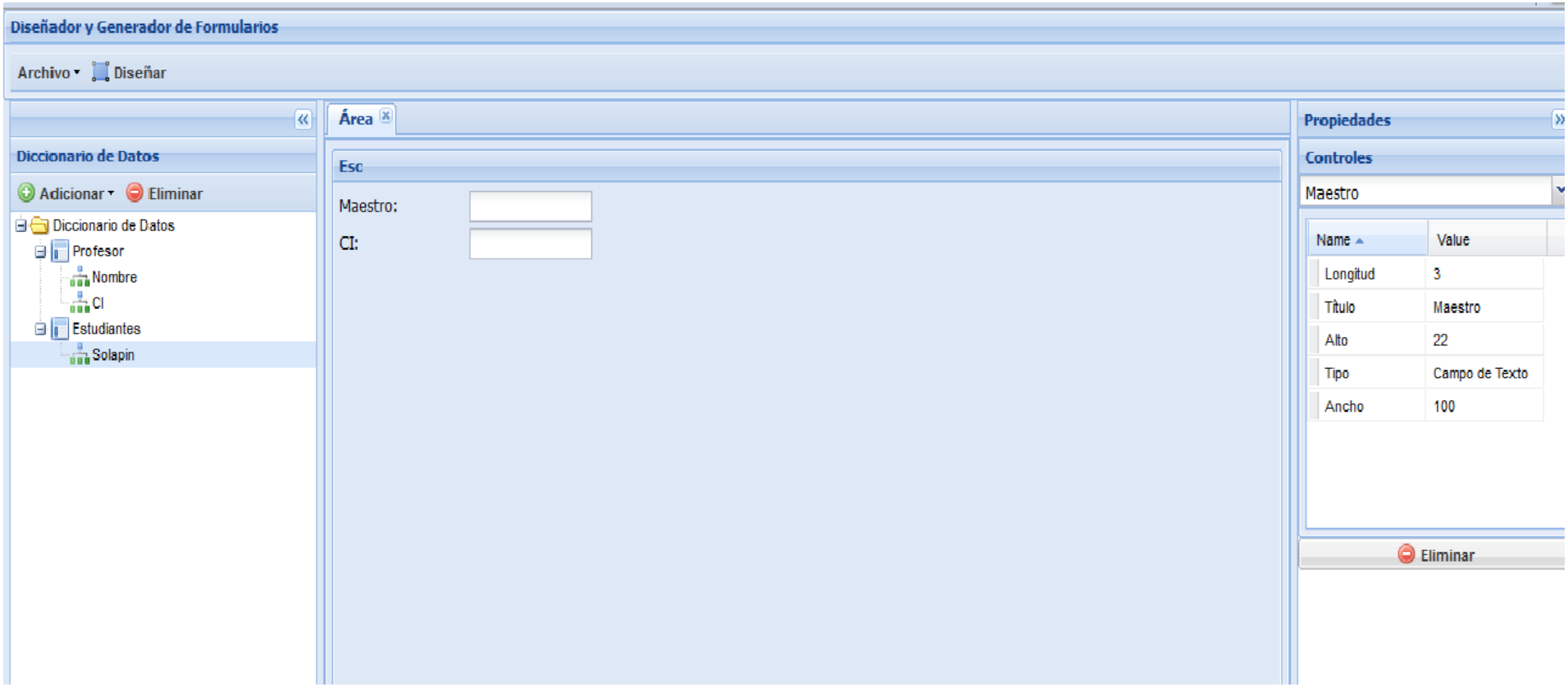
Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo
Componente	1	En el diccionario de datos, se acepta que se introduzcan entidades con el mismo nombre.	CU1. Administrar diccionario de datos, ejecutar el paso 1 (Adicionar una entidad al diccionario de datos).	Prueba a nivel de desarrollador utilizando la técnica de Prueba Funcional.	X			23/5/2011 Pendiente 5/6/2011 RA	Se solucionó la dificultad detectada.
Componente	2	Se admiten campos vacíos.	CU1. Administrar diccionario de datos, ejecutar el paso 1 (Adicionar una entidad al diccionario de	Prueba a nivel de desarrollador utilizando la técnica de Prueba	X			25/5/2011 PD Pendiente 2/6/2011 RA	Se solucionó la dificultad detectada.

			datos).	Funcional.					
Componente	3	Cuando se elimina un elemento del diccionario de datos no elimina los controles asociados a el que fueron diseñados.	CU1. Administrar diccionario de datos, ejecutar el paso 3 y 4 (Eliminar una entidad o un atributo al diccionario de datos).	Prueba a nivel de desarrollador utilizando la técnica de Prueba Funcional.	X			30/5/2011 PD Pendiente 6/6/2011 RA	Se solucionó la dificultad detectada.

Anexo 2: Tabla de No conformidades para el caso de prueba Diseñar formularios.

Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo
Componente	1	En área de trabajo, se acepta que se adicione más de un contenedor.	CU1. Diseñar formularios, ejecutar el paso 1 (Adicionar una entidad al diccionario de datos).	Prueba a nivel de desarrollador utilizando la técnica de Prueba	X			28/05/2011 1 Pendiente 5/6/2011 RA	Se solucionó la dificultad detectada.

				Funcional.					
Componente	2	Se admiten campos vacíos cuando se introduce el nombre de un control básico.	CU1. Diseñar formularios, ejecutar el paso 1 (Adicionar una entidad al diccionario de datos).	Prueba a nivel de desarrollador utilizando la técnica de Prueba Funcional.	X			02/06/2011 PD Pendiente 2/6/2011 RA	Se solucionó la dificultad detectada.



Anexo 3: Imagen de la aplicación.