Universidad de las Ciencias Informáticas

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

Módulo de Reportes

del Sistema Único de Identificación Nacional

Autor

Alejandro Torres Veiga

Tutores

Gleydis Claro Morgado

Erick de la Vega García

Ciudad de la Habana, Junio de 2010

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.

Albert Einstein

Declaración de Autoría

Declaro ser autor de este trabajo y autorizo a la Universidad de la Ciencias Informáticas y a la Dirección de Informática y Comunicaciones del Ministerio del Interior a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los __ días del mes __ del 2010

_____
Firma del Autor
Alejandro Torres Veiga

_____
Firma del Tutor
Ing. Gleydis Claro Morgado

_____
Firma del Tutor
Msc. Erick de la Vega García

Tutor: Msc. Erik de la Vega García
Ingeniero en Ciencias Informáticas, UCI 2007, Título de Oro.
Máster en Informática Aplicada
Categoría docente: Profesor Instructor
Líder de desarrollo del proyecto Sistema Único de Identificación Nacional.
Correo electrónico: edelavega@uci.cu

Tutora: Ing. Gleydis Claro Morgado
Ingeniera Informática, UCI, 2009.
Categoría docente: Adiestrada
Trabaja en el Departamento de Automatización de la Dirección de Identificación y Registro del Ministerio del Interior.
Analista de sistema del Sistema de Identificación, Migración y Extranjería.
Correo electrónico: gclaro09@graduados.uci.cu

A la memoria de mi abuela, por todo el amor que me profesó, por todo lo que soy…

A mis padres, mi luz, mi guía, mi ejemplo…

Agradezco a:

A mis padres por su amor y su cariño, por ser mi guía y ejemplo, por ser lo más grande del mundo.

A mis tutores por su tiempo y dedicación, por guiarme en la realización de este sueño.

A mi familia por su apoyo incondicional.

A mis amigos, mi otra familia, por estar siempre presente, en las buenas y en las malas.

A Yadira por su apoyo, gracias por ser la mejor compañera de tesis.

A Karmin por su comprensión y cariño.

A todos los que en un momento u otro me tendieron una mano.

Alejandro

Resumen

Actualmente, el mercado de aplicaciones informáticas es tremendamente variado y abarca un sin número de utilidades para desarrollar tareas en todos los campos activos de la sociedad, donde la informática juega un papel fundamental.

Como parte del proceso de informatización de los organismos encargados de la identificación de los ciudadanos en el territorio nacional, el Ministerio del Interior se ha propuesto la creación de un software para la Dirección de Identificación y Registro (DIR) que garantice mayor seguridad, una mejor calidad de los documentos de identificación y un registro único de la población.

En el presente trabajo de diploma se presenta un Módulo de Reportes, perteneciente al Sistema Único de Identificación Nacional, que tiene como objetivo desarrollar un software que procese la información almacenada por el sistema, de forma tal que facilite su análisis y sirva como herramienta de apoyo para la dirección de la organización en el proceso de toma de decisiones.

Palabras claves: aplicaciones informáticas, documentos de identificación, software.

Índice de figuras

Índice de tablas

Introducción

Con el triunfo de la Revolución Cubana y el aumento de la población surge la necesidad de crear en Cuba un sistema de identidad seguro. Su impacto en las actividades de la sociedad fue enunciado desde inicios del siglo pasado por Fernando Ortiz, momento que puede ser citado como punto de partida para la aparición de diferentes regulaciones relativas a esta actividad en el territorio nacional.

Desde el año 1997, en la República de Cuba, se otorga la gestión y control de los procesos relativos a la identificación de personas a la Dirección de Identificación y Registro (DIR) del Ministerio del Interior (MININT), la cual posee como responsabilidad primaria establecer las políticas y estrategias de acción para controlar y coordinar las actividades de los ciudadanos dentro del país.

En la actualidad, gran parte del flujo de trabajo es realizado con un nivel de informatización escaso o manual, lo que ha obstaculizado el correcto funcionamiento del mismo. Durante el último período del 2008 y comienzos del 2009 se ha realizado un estudio del Sistema Nacional de Identificación de la Población en favor de garantizar documentos y procesos seguros, con el fin de obtener un registro único.

Con tal motivo se sostuvieron intercambios con la Universidad de las Ciencias Informáticas, institución comprometida con la informatización de la sociedad. De este modo surge el proyecto Identidad Cuba, encargado de desarrollar el SUIN (Sistema Único de Identificación Nacional), cuyo objetivo primordial es la reestructuración, modernización y automatización de todos los procesos relacionados con las gestiones realizadas por las oficinas de Carné de Identidad y Registro de la Población (CIRP).

Para la realización de este software se desarrolló un estudio con el fin de determinar los principales problemas que afectan a las oficinas de CIRP a partir del cual fueron detectadas la excesiva demora de los trámites, la mala atención a la población y la falta de interés de los funcionarios en la realización de su trabajo. Entre las causas que provocan estos problemas, pueden mencionarse la escasa supervisión que existe sobre los puestos de trabajo y la desorganización de los mecanismos automatizados de control sobre el trabajo de la institución.

Actualmente los controles establecidos por la DIR acerca del desempeño de los funcionarios y en general de los resultados de la organización, es a través de reportes estadísticos y trazas, generadas por los sistemas que se utilizan en las oficinas de CIRP para la realización de los diferentes tipos de trámites. Estos reportes son estáticos y carecen de información visual, lo que provoca que su procesamiento y análisis se tornen complejos; además de que al ser generados por distintas aplicaciones ocasiona que la información sea discordante y eventualmente inconsistente.

Esto a su vez impacta en los resultados de la organización, al no tener la dirección de la misma información coherente y confiable que sirva de apoyo a la toma de decisiones en cada uno de los niveles directivos.

Derivado de esta **situación problémica** que presenta el Sistema de Identificación Nacional se infiere el siguiente **Problema Científico**:

¿Cómo proveer un mecanismo que procese la información almacenada por el Sistema Único de Identificación Nacional de la población de la República de Cuba con el objetivo de facilitar su análisis?

Partiendo de este problema se tiene como **objeto de estudio** los módulos de reportes en sistemas de gestión de la información y el **campo de acción** lo constituye el Módulo de Reportes del Sistema Único de Identificación Nacional de la República de Cuba.

Para resolver el problema planteado con anterioridad se propone como **objetivo general**: Desarrollar un Módulo de Reportes para el Sistema Único de Identificación Nacional de la población de la República de Cuba.

El citado objetivo general se desglosó en otros más específicos:

- ✓ Desarrollar el marco teórico de la investigación.
- ✓ Realizar el modelado del negocio para el Módulo de Reportes.
- ✓ Obtener requisitos funcionales y no funcionales para el Módulo de Reportes.
- ✓ Realizar el diseño del Módulo de Reportes.
- ✓ Realizar la implementación del software.
- ✓ Garantizar la calidad del software.

Para encaminar la investigación en vista a resolver el problema planteado se propone la siguiente **hipótesis**: Con la implementación de un Módulo de Reportes en el Sistema Único de Identificación Nacional, se facilitará el análisis de la información por parte de la dirección de la organización en cada uno de sus niveles.

Variables:

Independientes:

Módulo de Reportes en el Sistema Único de Identificación Nacional

Dependientes:

Análisis de la información.

Para satisfacer los objetivos planteados se realizaron las siguientes tareas de la investigación:

- ✓ Realización de un estudio de los sistemas existentes de reportes.
- ✓ Realización de un estudio bibliográfico permita fundamentar las herramientas informáticas y metodologías utilizadas para desarrollar dicha aplicación.
- ✓ Identificación de los procesos de negocio.
- ✓ Modelación y descripción de los procesos de negocio.
- ✓ Aplicación de técnicas de recopilación de información a los proveedores de requisitos.
- ✓ Especificación de los requisitos funcionales y no funcionales del software.
- ✓ Realización de los prototipos de interfaz de usuario.
- ✓ Realización de la validación de los requisitos.
- ✓ Ejecución del diseño del software.
- ✓ Implementación del Módulo de Reportes del Sistema Único de Identificación Nacional.
- ✓ Ejecución de las pruebas de aceptación.

Los métodos científicos utilizados en la investigación fueron:

Métodos Teóricos:

Analítico-Sintético: Mediante este método se analizó toda la teoría recopilada a través de los diferentes medios bibliográficos, que puedan servir para desarrollar mejor el diseño del sistema, y poder aplicar así estos conocimientos en la práctica de manera que se adquiera una mayor preparación sobre el tema de los reportes.

Modelación: Se llevó a cabo una modelación del objeto pues la modelación de un proceso permite predecir la respuesta de dicho proceso a variaciones de algunos de sus parámetros, sin tener que ejecutar el proceso en la realidad.

Hipotético-Deductivo: A partir de la hipótesis y siguiendo la lógica de deducción tomada se llegó a nuevos conocimientos y predicciones, que fueron sometidos a verificaciones.

Métodos empíricos: Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional. Estos métodos son empleados durante todo el proceso de implementación.

Observación: Permitió analizar cada fase del proceso e ir observando cada tarea que se realizó y tomar experiencia de esta para aplicarla en todas las demás. Se empleó en todo momento.

Entrevista: Permitió entrevistar a varias personas con amplios conocimientos sobre el tema.

El presente documento consta de cuatro capítulos:

Capítulo 1 Fundamentación Teórica: Se describen los conceptos fundamentales asociados al dominio del problema y el objeto de estudio, realizándose un análisis de la situación actual. Se presenta la fundamentación de las tecnologías utilizadas para el diseño del sistema y las propuestas para su implementación y desarrollo.

Capítulo 2 Características del sistema: Se realiza un análisis de las características del sistema a desarrollar a partir de la problemática planteada, se describen las principales funcionalidades que brindará el Módulo de Reportes a desarrollar e identifican a los trabajadores que intervienen.

Capítulo 3 Análisis y diseño del sistema: Se realiza una descripción de los principales elementos que componen la arquitectura y que guían el proceso de construcción del software. Se definen las clases que intervienen en la solución, así como los servicios que dan soporte a las funcionalidades y los diseños de *workflow*. También se obtienen los artefactos generados a partir de la aplicación de la metodología de desarrollo utilizada y una descripción detallada de las entidades que tienen persistencia en la base de datos.

Capítulo 4 Implementación y Prueba: Se muestra el modelo de implementación además del diagrama de despliegue. También se llevan a cabo las diferentes pruebas sobre la aplicación para determinar el nivel de calidad y eficiencia de la misma.

Capítulo1: Fundamentación Teórica

Introducción:

En el presente capítulo se describen los conceptos fundamentales asociados al dominio del problema y el objeto de estudio, haciéndose un análisis de la situación actual. Se presenta la fundamentación de las tecnologías utilizadas para el diseño del sistema y las propuestas para su implementación y desarrollo.

1.1 Conceptos fundamentales asociados al dominio del problema

Reporte: Un reporte es un documento, generado por un Sistema, que nos presenta de manera Estructurada y/o Resumida, datos relevantes guardados o generados por una aplicación.

El ciclo de vida de un reporte está formado por dos estados fundamentales: la definición del informe y la representación de este.

Una definición de informe es un archivo que se crea mediante un diseñador de informes o un generador de informes. Esta definición proporciona una descripción completa de conexiones de orígenes de datos, consultas utilizadas para recuperar datos, expresiones, parámetros, imágenes, cuadros de texto, tablas y cualquier otro elemento de tiempo de diseño se que pueda incluir en el informe.

Un informe representado es un informe totalmente procesado que contiene datos e información de diseño en un formato que permite su visualización (por ejemplo, HTML[1]). Un informe no puede visualizarse hasta que no se haya representado en un formato de salida.

Un informe está compuesto de un encabezado, un cuerpo y un pie de página. Puede incluir imágenes, cuadros de texto y líneas en los encabezados y pies de página. El cuerpo del informe contiene los datos del informe. El informe puede contener cualquier tipo de elemento de informe en el cuerpo, por ejemplo tablas, matrices, listas, gráficos y medidores.

Un informe básico debe especificar qué datos necesita, cómo desea organizarlos en la página y cómo desea que los usuarios vean el informe.

[1] ***H**yper**T**ext **M**arkup **L**anguage*

La creación de un informe consta de varios pasos fundamentales entre los que se encuentran definir los datos que se utilizarán en el informe, agregar regiones de datos a la superficie de diseño del informe y crear o modificar parámetros, para proporcionar interactividad a los usuarios.

La definición de origen de datos especifica el tipo de origen de datos, una cadena de conexión y las credenciales. Cada tipo de origen de datos requiere información de conexión diferente. Las definiciones de origen de datos pueden almacenarse en un servidor de informes y administrarse como archivos independientes o incrustarse en la definición de informe.

Organizar los datos del informe en la superficie de diseño se puede realizar agregando los elementos de informe siguientes: regiones de datos, imágenes, líneas, rectángulos, cuadros de texto y subinformes. Los elementos de informe son elementos de diseño que están asociados a distintos tipos de datos de informe. Tabla, matriz, lista, gráfico y medidor son elementos de informe de la región de datos, cada uno de los cuales establece un vínculo al conjunto de datos de informe. La organización de los datos del informe establece la forma en que el usuario visualizará la información requerida.

La creación de parámetros es una actividad opcional que proporcionar interactividad a los usuarios. Los informes con parámetros utilizan valores de entrada para completar el procesamiento del informe o de los datos. Con un informe con parámetros, puede modificar el resultado del informe en función de los valores que se establecen cuando se ejecuta el mismo. Los informes con parámetros se suelen utilizar para informes de obtención de detalles, informes vinculados y subinformes, y conectan y filtran los informes con los datos relacionados.

Entre las herramientas más conocidas para la generación de informes se encuentran Crystal Report y Active Report.

1.2 Análisis de otras soluciones existentes

Actualmente, el mercado de aplicaciones informáticas es tremendamente variado y abarca un sin número de utilidades para desarrollar tareas en todos los campos activos de la sociedad, donde la informática juega un papel fundamental. Hoy en día, el desarrollo de las organizaciones demanda una enorme cantidad de información, de ahí que las empresas están obligadas a tomar decisiones cada vez más precisas y con mayor rapidez.

Muchas veces se realizan sistemas de información que se dedican a capturar datos, pero que también necesitan una manera de procesarlos y mostrarlos. Es importante mencionar que los datos almacenados son útiles en la misma medida que se puedan convertir en información para las personas que los necesitan. Para esta tarea entran en juego los reportes, que no son más que objetos que entregan información en un formato particular y permiten realizar ciertas operaciones como imprimirlos, enviarlos por correo electrónico, guardarlos a un archivo, etc.

A continuación se muestran varias aplicaciones que hacen uso de los reportes para una mejor apreciación de los datos con que trabajan.

Info@tletas: Aplicación Web Oficial del Instituto Nacional de Deportes, Educación Física y Recreación de Cuba (INDER) para la administración de los expedientes técnico-acumulativos de atletas y entrenadores nacionales e internacionales. La aplicación cuenta con 12 módulos, entre los que se incluye el Módulo de Reporte. Cada módulo tiene una parte administrativa y una pública. En cada pedido la plataforma gestiona la cantidad de datos necesaria y suficiente evitando así que tanto el canal de conexión a la base de datos, como el establecido entre el servidor web y el cliente, se congestionen. Esto trae consigo un mayor rendimiento de la aplicación y el ahorro de recursos técnicos. (1)

Módulo de Reportes del Atta en la Web. Este módulo permite consultar a través de Internet la base de datos de especímenes del Instituto Nacional de Biodiversidad (INBio). Esta base de datos contiene la información correspondiente a todos los especímenes que el instituto ha recolectado y estudiado desde su fundación en 1989 hasta hoy.

Durante su recolección y posterior estudio a cada espécimen se le asocia información taxonómica, geográfica y biológica. En la base de datos esta información se almacena en diferentes campos que en los reportes son desplegados en un formato de columnas. Esta estructura permite seleccionar para su despliegue solamente aquellos especímenes que tengan un valor determinado en una columna o bien una combinación de valores en varias columnas.

Las consultas realizadas al módulo de reportes devuelven como resultado un conjunto de especímenes filtrado según los criterios que el usuario haya indicado y que incluye todas las columnas que éste considere convenientes. Cada registro del reporte corresponde a un espécimen y cada columna presenta un dato relativo a éste. (2)

SAIME (Servicio Administrativo de Identificación, Migración y Extranjería): Este sistema tiene como objetivo fundamental la reestructuración, modernización y automatización de todos los procesos que se desarrollan en la ONIDEX (Oficina Nacional de Identificación y Extranjería), de la República Bolivariana de Venezuela. Entre sus módulos se encuentra un Módulo de Reportes, encargado de trabajar con los reportes generados por la aplicación. Este módulo cuenta con una parte administrativa y una pública. La pública permite ver todos los reportes que han sido elaborados para los diferentes procesos y trámites. Desde aquí se puede acceder a cualquier reporte de la aplicación. Ver Fig. 1 Ventana que muestra los diferentes reportes de la aplicación.

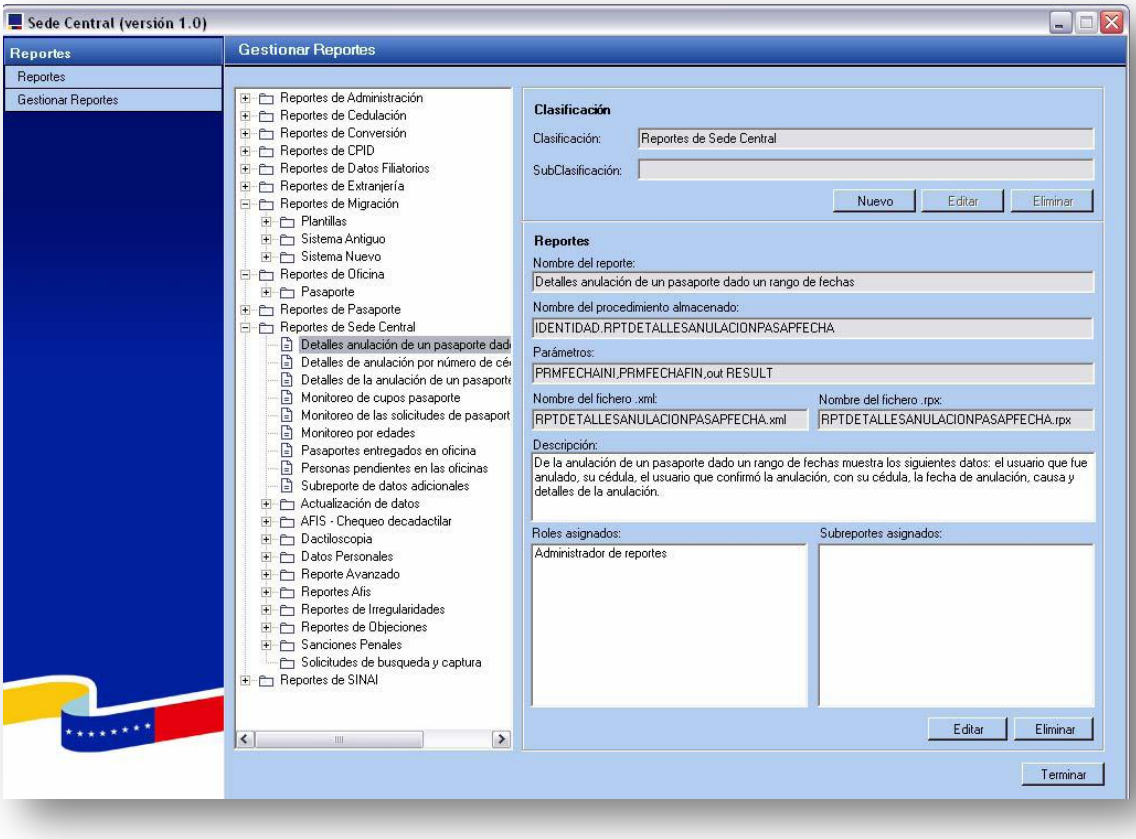


Fig. 1 Ventana que muestra los diferentes reportes de la aplicación

La parte administrativa permite la creación de un nuevo reporte o de una nueva clasificación. Una clasificación agrupa varios reportes en dependencia de las características que tenga. Comúnmente una clasificación tiene el nombre del módulo o sistema al cual se le generarán los diferentes reportes.

La creación de reportes permite el diseño del formulario de captura de parámetros, el diseño del reporte y la asignación de los roles que pueden visualizar el reporte, además de la asignación de distintos subreportes. En la Fig. 2 Ventana que permite la creación de un nuevo reporte. (3)

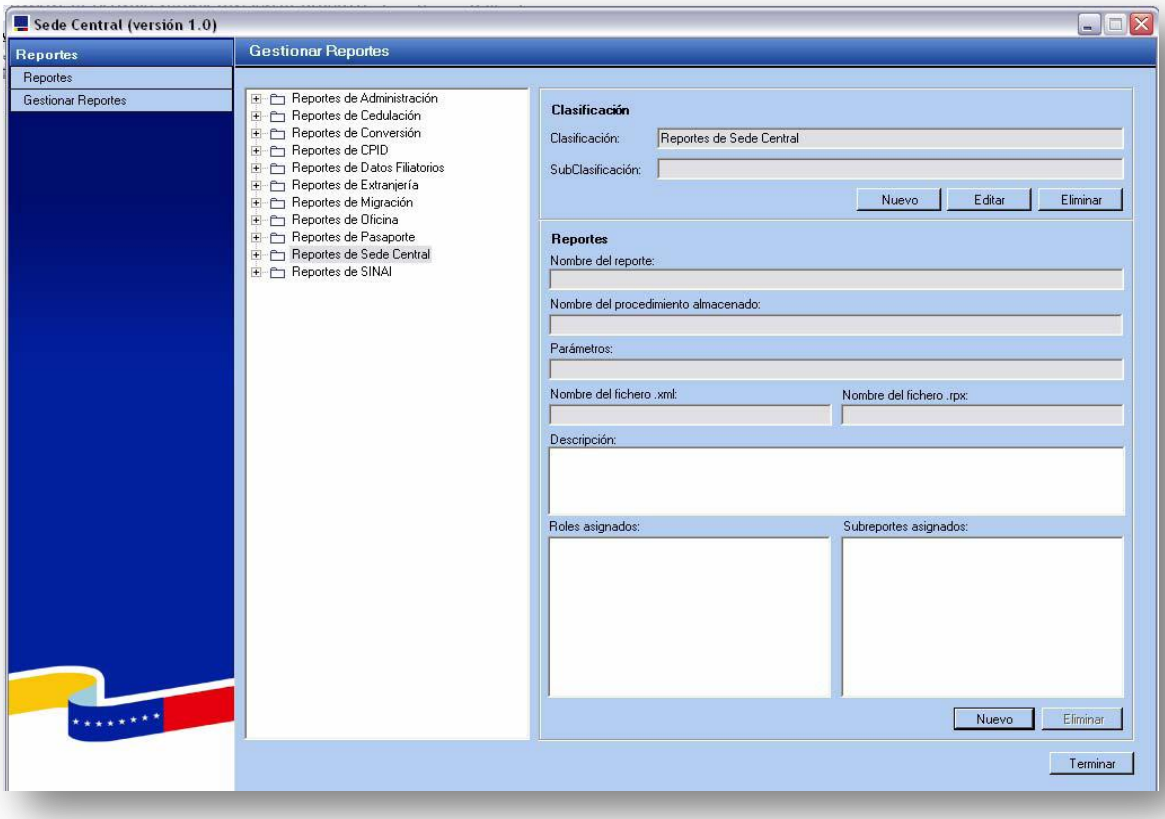


Fig. 2 Ventana que permite la creación de un nuevo reporte

SACIRP (Sistema Automatizado para el Carné de Identidad y Registro de la Población)

El Sistema Automatizado para el Carné de Identidad y Registro de la Población (SACIRP) pretende facilitar la actualización de la información del Carné de Identidad y Registro de la Población. Entre sus principales prestaciones se pueden citar que garantiza la seguridad informática a nivel de usuario, a partir de que individualiza el trabajo según el rol asignado, permitiendo conocer las acciones realizadas por el mismo y crear trazas de las acciones que se efectúan en la aplicación. Además permite la generación de diferentes reportes estadísticos referentes a la gestión de los usuarios a nivel de unidad o

de algunos tipos de trámites como por ejemplo la cantidad de transitorias o de fallecidos que fueron tramitados en el día. (4)

SAAP (Sistema Automatizado de Atención a la Población)

Este sistema está diseñado para automatizar los procesos de trabajo que van desde la información que se le brinda a la ciudadanía hasta la entrega del documento de identidad, o satisfacer la demanda del servicio que la misma solicite, transitando en el intermedio por la captación de la información y la realización del trámite solicitado. Permite la generación de diferentes reportes estadísticos referentes al trabajo realizado por los usuarios del sistema en un rango de fechas determinado. En la Fig. 3 Reportes en el SAAP se muestran algunas pantallas de los reportes del sistema. (5)

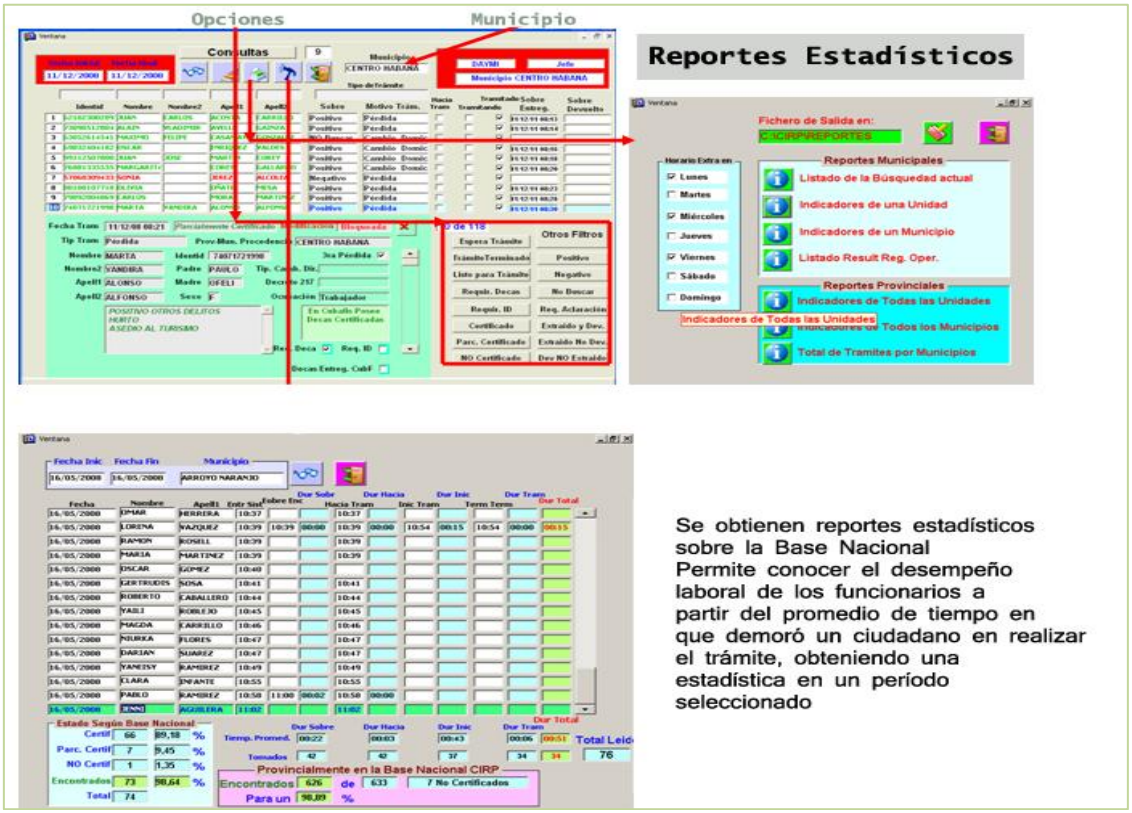


Fig. 3 Reportes en el SAAP

1.2.1 Inconvenientes de los sistemas analizados

Todos los sistemas anteriormente analizados, de forma general, procesan los datos almacenados a través del uso de los reportes para una mejor apreciación de la información con que trabajan. A continuación se enuncian varias inconformidades detectadas a partir de las que se ha decidido la creación de un nuevo Módulo de Reportes para el SUIN, de manera tal que contenga las funcionalidades requeridas por el cliente.

Info@tletas

El módulo de reportes de info@tletas se especializa en la visualización de reportes sobre el rendimiento de los atletas y entrenadores, tanto nacionales como internacionales, en un período de tiempo determinado y no en la visualización de reportes sobre los trámites realizados en las oficinas de Carné de Identidad y Registro de la Población, razón por la cual fue fácil de descartar, ya que no cumplía con las necesidades del Módulo de Reportes para el SUIN. No obstante, los reportes que realizan sobre el rendimiento de los atletas sirvieron como guía para el diseño de los reportes sobre el rendimiento de los funcionarios de las oficinas de carné.

Módulo de Reportes del Atta en la Web

Las consultas realizadas al módulo de reportes de Atta en la Web devuelven como resultado un conjunto de especímenes filtrado según los criterios que el usuario haya indicado y que incluye todas las columnas que éste considere convenientes, muy distante de las necesidades del Módulo de Reportes para el SUIN por lo que fue descartado como posible solución a utilizar.

Sistema SAIME

Este sistema fue pensado y modelado según las especificaciones dictadas por el cliente y las particularidades de Venezuela, país en el cual fue desplegado. No fue elaborado para uso de manera general en cualquier ambiente, aunque cabe reconocer que por la similitud de los procesos que realiza, sirvió como guía en el desarrollo de la solución propuesta.

SACIRP (Sistema Automatizado para el Carné de Identidad y Registro de la Población)

Este sistema permite la generación de diferentes reportes estadísticos referentes a la gestión de los usuarios a nivel de unidad y de algunos tipos de trámites, pero no de todos los trámites que se llevan a

cabo en las oficinas, ni del trabajo de los funcionarios, así que se puede afirmar que no cumple con los requerimientos del Módulo de Reportes. Cabe reconocer que su estudio fue necesario para una mayor comprensión de los reportes relacionados con el trabajo de los funcionarios.

SAAP (Sistema Automatizado de Atención a la Población)

Este sistema permite la generación de diferentes reportes estadísticos referentes al trabajo realizado por los usuarios del sistema en un rango de fechas determinado, pero no muestra reportes sobre los trámites que se llevan a cabo en las oficinas de carné, así que al igual que el SACIRP, no satisface a cabalidad las necesidades requeridas, por lo que solo se utilizó como guía en la creación de los reportes del Módulo de Reportes del SUIN.

1.3 Tendencias Tecnológicas

1.3.1 Desarrollo orientado a procesos.

Las exigencias del mercado y la presión de la competencia obligan a las organizaciones a ser más eficaces y eficientes en todas las áreas de servicio al cliente, producción, servicios internos y control. Con el objetivo de mejorar el tiempo total del proceso que realizan, mantener un mejor registro de la información, proporcionar elementos de control y brindar un mejor servicio, frecuentemente las empresas desarrollan sistema orientados a procesos, ya que saben que los recursos bien integrados y orquestados, que permitan una verdadera agilidad, son los que hacen a las organizaciones ser más competitivas. (6)

El desarrollo orientado a procesos posibilita una visión más clara del funcionamiento de la institución, permitiendo el seguimiento del flujo y el control de acceso a nivel de cada operación que se realice, además de la modificación o incorporación de nuevos procesos con un alto grado de reutilización de los componentes existentes. Una forma de llevar a cabo este desarrollo es a través de sistemas basados en *workflows*[2].

El término *workflow* se refiere a toda un área dentro de la informática cuyo objetivo es el modelado y automatización de procesos que manejan información normalmente no estructurada o muy poco estructurada dentro de una organización, o incluso entre distintas organizaciones. Esta información,

[2] Flujo de trabajo

habitualmente en forma de documentos, fluye a través de un grupo de personas y/o máquinas que participan en el proceso trabajando sobre dichos documentos. (7)

La principal ventaja que permiten los sistemas basados en *workflow,* es la facilidad con que asimilan variaciones ocurridas en los procesos modelados. A diferencia de lo que puede ser un proceso rígido con pocas posibilidades de variar, los procesos de *workflow* son propicios al cambio. Esta característica se debe principalmente a la alta participación de personas en tales procesos las cuales pueden ser reubicadas, ascendidas o suprimidas en el organigrama de una organización, implicando de esta manera la variación temporal o definitiva de un determinado proceso. También la búsqueda constante de mejora en los tiempos de procesos por parte de la organización hace que los mismos sean modificados para eliminar “cuellos de botella” detectados luego de modelarlos. (8)

Teniendo en cuenta lo antes expuesto el equipo de trabajo decidió optar por un desarrollo orientado a procesos, de forma tal que la aplicación no se vea seriamente afectada en caso de que ocurran variaciones en los procesos que se llevan a cabo actualmente en la institución.

1.3.2 Metodología de desarrollo.

Una metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Principalmente se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente. (9)

Para guiar el desarrollo del software se utilizará MSF for CMMI, MSF ya que es la metodología nativa para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías Microsoft y CMMI buscando un lineamiento con las buenas prácticas establecidas por la Universidad de las Ciencias Informáticas para el desarrollo de software.

Microsoft Solutions Framework (MSF)

MSF es un marco de trabajo de referencia para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft. MSF comprende un conjunto de modelos, conceptos y guías que contribuyen a alinear los objetivos de negocio y tecnológicos, reducir los costos

de la utilización de nuevas tecnologías, y asegurar el éxito en la implantación de las tecnologías Microsoft. MSF es el resultado de las experiencias de diferentes áreas de Microsoft con proyectos exitosos. (9)

Capability Maturity Model Integration (CMMI)

CMMI es un acercamiento a la mejora de procesos que provee a las organizaciones de los elementos esenciales para ejecutar procesos eficaces. Este puede ser usado para guiar la mejora de procesos a través de la vida de un proyecto, una división o una organización completa. CMMI establece un conjunto de buenas prácticas para el desarrollo y mantenimiento de productos y servicios, cubriendo el ciclo de vida de un producto desde su concepción hasta su entrega y mantenimiento. (11)

MSF for CMMI permite el acercamiento a una nueva dimensión, donde el desarrollo integrado fortalece el desarrollo de aplicaciones. Con el uso de la herramienta Visual Studio 2008 se han unificado dos metodologías que organizan y controlan los procesos de desarrollo de software permitiendo la supervisión de los códigos implementados así como la documentación generada como parte del proceso de desarrollo de software. Esta metodología establece un proceso disciplinado de desarrollo de software que lo hace más predecible y eficiente.

1.3.3 Herramientas para el modelado de procesos del negocio

Frecuentemente los sistemas (conjuntos de procesos y subprocesos integrados en una organización) son difíciles de comprender, amplios, complejos y confusos; con múltiples puntos de contacto entre sí y con un buen número de áreas funcionales, departamentos y puestos implicados. Con el objetivo de organizar y documentar la información sobre un sistema se crea un modelo de negocio que no es más que una representación de una realidad compleja. Modelar es desarrollar una descripción lo más exacta posible de un sistema y de las actividades llevadas a cabo en él.

A través del modelado de las actividades y procesos puede lograrse un mejor entendimiento del negocio y muchas veces esto presenta la oportunidad de mejorarlos. Es por eso que una vez definida la metodología a utilizar el equipo de trabajo comienza el modelado de los procesos de negocio de la organización donde se utilizará el software. Para llevar a cabo esta tarea se utilizó la herramienta Altova UModel.

Altova UModel 2009

Altova UModel 2009 es el punto de partida para el desarrollo acertado de software. UModel se utiliza para crear e interpretar diseños de software del estándar de UML 2.1. Diseña visualmente modelos de aplicaciones en UML[3], genera código Java, C#, o Visual Basic .NET, documentación del proyecto y realiza ingeniería inversa de los programas existentes en diagramas claros y exactos de UML para comprender rápidamente la arquitectura del software.

UModel 2009 combina una rica interfaz visual con funciones de usabilidad superiores para ayudar a nivelar la curva de aprendizaje de UML, además de incluir las más altas funcionalidades para potenciar a los usuarios con las más completas ventajas del desarrollo de software UML. (11)

1.3.4 Herramienta para Gestor de Base de Datos

Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (*DataBase Management System*) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Algunos ejemplos de SGBD son DB2, PostgreSQL, MySQL, MS SQL Server y Oracle seleccionado por el cliente como el gestor de base de datos a utilizar en el software. (12)

Oracle Database 11g

Oracle Database 11g proporciona nuevas e innovadoras funcionalidades que garantizan alto rendimiento, alta escalabilidad, fiabilidad y seguridad mediante el uso de plataformas grid, asegurando altos niveles de calidad de servicio e incrementos de la flexibilidad de negocio reduciendo además los costes de explotación. Con Oracle Database 11g los clientes pueden resolver las problemáticas de negocio más exigentes en todas las áreas, incluyendo aplicaciones transaccionales, de inteligencia de negocio y de gestión de contenidos. (13)

[3] *Unified Modeling Language*

1.3.5 Herramientas para el entorno de desarrollo integrado.

Un entorno de desarrollo integrado (IDE[4]) es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI[5]). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (14)

Los IDEs están diseñados para maximizar la productividad del desarrollador, su objetivo es reducir la configuración necesaria para recomponer las utilidades de línea de comandos en una unidad cohesiva. Esto significa para el programador un modo de hacer mucho menos cambios en el código que cuando se utilizan los programas de desarrollo discreto.

Visual Studio Team System 2008 es una solución integrada de *Application Lifecycle Management* (ALM, Gestión del Ciclo de Vida de las Aplicaciones), que comprende herramientas, procesos, y guías para ayudar a los miembros de un equipo a mejorar sus conocimientos y trabajar juntos de forma más efectiva. Esta potente herramienta de Microsoft fue seleccionado por el equipo de trabajo como IDE de desarrollo a utilizar en el proyecto.

Visual Studio Team System 2008

Visual Studio Team System facilita la comunicación y la colaboración del equipo de desarrollo, proporcionando un repositorio unificado de todos los datos del proyecto, junto con las herramientas para definir, ejecutar y automatizar los procesos deseados. En su conjunto de herramientas integra Team Foundation Server 2008, que permite dar soporta el código fuente y al control de versiones, así como dar seguimiento a los elementos de trabajo, construir y automatizar los controles de calidad, y más.

Ofrece una amplia gama de herramientas para todas las fases del desarrollo de software, incluidos la creación, la prueba, la implementación, la integración y la administración. Además, permite a los desarrolladores comunicarse mediante PC[6]s, servidores, la Web y dispositivos móviles. Visual Studio Team System 2008 ofrece herramientas de desarrollo de aplicaciones, como:

[4] ***I**ntegrated **D**evelopment **E**nvironment*
[5] ***G**raphical **U**ser **I**nterface*
[6] ***P**ersonal **C**omputer*

- ✓ Sistema de proyectos para administrar los datos requeridos para el desarrollo de aplicaciones de su proyecto.
- ✓ Herramientas de edición de código para escribir y modificar texto y código.
- ✓ Herramientas de refactorización y depuración para mejorar el código e identificar y resolver errores lógicos.
- ✓ Herramientas de plataformas para la escritura de aplicaciones para tecnologías de Office, Windows CE, .NET y Windows.
- ✓ Herramientas avanzadas para diseñar, implementar, evaluar, analizar, probar y evaluar el progreso del desarrollo de aplicaciones.
- ✓ Opciones de lenguajes que incluyen JScript 8.0, Visual Basic 2008, Visual C# 2008 y Visual C++ 2008. (15)

Framework .Net 3.5

.NET Framework es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- ✓ Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- ✓ Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- ✓ Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- ✓ Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- ✓ Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en la Web.
- ✓ Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

ASP .NET

ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del Common Language Runtime, seguridad de tipos y herencia

ASP.NET incluye:

- ✓ Marco de trabajo de página y controles
- ✓ Compilador de ASP.NET
- ✓ Infraestructura de seguridad
- ✓ Funciones de administración de estado
- ✓ Configuración de la aplicación
- ✓ Supervisión de estado y características de rendimiento
- ✓ Capacidad de depuración
- ✓ Marco de trabajo de servicios Web XML
- ✓ Entorno de host extensible y administración del ciclo de vida de las aplicaciones
- ✓ Entorno de diseñador extensible (16)

Windows Workflow Foundation (WWF)

Windows Workflow Foundation es un marco que permite a los usuarios crear flujos de trabajo humanos o de sistema en sus aplicaciones escritas para los sistemas operativos Windows Vista, Windows XP y Windows Server 2003. Está compuesto por un espacio de nombres, un motor de flujo de trabajo en proceso y diseñadores para Visual Studio 2005. Windows Workflow Foundation se puede utilizar para resolver escenarios simples, como mostrar controles de interfaz de usuario basados en datos proporcionados por el usuario, o escenarios complejos encontrados por empresas grandes, como procesamiento de pedidos y control de inventario. Windows Workflow Foundation se incluye con un modelo de programación, un motor de flujo de trabajo personalizable y rehospedable, y herramientas para generar rápidamente las aplicaciones habilitadas para el flujo de trabajo en Windows.

Windows Workflow Foundation proporciona una experiencia de desarrollo coherente y familiar con otras tecnologías .NET Framework 3.0, como Windows Communication Foundation y Windows Presentation Foundation. La API[7] de la Windows Workflow Foundation proporciona la compatibilidad completa para Visual Basic .NET y C#, un compilador del flujo de trabajo especializado, depura dentro de un flujo de trabajo, un Workflow Designer gráfico, y desarrolla completamente su flujo de trabajo en código o en marcado. Windows Workflow Foundation también proporciona un modelo extensible y un diseñador para generar actividades personalizadas que encapsulan la funcionalidad del flujo de trabajo para los usuarios finales o para reutilizarse en varios proyectos. (16)

C Sharp(C#)

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores generar diversas aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Este lenguaje se puede utilizar para crear aplicaciones cliente para Windows tradicionales, servicios web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y muchas tareas más. Visual C# 2008 proporciona un editor de código avanzado, diseñadores de interfaz de usuario prácticos, un depurador integrado y muchas otras herramientas para facilitar el desarrollo de aplicaciones basado en la versión 3.0 del lenguaje C# y en la versión 3.5 de .NET Framework.

Además de los principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen las siguientes:

- ✓ Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- ✓ Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- ✓ Comentarios en línea de documentación XML.
- ✓ Language-Integrated Query (LINQ) que proporciona funciones de consulta integradas en una gran variedad de orígenes de datos. (16)

[7] ***A**pplication **P**rogramming **I**nterface*

1.3.6 Capa de Acceso a datos

Los arquitectos y programadores de aplicaciones orientadas a datos frecuentemente se enfrentan a la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas empresariales que resuelven, y también deben trabajar con los motores de datos que se usan para almacenar y recuperar los datos. Para darle solución a este problema la dirección del proyecto decidió la utilización de Entity Framework y LinQ, herramientas que permitirán el desarrollo de aplicaciones de acceso a datos programadas en un modelo de la aplicación conceptual en lugar de hacerlo directamente con un esquema de almacenamiento relacional. El objetivo fundamental que se persigue es el de reducir la cantidad de código y mantenimiento que se necesita para las aplicaciones orientada a datos. Esto ofrece grandes ventajas que favorecen al desarrollo, entre las que se pueden mencionar, el hecho de que las aplicaciones estén libres de dependencias de codificación rígida de un motor de datos, o de un esquema de almacenamiento y que las asignaciones entre el modelo conceptual y el esquema específico de almacenamiento puedan cambiar sin tener que cambiar el código de la aplicación.

Entity Framework

Entity Framework es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Esta tecnología facilita a los programadores trabajar con datos en forma de objetos y propiedades específicos del dominio, por ejemplo, con clientes y direcciones, sin tener que pensar en las tablas de las bases de datos subyacentes y en las columnas en las que se almacenan estos datos. Para ello, se eleva el nivel de abstracción en la que los programadores pueden trabajar al tratar con datos y se reduce el código requerido para crear y mantener las aplicaciones orientadas a datos. Dado que Entity Framework es un componente de .NET Framework, las aplicaciones de Entity Framework se pueden ejecutar en cualquier equipo en que esté instalado .NET Framework 3.5 Service Pack 1 (SP1). (16)

Language-Integrated Query (LINQ)

LINQ es una importante innovación en Visual Studio 2008 y .NET Framework versión 3.5 que elimina la distancia que separa el mundo de los objetos y el mundo de los datos. LINQ permite a los programadores formar consultas basadas en conjuntos en el código de su aplicación, sin tener que usar

un lenguaje de consulta independiente. Se puede escribir consultas de LINQ en varios orígenes de datos enumerables (es decir, un origen de datos que implementa la interfaz *IEnumerable*), como estructuras de datos en memoria, documentos XML, bases de datos SQL[8] y objetos DataSet. Aunque esos orígenes de datos enumerables se implementan de varias formas, todos revelan las mismas construcciones de lenguaje y sintaxis. Como las consultas se pueden formar en el lenguaje de programación mismo, no es necesario utilizar otro lenguaje de consultas que esté incrustado como literales de cadena que el compilador no pueda entender o verificar. La integración de consultas en el lenguaje de programación también permite a los programadores de Visual Studio ser más productivos proporcionando comprobación de sintaxis y tipo en tiempo de compilación e *IntelliSense*. Estas características reducen la necesidad de depuración y corrección de errores de consultas. (16)

1.3.7 Herramientas para generación de reportes

Los generadores de reportes son herramientas que procesan la información almacenada por el sistema y la entregan en un formato particular conocido como reporte, que tiene como objetivo mostrar y procesar los datos capturados por las aplicaciones, de forma que se puedan convertir en información útil para las personas que los necesitan.

Para lograr un mayor conocimiento en lo que respecta a generación de reportes se hace un análisis de algunas herramientas existentes para esta tarea como son: Crystal Reports Basic for Visual Studio 2008, Active Report 6, Stimulsoft Report .Net y API del Sistema de Gestión de Reportes Dinámicos.

Crystal Reports Basic for Visual Studio 2008

Crystal Reports para Visual Studio proporciona una forma muy productiva de crear e integrar informes interactivos con calidad de presentación en las aplicaciones para Windows, Web o servicios Web XML. El diseñador gráfico incrustado permite crear fácilmente informes desde el IDE y minimiza la codificación intensiva asociada al desarrollo de informes.

Crystal Reports ha formado parte de Visual Studio desde 1993, y ahora es el estándar de elaboración de informes de Visual Studio. Se incluye en todas las copias de Visual Studio Professional y se integra directamente en el entorno de desarrollo.

[8] ***S****tructured* ***Q****uery* ***L****anguage*

Crystal Reports para Visual Studio incorpora la posibilidad de crear contenido interactivo con calidad de presentación al entorno de Windows. Con Crystal Reports para Visual Studio, puede crear informes complejos y profesionales en un programa basado en GUI. Después puede conectar el informe a casi todos los orígenes de base de datos, así como a datos proxy, como un conjunto de resultados (por ejemplo, un ADO.NET DataSet). Los asistentes del diseñador de GUI le permiten establecer fácilmente los criterios de formato y agrupamiento y gráficos.

Puede almacenar el informe en una aplicación Web o para Windows, con uno de los controles de visores de Crystal Reports para Visual Studio. La presentación de informes, tanto en clientes Windows como en HTML 3.2 ó 4.0, es muy interactiva y proporciona funciones como la profundización en gráficos, la exploración de informes y la búsqueda de texto.

Crystal Reports para Visual Studio incluye un SDK[9] extenso. Puede utilizarlo para interactuar con el informe mediante programación en tiempo de ejecución, usando uno de los cuatro modelos de objetos posibles:

- **CrystalReportViewer**, el modelo de objetos más sencillo.
- **ReportDocument**, el modelo de objetos más completo.
- **ReportClientDocument**, el modelo de objetos más completo. Este modelo de objetos está disponibles con Crystal Reports 2008 o con un servidor RAS.
- **InfoObject**, un modelo de objetos muy eficaz para la programación y configuración de informes en el marco de Crystal Reports Server o BusinessObjects Enterprise.

Active Reports 6

Active Reports es una herramienta de presentación de informes flexibles para el mundo de .NET. Ofrece un extenso código subyacente, basado en eventos de la API para la creación de informes, además de la capacidad para modelar casi cualquier comportamiento de la presentación de informes en tiempo de ejecución. Los desarrolladores han utilizado durante mucho tiempo ActiveReports para integrar características de presentación de informes avanzados a sus aplicaciones. Esta herramienta le permite a los informes conectarse a una gran variedad de fuentes de datos, incluyendo tiempo de ejecución de

[9] ***S**oftware **D**evelopment **K**it*

datos independiente, así como cambiar las propiedades de fuente de datos para proporcionar informes *ad-hoc*. Es compatible con OLEDB, XML, .NET clientes de SQL, las colecciones de conjuntos de datos, vistas de datos, tablas de datos y cualquier entidad que apoya la interfaz *IList*.

ActiveReports permite exportar sus informes en una variedad de formatos, incluyendo PDF, HTML, MHT, RTF, Excel (. Xls), texto sin formato (. TXT), valores separados por comas (*. CSV), y. TIFF. (18)

Stimultsoft Reports.Net

Stimultsoft Reports.Net es un generador de informes basado en .NET que ayuda a crear informes flexibles y con muchas características especiales. Stimulsoft Reports.Net se entrega con todo el código fuente. Todos los informes se crean mediante un diseñador de informes. Permite utilizar el diseñador de informes tanto en tiempo de diseño como en tiempo de ejecución. No es necesario pagar derechos por el uso del diseñador en tiempo de ejecución. Utilizando Stimulsoft Reports.Net se pueden crear informes basados en una gran variedad de orígenes de datos. Los informes creados se pueden utilizar en aplicaciones WinForms y Asp.Net. Los informes visualizados se pueden exportar a: Pdf, Xps, Xml, Html, Word, Excel, Rtf, Txt, Csv, Emf, Bmp, Jpeg, Gif, Png, and Tiff. (19)

API del Sistema de Gestión de Reportes Dinámicos

El Generador de Reportes Dinámico es una aplicación Web que tiene como objetivo generar reportes de forma rápida, interactiva y con una amplia gama de alternativas para los usuarios. La extensión en su uso puede estandarizar la generación de reportes en diferentes aplicaciones independientemente del Sistema Gestor de Base de Datos que utilicen ya sea MySQL o PostgreSQL.

El Generador permite a los usuarios, entre otras opciones, abstraerse a los conocimientos relacionados con los Gestores de Bases de Datos, agilizar la toma de decisiones y generar reportes en varios formatos y con gran variedad de opciones en su diseño, marcando una diferencia entre los reportes tradicionales y los reportes dinámicos, objetos de este producto.

El sistema está compuesto por varias aplicaciones entre las que se encuentran el Visor de reportes, Diseñador de modelos, Diseñador de reportes, Diseñador de consulta y Administrador de reportes. (20)

Selección de la Herramienta para generación de reportes

Después de realizada una investigación de las principales herramientas de generación de reportes y teniendo en cuenta las necesidades del cliente en cuanto a las características que debe presentar el sistema, se decide hacer uso de la herramienta Crystal Reports por las siguientes razones:

- ✓ Crystal Reports es el software líder en la elaboración de informes.
- ✓ Proporciona una amplia visión de su información y acelera el desarrollo de informes.
- ✓ Presenta información a los usuarios en formatos de manejo sencillo.
- ✓ Es el estándar mundial en el campo de la generación de informes, desde el acceso a los datos y el diseño de los informes hasta la gestión, distribución e integración con portales y aplicaciones.
- ✓ Es el estándar de elaboración de informes de Visual Studio. Se incluye en todas las copias de Visual Studio y se integra directamente en el entorno de desarrollo.

1.4 Conclusiones

En el transcurso del capítulo se realizó un análisis de distintos sistemas existentes para el manejo de reportes, se establecieron las herramientas y tecnologías que se utilizarán en la modelación e implementación del Módulo de Reportes para el Sistema Único de Identificación Nacional y se hizo un estudio de las principales herramientas para la generación de reportes, donde se seleccionó Crystal Report como el gestor de reportes a utilizar.

Capítulo 2: Características del Sistema

Introducción

En el presente capítulo se realiza un análisis de las características del sistema a desarrollar a partir de la problemática planteada, se describen las principales funcionalidades que brindará el módulo de reportes a desarrollar e identifican a los trabajadores que intervienen.

2.1 Modelo de Dominio

Teniendo en cuenta que el proceso de negocio de los reportes no está bien definido ya que no se ven claramente las fronteras del mismo, así como personas que lo inician o, incluso, quién o quiénes van a ser los beneficiados, se arribó a la conclusión de que el negocio que se está estudiando no necesita de un modelado completo, por lo que se propone un modelo de dominio.

Con la realización del modelo de dominio se pretende ayudar a comprender los conceptos con que trabajan los usuarios y con los que deberá trabajar la aplicación. Para su elaboración fue necesaria la participación de expertos del negocio, junto a los que se llevaron a cabo tres tareas fundamentales:

1) Identificar las Clases Conceptuales.
2) Graficarlas en un Diagrama de Clases.
3) Añadir relaciones y atributos.

El Modelo de Dominio queda recogido en un diagrama UML, parecido a un diagrama de clases pero más simplificado, que muestran a los interesados las clases del dominio y como se relacionan unas con otras mediantes asociaciones.

Como primer paso para la modelación del dominio se definieron todos los conceptos que se van a utilizar en el diagrama y se recogieron en el glosario de términos de nombres que se muestra a continuación:

Oficina DIR: Oficina de Dirección de Identificación y Registro que atiende a nivel nacional todos los procesos de identidad de las personas.

Oficina Provincial: Oficina provincial que se encarga de dirigir y controlar los procesos llevados a cabo en cada oficina del carné de identidad de los municipios.

Oficina CIRP: Oficina del Carné de Identidad y Registro de la Población donde se realizan los trámites correspondientes con la identidad de cada persona.

Área CIRP: Agrupación de áreas donde se realizan los trámites del carné de identidad.

Recepción, **Archivos**, **Trámite**, **Confección** y **Entrega**: Áreas que conforman las oficinas del CIRP.

Sistema SAAP: Es el Sistema Automatizado de Atención a la Población, permite viabilizar y controlar los procesos internos, como son: la recepción, archivo e inicio de trámites.

Sistema SCORE: Es un servicio que integra la información de los ciudadanos almacenada en los sistemas existentes en las diferentes áreas.

Sistema SACIRP: Es el Sistema Automatizado para el Carné de Identidad y Registro de la Población, permite la introducción y verificación de datos durante el proceso de trámite. Las operaciones son realizadas sobre la Base Provincial, permitiendo hacer búsquedas de personas en la Base Nacional e importar los datos de las mismas a la correspondiente Base Provincial.

Funcionario Recepción: Persona encargada de atender a la persona que solicita un trámite e inserta los datos de esta persona en el SAAP.

Jefe Unidad: Asume el rol de Administrador de los sistemas, es el encargado de gestionar los permisos sobre estos, y la información que hay en ellos.

Funcionario Archivo: Persona encargada de buscar los antecedentes penales de la persona en el SCORE y los datos personales en los archivos.

Funcionario Trámite: Persona encargada de realizar el trámite solicitado.

Usuario Reportes: Rol que asume la persona encargada de gestionar los reportes en los sistemas SAAP y SACIRP.

Módulo Reportes SAAP: Módulo donde se gestionan los reportes del SAAP.

Módulo Reportes SACIRP: Módulo donde se gestionan los reportes del SACIRP.

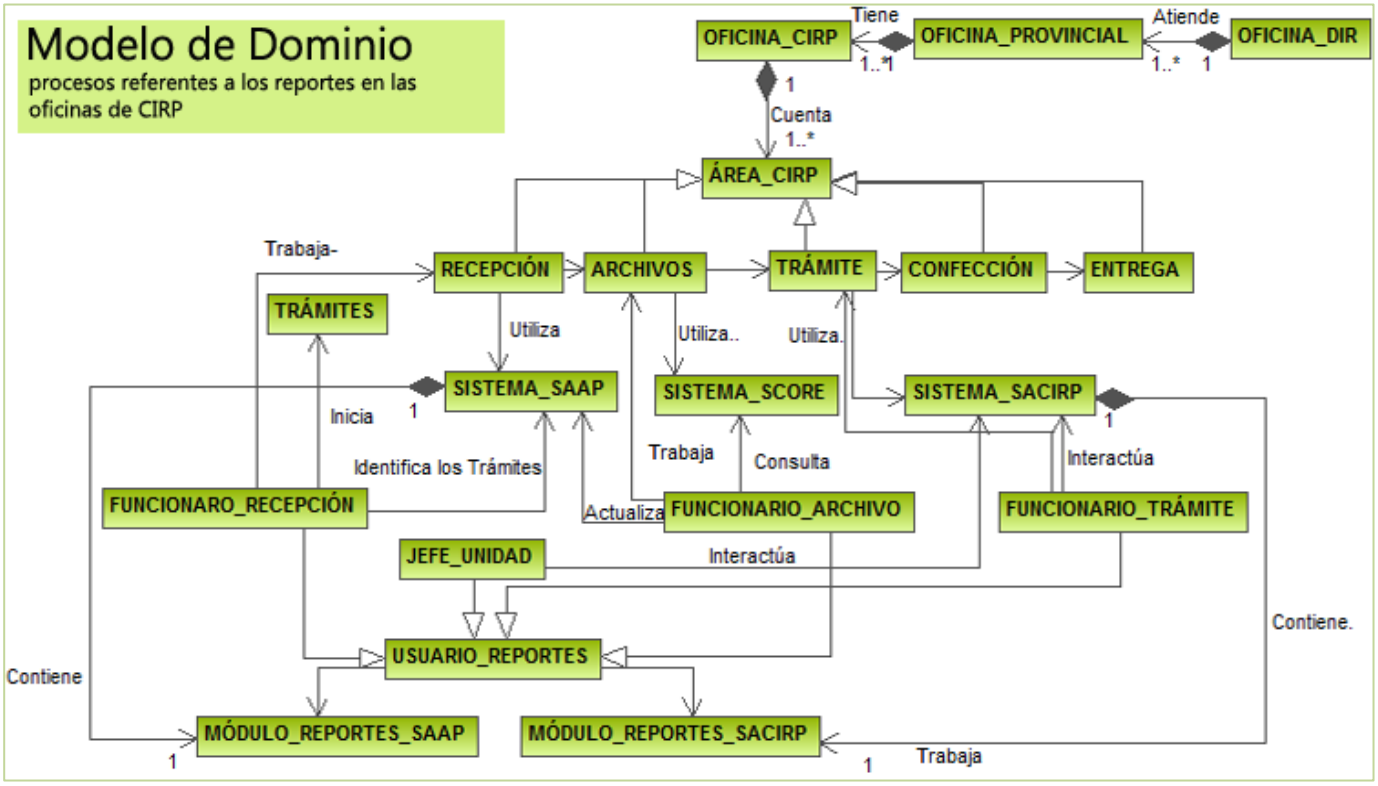


Fig. 4 Modelo de Dominio

Descripción del modelo de dominio:

Las oficinas del Carné de Identidad y Registro de la Población (CIRP) están estructuradas por áreas de procesos para llevar a cabo la tramitación de los documentos de identidad nacional, estas son: Área de Recepción, Archivos, Trámite, Confección y Entrega. En estas áreas los procesos son llevados a cabo por funcionarios los cuales juegan roles diferentes. Cuando una persona se dirige a una oficina del carné de identidad y solicita un trámite es atendido en la Recepción por el funcionario de dicha área. En esta área se recogen los datos de la persona en el Sistema de Solicitud de Certificación de Identidad (SAAP), para ello el funcionario del Área de Archivos hace una búsqueda en el sistema que contiene los antecedentes penales de todas las personas (SCORE), además de buscar los datos de las personas en los archivos de la oficina. Después el funcionario del Área de Trámite con el uso del Sistema de

Automatización de los procesos del CIRP (SACIRP) gestiona los datos de la dirección de la persona para culminar el trámite que se esté llevando a cabo. Finalmente se confecciona el documento de identidad en el área correspondiente y se le entrega a la persona. Una vez concluido el proceso los sistemas SACIRP y SAAP permiten la generación de diferentes reportes estadísticos que pueden ser generados por cualquiera de sus usuarios. Esto se hace a través de los Módulos de Reportes de cada uno de estos sistemas.

2.2 Especificación de Requisitos Funcionales

Un requerimiento funcional, según la IEEE Standard Glossary of Software Engineering Terminology, se define como una "condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo" (21). Los requerimientos son condiciones o capacidades que tienen que ser alcanzadas o poseídas por un sistema para satisfacer las necesidades de los clientes. Para el desarrollo de la aplicación se deben describir estas funcionalidades con el objetivo de lograr un mayor entendimiento entre clientes y desarrolladores como premisa para lograr un producto con la calidad requerida. En la realización del Módulo de Reportes se llevó a cabo una mejora de los procesos que se realizan en la institución de forma tal que tributen a las principales funcionalidades que debe cumplir el software y de lograr una armonía entre clientes y los desarrolladores con respecto a las capacidades de la futura aplicación.

2.2.1 Concepción de los modelos de procesos

La norma internacional ISO-9001 define un proceso como "una actividad que utiliza recursos, y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados" (22). El Módulo de Reportes supone una herramienta de apoyo para la mejora del proceso de toma de decisiones de la dirección de la organización. Este módulo permitirá el manejo de información de forma dinámica y facilitará el control sobre actividades que conforman el sistema y las personas que interactúan con él. Por su importancia se decidió independizarlo de forma que pueda ser de utilidad para los restantes módulos que conforman el SUIN como se muestra en la Fig. 5 Relación entre el Módulo de Reportes y los restantes módulos del SUIN. El módulo de reportes queda separado del flujo principal de los módulos del SUIN pero se relaciona con todos los módulos como una herramienta que permitirá una mejor visualización de la información solicitada.

Fig. 5 Relación entre el Módulo de Reportes y los restantes módulos del SUIN

A través del modelado de las actividades y procesos puede lograrse un mejor entendimiento del negocio y muchas veces esto presenta la oportunidad de mejorarlos. A continuación se muestran los modelos de procesos correspondientes a cada funcionalidad de la solución:

El flujo principal del Módulo de Reportes permitirá la gestión de los reportes, proceso este que incluye los subprocesos adicionar, mostrar y eliminar un reporte tal y como se muestra en la Fig. 6 Procesos del Módulo de Reportes.

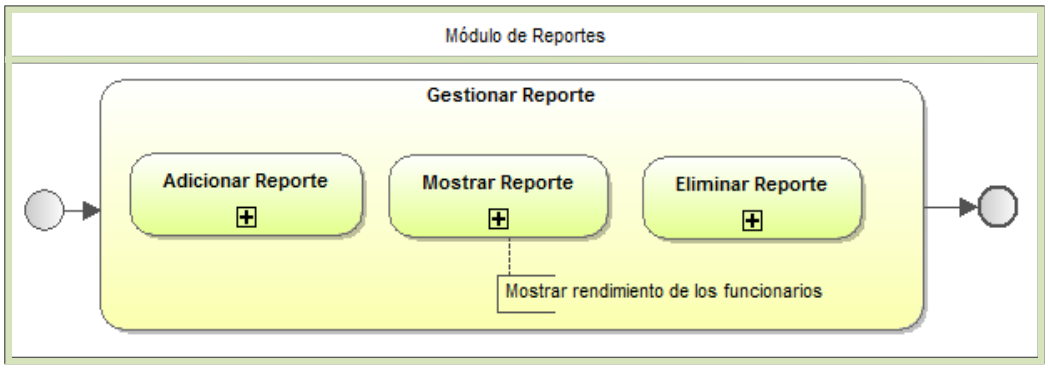


Fig. 6 Procesos del Módulo de Reportes

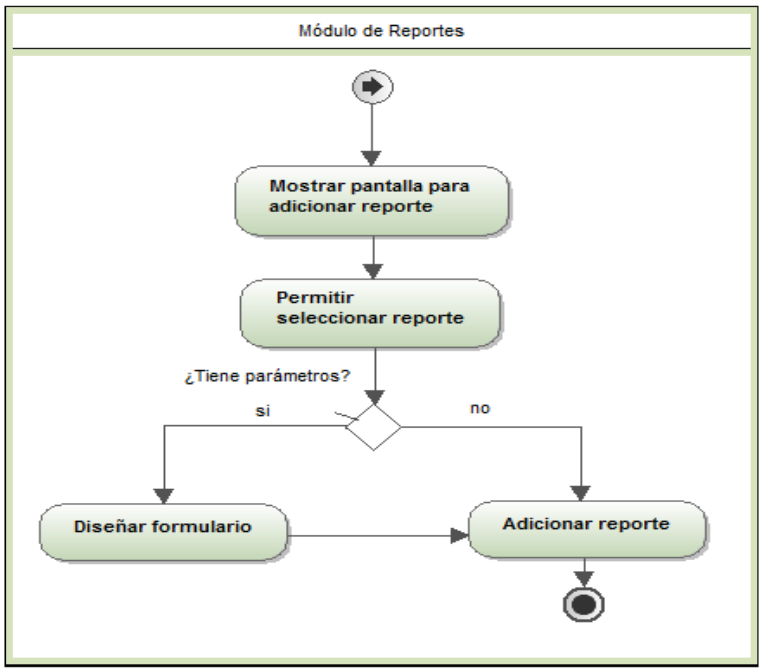


Fig. 7 Subproceso Adicionar Reporte

Subproceso Adicionar Reporte

En la Fig. 7 Subproceso Adicionar Reporte se puede apreciar el flujo de actividades que se llevan a cabo en el subproceso adicionar reporte. Este subproceso da inicio cuando el administrador de reportes necesita incorporar uno nuevo al sistema. La aplicación mostrará una interfaz donde el administrador podrá seleccionar la ruta del documento de Crystal Report (.rpt) que representará al nuevo reporte. En caso de que el documento seleccionado necesite parámetros, se procederá a diseñar el formulario de

captura de los datos asociado al mismo. Posteriormente se adicionará el reporte al sistema y concluirá el flujo de eventos de este subproceso.

Subproceso Mostrar Reporte

Mostrar un reporte es un subproceso que puede ser iniciado tanto por el administrador como por el usuario de reportes. El flujo comienza con la necesidad de visualizar un reporte determinado. Para ello un usuario autenticado en el sistema seleccionará el reporte que desea visualizar, especificando los parámetros necesitados por el reporte en caso de que este los requiera y el sistema procederá a su visualización. La Fig. 8 Subproceso Mostrar Reporte muestra el flujo de eventos que se llevan a cabo en este subproceso.

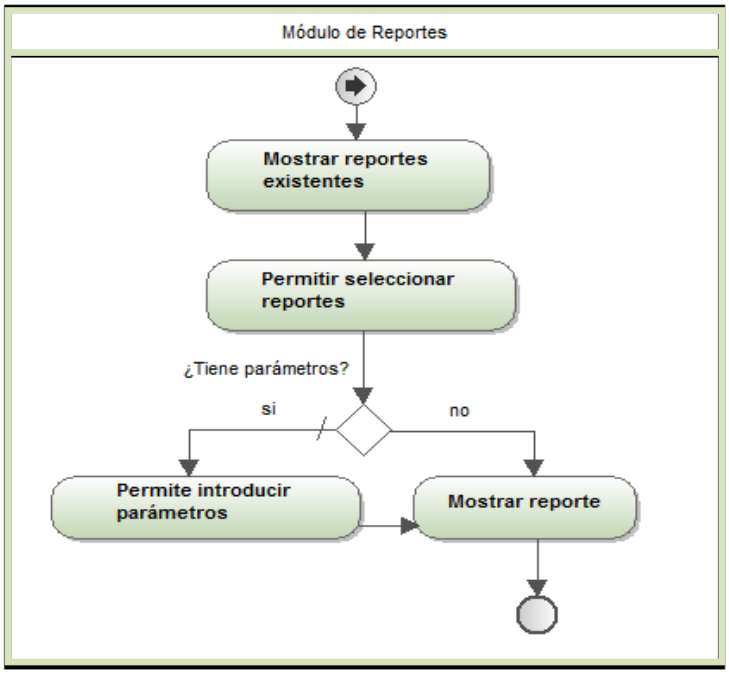


Fig. 8 Subproceso Mostrar Reporte

Subproceso Eliminar Reporte

El administrador de reportes es el único autorizado para eliminar un reporte del sistema. Para esto seleccionará el reporte que desee y lo eliminará de la aplicación. Ver Fig. 9 Subproceso Eliminar Reporte.

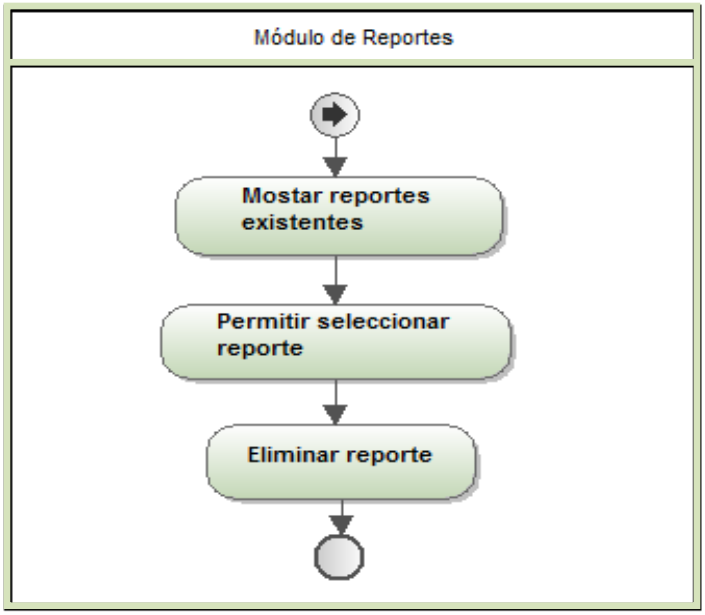


Fig. 9 Subproceso Eliminar Reporte

Para mayor información de los procesos del módulo de reportes consultar el Anexo 6: Concepción de los modelos de procesos.

2.2.2 Propuesta de solución

La solución que se propone permitirá la incorporación de nuevos reportes al sistema sin necesidad de llevar a cabo una recompilación de la aplicación. A través del Módulo de Reportes, el usuario será capaz de diseñar el formulario de captura de parámetros que desea utilizar para introducir valores al documento de reporte, haciendo este proceso más cercano a su destinatario y a satisfacer sus necesidades.

La aplicación funcionará como un repositorio de reportes, permitiendo almacenar el documento del reporte asociado a un conjunto de controles especificados por el usuario, que facilitarán la incorporación de los valores requeridos por el documento.

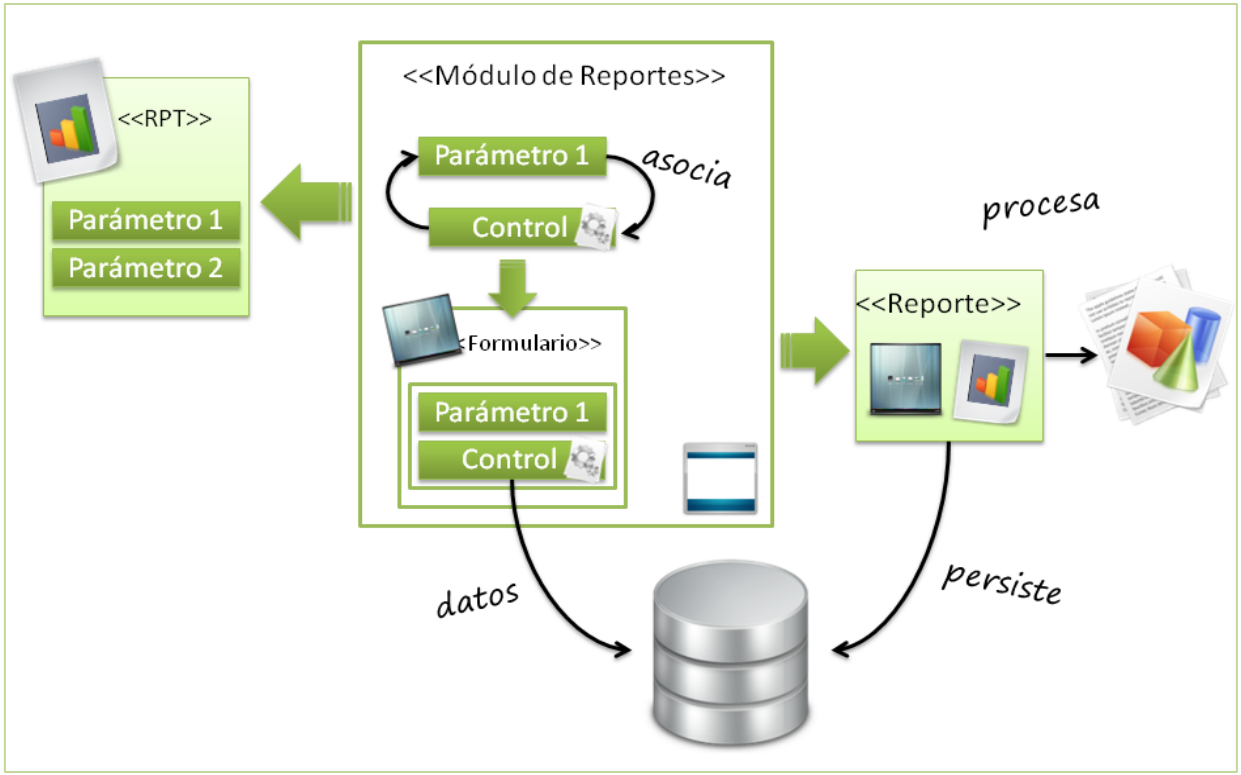


Fig. 10 Funcionamiento del Módulo de Reportes

Como se muestra en la Fig. 10 Funcionamiento del Módulo de Reportes, la aplicación recibirá un documento de reporte creado por el usuario, del cual extraerá los parámetros requeridos y las conexiones de cada uno de estos. A través de la asociación de cada uno de los parámetros con los controles del módulo se crean nuevos campos a los que se les asigna la conexión del parámetro. Este proceso elabora un nuevo formulario de captura de parámetros, el cual una vez terminado es persistido junto a su documento de Crystal Report. Finalmente el usuario será capaz de introducir valores al reporte a través del formulario construido y posteriormente visualizar el mismo.

2.2.3 Definición de los actores del sistema a automatizar

Rol	Objetivo
Administrador de reportes	Garantizar la administración de los reportes de la aplicación. Específicamente puede realizar acciones tales como: adicionar, mostrar o eliminar los reportes

	del sistema.
Usuario de reportes	Tiene permisos para mostrar los reportes del área a la que pertenece.

Tabla 1 Roles del Sistema

2.2.4 Catálogo de requisitos funcionales

Módulo Reporte

RF1. Gestionar reporte.

1.1 Mostrar opción “Adicionar reporte”.

1.2 Mostrar opción “Mostrar reporte”.

1.3 Mostrar opción “Eliminar reporte”.

1.4 Permitir seleccionar la opción que se desea.

1.5 Si selecciona “Adicionar reporte”.

1.5.1 Ver **RF2**. Adicionar Reporte.

1.6 Si selecciona “Mostrar reporte”.

1.6.1 Ver **RF3**. Mostrar Reporte.

1.7 Si selecciona “Eliminar reporte”.

1.7.1 Ver **RF4.** Eliminar Reporte.

RF2. Adicionar Reporte.

2.1 Mostrar pantalla para adicionar reporte.

2.2 Permitir seleccionar reporte.

2.3 Si tiene parámetros.

2.3.1 Diseñar reporte.

2.3.2 Adicionar Reporte.

2.4 Si no tiene parámetros.

2.4.1 Adicionar Reporte.

RF3. Mostrar Reporte.

3.1 Mostrar reportes existentes.

3.2 Permitir seleccionar reporte.

3.3 Tiene parámetros.

3.3.1 Permitir introducir parámetros.

3.3.2 Mostrar reporte.

3.4 Si no tiene parámetros.

3.4.1 Mostrar reporte.

RF4. Eliminar Reporte.

4.1 Mostrar reportes existentes.

4.2 Permitir seleccionar reporte.

4.3 Eliminar Reporte.

Para mayor información de los requisitos del sistema consultar Anexo 4: Catálogo de Requisitos**¡Error! No se encuentra el origen de la referencia.**.

2.2.5 Descripción de requisitos funcionales

A continuación se muestra la descripción del requisito funcional Adicionar Reporte. Para más información de los requerimientos del sistema ver Anexo 1: Descripción de los Requisitos Funcionales.

RF2. Adicionar Reporte

Propósito	Permitir adicionar reportes	
Roles	Administrador de reportes	
Precondiciones	1. El administrador de reportes debe estar autenticado en el sistema	
Entidades Tratadas	**Entidad**	**Atributos**
	Reporte	• Nombre • Conexión • Usuario • Contraseña • Parámetros

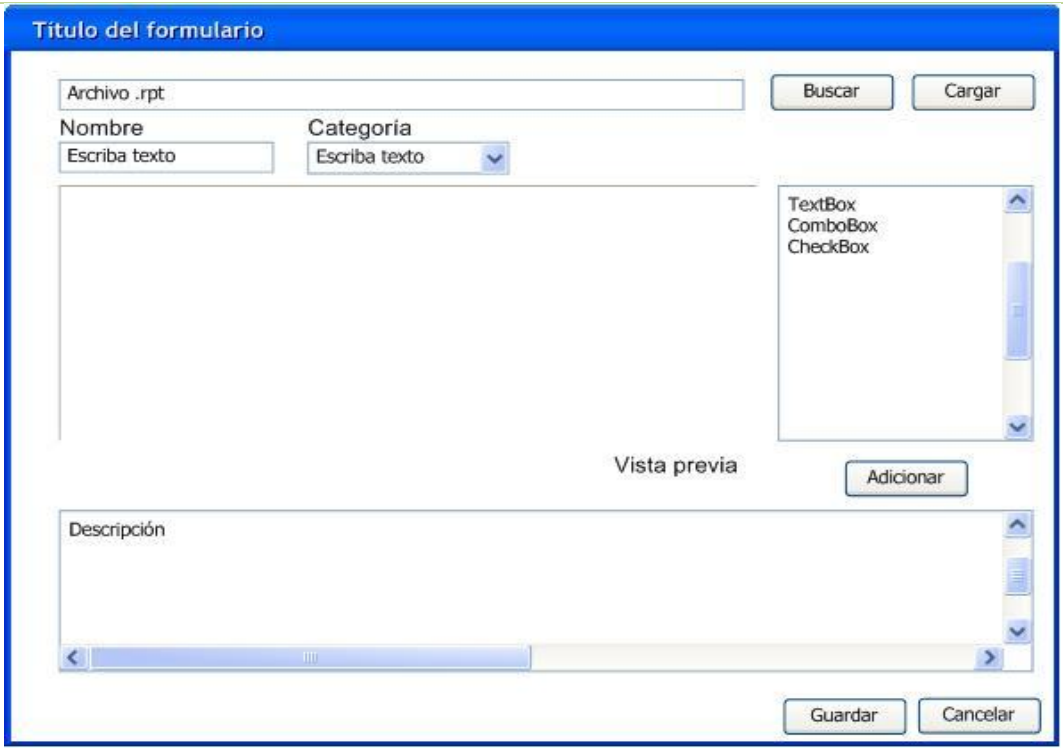
Descripción	2.2 Mostrar pantalla para adicionar reporte. 2.3 Permitir seleccionar reporte. 2.4 Si tiene parámetros. 2.4.1 Diseñar reporte. 2.4.2 Adicionar Reporte. 2.5 Si no tiene parámetros. 2.5.1 Adicionar Reporte.
Validaciones	Validar la corrección y completamiento de los datos definidos en el Modelo de Datos.
Postcondiciones	Se adicionaron reportes.
Prototipo	

Fig. 11 PIU Adicionar Reporte

2.3 Modelo de entidades conceptuales

El modelo de entidades conceptuales del Módulo de Reportes está representado por las clases Categoría y Reporte, tal y como se muestra en la Fig. 12 Modelo de entidades conceptuales, por ser estas las entidades que se relacionan directamente con el módulo. Indirectamente se relacionan con la

aplicación todas las entidades pertenecientes al sistema de identificación nacional, de donde se obtendrán los datos para mostrar los reportes.

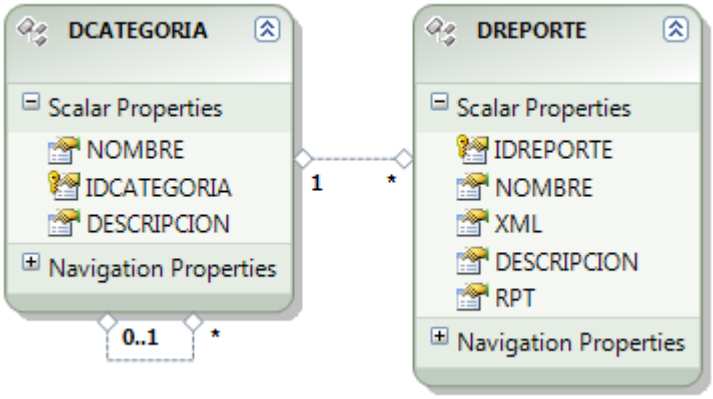


Fig. 12 Modelo de entidades conceptuales

2.3.1 Descripción de las entidades fundamentales

Tabla	DREPORTE
Descripción	Representa a los reportes almacenados en la aplicación
Atributos	IdReporte, Nombre, Descripción, RPT, XML

Tabla 2 Descripción de la entidad conceptual REPORTE

Tabla	DCATEGORIA
Descripción	Representa a las diferentes categorías a las que pertenecerán los reportes de la aplicación.
Atributos	IdCategoría, Nombre, Descripción

Tabla 3 Descripción de la entidad conceptual DCATEGORIA

2.4 Especificación de requisitos no funcionales

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, ya que aunque se conozca que el sistema cumple con la toda la funcionalidad requerida, propiedades no funcionales como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (21)

Usabilidad

- El sistema estará diseñado para una resolución de 1024x768 pixel.
- Su diseño será limpio y claro, facilitando la navegación.

Fiabilidad

- El sistema estará disponible las 24 horas durante los 7 días de la semana.
- No se realizarán mantenimientos preventivos en horario laboral, deberán ejecutarse en un horario estipulado o los fines de semana, para no afectar la disponibilidad del sistema.
- Las fallas del software se dividirán en dos categorías:
  - Simples: la solución y la actualización se realizarán en línea en un período inferior a 4 horas.
  - Complejas: la solución y actualización se realizarán en un tiempo que se definirá posterior a una evaluación detallada.
- El sistema llevará un sistema de tracking de errores.
- Solo se accederá a la BD desde la aplicación, nunca directamente desde el gestor de BD.
- Solo podrán acceder a las funcionalidades del sistema los usuarios que posean los permisos suficientes.
- Se garantizará la consistencia de los datos, se realizarán comprobaciones y validaciones automáticas en todos los casos posibles.
- La información manejada por el sistema será eliminada una vez terminada de procesar.

Eficiencia

- El sistema debe ser capaz de soportar una cantidad escalable de dispositivos de impresión.

Soporte

- El sistema debe codificarse siguiendo los estilos de código definidos por el proyecto.

Restricciones de diseño

- El sistema debe implementarse usando el lenguaje C#, sobre la plataforma ASP.NET.
- El sistema gestor de bases de datos, será Oracle 11g.
- El sistema debe desarrollarse usando el IDE Visual Studio Team System 2008.

- Se utilizará el Team Explorer como control de código fuente.

Consultar Anexo 5: Especificación de Requisitos no funcionales para mayor información de los requisitos no funcionales del sistema.

2.5 Conclusiones

Este capítulo muestra una clara definición de los requisitos que deberá cumplir el sistema, además de una representación de las funcionalidades a construir mediante el modelado de procesos, donde se representan los flujos por los que deben transitar las funcionalidades. Se elaboró un modelo de dominio con el objetivo de modelar el negocio tal y como se lleva a cabo en la actualidad y se realizó un modelo de entidades conceptuales describiéndose las entidades fundamentales y los atributos que las representan.

Capítulo 3: Análisis y Diseño

Introducción

En el presente capítulo se describen los principales elementos que componen la arquitectura y que guían el proceso de construcción del software. Se definen las clases que intervienen en la solución así como los servicios que dan soporte a las funcionalidades y los diseños de los *workflows*. También se obtienen los artefactos generados a partir de la aplicación de la metodología de desarrollo utilizada y una descripción detallada de las entidades que tienen persistencia en la base de datos.

3.1 Arquitectura de la solución

3.1.1 Vista lógica

Como parte del Sistema Único de Identificación Nacional la arquitectura se encuentra representada por 5 capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas. Con esta distribución se podrán realizar grandes cambios en una capa sin siquiera tener que realizar cambios en las demás. La comunicación entre las diferentes capas se realizará solo a nivel de interfaces, lo que permitirá trabajar de manera transparente a las instancias reales.

Fig. 13 Vista lógica del Módulo de Reportes

3.1.2 Capas de los módulos de la aplicación

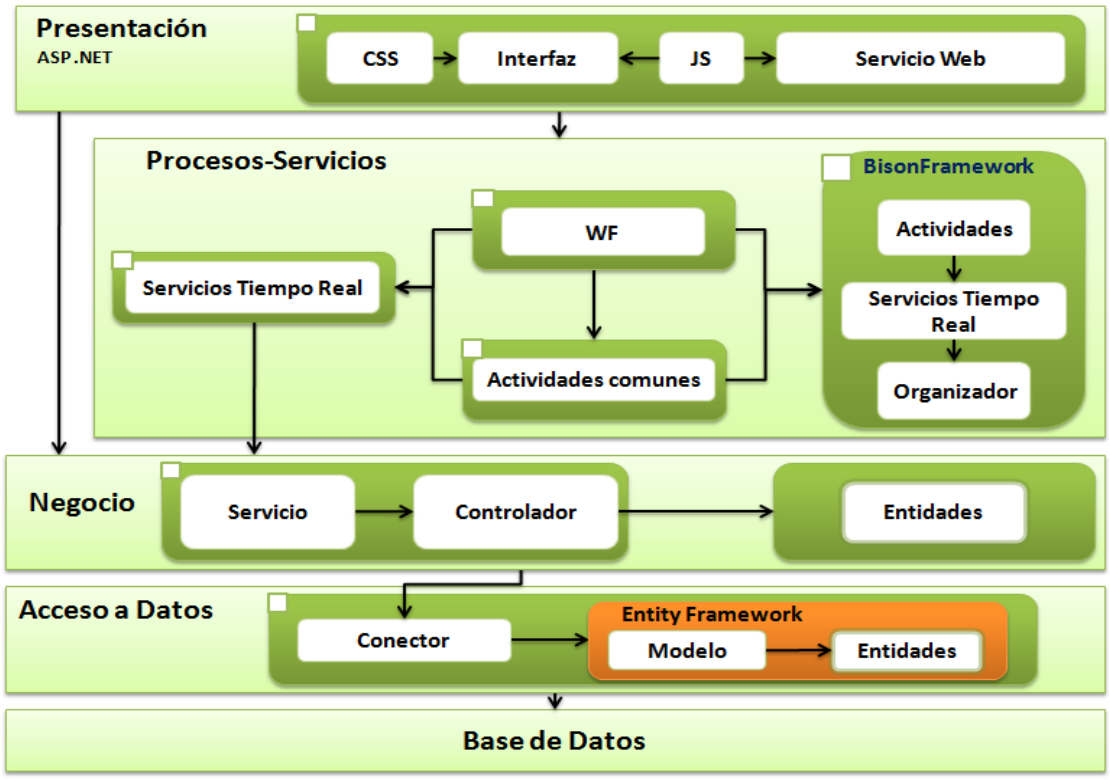


Fig. 14 Representación de las capas de la aplicación

Capa de Presentación

Está compuesta por todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Estos elementos pueden ser ficheros Java Script, CSS[10], servicios consumidos por Java Script, etc.

Esta capa se encuentra representada por el proyecto Web de la aplicación, y tiene interacción directa con las capas de Procesos-Servicios y con la de Negocio.

Capa de Procesos-Servicios

Esta capa está compuesta por *workflows*, que representan la lógica de los procesos de negocio, actividades, que por su nivel de reutilización o importancia lógica se encapsulan como propias del proyecto y los Servicios de Tiempo Real (*Runtime*), los cuales son los encargados de la interacción con los *workflows* definidos.

[10] *Cascading Style Sheets*

Los *workflows* definidos tienen la misión de interactuar con los servicios de la capa de negocio, haciendo uso para ello de una fábrica de servicios. A este mismo nivel se encuentra definida una fábrica para los Servicios de Tiempo Real, encargada de que la capa de presentación pueda interactuar con estos servicios sin conocer específicamente el servicio que utiliza. Todos estos elementos se encuentran vinculados directamente con el *framework* Bison.

Esta capa se encuentra relacionada con la capa de Presentación a la que le brinda servicios y con la capa de Negocio de la que consume servicios. Los proyectos que se encuentran relacionados son Report.Workflows donde se encuentran definidos los *workflows* y las actividades comunes, Report.Services donde se encuentran los servicios de *workflow* y la fábrica correspondiente a estos y el BisonFramework.

Bison Framework

Es un *framework* para la orquestación de procesos de negocio con Windows Workflow Foundation (WF). Su principal objetivo es proporcionar un componente que permita gestionar las instancias de *workflow*, además de encapsular un conjunto de actividades y servicios que dan mayor dinamismo al desarrollo de sistemas centrado en la orquestación de procesos de negocio con WF, específicamente para un ambiente web.

- **Actividades**

En este paquete se definen todas las actividades que por su comportamiento son necesarias en la arquitectura base.

- **Servicios de Tiempo Real**

En este paquete se encuentran todas las interfaces e implementaciones de servicios pertenecientes a la arquitectura base que propone el uso del *framework*. Estos servicios representan un factor clave dentro de la aplicación ya que son los que permiten la interacción con las instancias de proceso creadas.

- **Organizador**

En este paquete se encuentran las clases necesarias que permiten la gestión de las instancias de *workflow*, además de contener otras clases útiles para el acceso a los servicios del *framework*. Contiene además una clase especializada en la creación de las instancias de los servicios a utilizar dentro del Tiempo Real (*Runtime*).

Actividades

Este paquete contiene las actividades creadas específicamente para las distintas funcionalidades dentro del módulo.

- **Actividades comunes**

Conjunto de actividades que por su nivel de reutilización son definidas como actividades independientes para ser utilizadas por varios *workflows* o por uno más de una vez.

- **WF (*Workflows*)**

Son todos aquellos procesos definidos dentro de la aplicación donde se define la lógica de negocio de la aplicación, estos a su vez interactúan con los Servicios de Negocio que son los que contienen la lógica de las funcionalidades.

Servicios

Para lograr una correcta integración entre los procesos y el negocio es necesaria una correcta definición de los servicios con los que el *workflow* debe interactuar. Estos servicios se dividen en dos grupos:

- **Tiempo Real**

Son todos aquellos servicios arquitectónicamente importantes.

- **Negocio**

Son todos los servicios con los que interactúa un *workflow* para darle solución a una lógica de negocio determinada.

Capa de Negocio

En esta capa se recogen todos los servicios necesarios para darle solución a los requerimientos de negocio que no pueden ser satisfechos por el *workflow*.

Los servicios se encuentran definidos según el contexto en el que se desenvuelven. Tienen la responsabilidad de manejar todas las operaciones sobre una entidad de negocio en específico, así como todas las entidades que por conceptos de composición se encuentran relacionadas con esta. Por cada entidad de negocio que se crea un controlador y una interfaz que debe ser implementada por el acceso a dato que le dará soporte.

- **Entidades**

Tienen como principal objetivo describir el modelo de objetos relacionados con el negocio. Dentro de este proyecto se encuentran definidas tanto las entidades de negocio que pasan a la persistencia como aquellas que están constituidas con composiciones y agregaciones de las antes mencionadas y que solo

tienen vida en el transcurso de un proceso. Son los contenedores de información que fluyen a través de las capas de Presentación, Procesos-Servicios y Negocio.

- **Controlador**

Estas clases son las encargadas de contener las funcionalidades del negocio, para su interacción con la base de datos hacen uso de las clases de tipo Conector.

- **Conector**

Son las clases que permiten un alto nivel de abstracción en relación con la base de datos. Esta capa es la encargada de mantener la comunicación con la Capa de Acceso a datos para lograr la persistencia. A su vez expone las interfaces y servicios a la Capa de Procesos-Servicios, específicamente la sub-capa Servicios. Además brinda la posibilidad de que las clases de tipo Entidad sean utilizadas en otras capas.

Capa de Acceso a datos

La capa de acceso a datos está directamente relacionada con los servicios definidos en el negocio. Para establecer esta relación hace uso de las interfaces de los conectores y de la fábrica de conectores que define la capa de negocio. De esta manera es posible realizar cambios en esta capa sin que se vean afectadas las demás. Su principal función es realizar una implementación de las interfaces definidas en la capa de negocio y al mismo tiempo trabajar directamente con la fuente de datos establecida.

Capa de Base de Datos

Esta capa concentra una pequeña parte de la lógica de funcionalidades dentro de la aplicación. En ella se almacenan los datos de la aplicación. La única capa encargada de interactuar con esta es la Capa de Acceso a datos.

Interacción de las capas de la arquitectura en el Módulo de Reportes

En la figura Fig. 15 Interacción de las capas de la arquitectura se muestra la relación de las capas durante el subproceso adicionar reporte.

Adicionar reporte: Interfaz para adicionar un reporte en el Módulo de Reporte.

AddReportWorkflow: Es el workflow que guía el proceso de adicionar un nuevo reporte al sistema.

ReportRService: Servicio de tiempo de ejecución del Módulo de Reportes.

ReportServices: Servicio de negocio del Módulo de Reportes.

ReportController: Clase controladora del Módulo de Reportes.

IdentityReport: Clase entidad del Módulo de Reportes.

Category: Clase entidad del Módulo de Reportes.

Dreporte: Entidad de acceso a datos mapeada por el Entity Framework.

DCategoría: Entidad de acceso a datos mapeada por el Entity Framework.

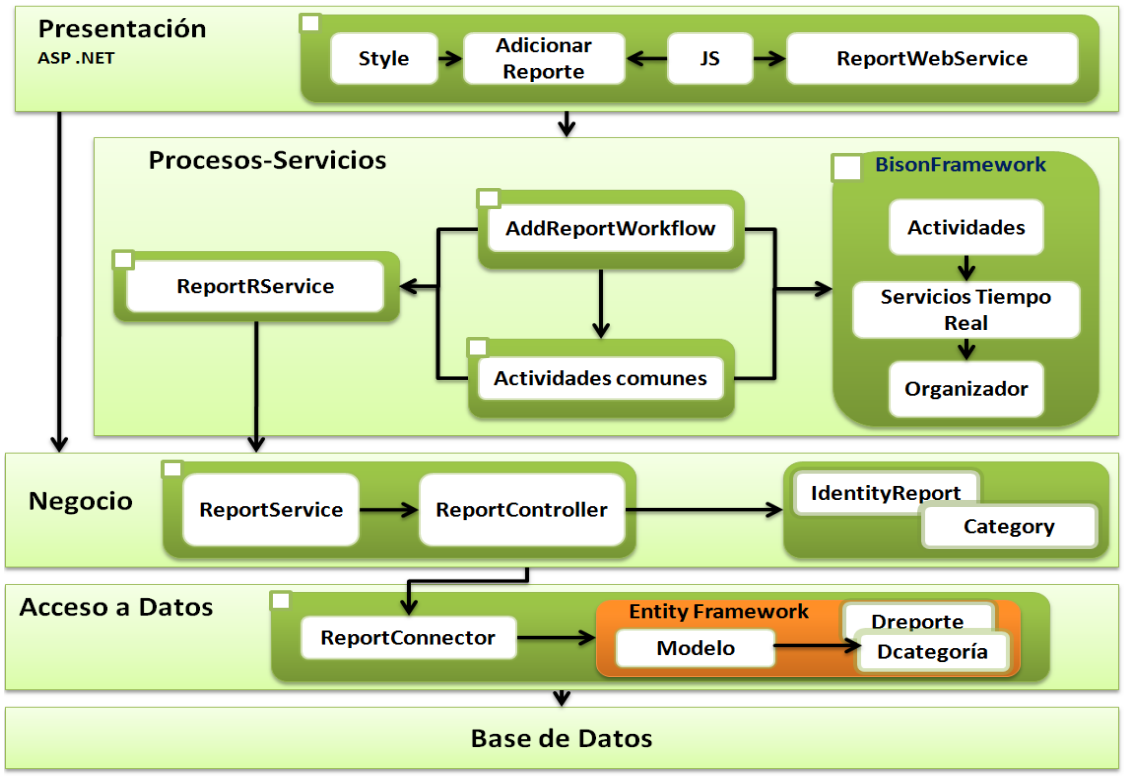


Fig. 15 Interacción de las capas de la arquitectura

3.2 Patrones de diseño utilizados

El desarrollo de un sistema conlleva, en la mayoría de las ocasiones, a darle solución a problemas muy complejos que ya alguien más ha resuelto. Por esta razón uno de los pasos a tener en cuenta cuando se decide desarrollar un proyecto de software es identificar que patrones pueden ser utilizados. Entiéndase por patrón como una solución estándar para un problema común de programación. A continuación se muestran varios de los patrones aplicados en el Módulo de Reportes.

Encapsulación: propone esconder algunos componentes, permitiendo solo accesos estilizados al objeto. Se hace uso de este patrón en casi todas las clases que componen al sistema permitiendo que estas solo posean como elementos públicos aquellos son exclusivamente necesarios.

Subclase: propone heredar miembros por defecto de una superclase, seleccionando la implementación correcta a través de resoluciones sobre qué implementación debe ser ejecutada. Se puede encontrar

este patrón con más fuerza en las entidades de negocio que por su conceptualización las funciones y la información que almacenan pueden estar diferenciadas en cierta medida. Ejemplo de utilización de este patrón en la aplicación:

```
[Serializable]
 public class ComboBoxField:ReportParameterField
       {

 public ComboBoxField(ReportParameter parameter):base(parameter)
 {

 }
```

Excepciones: propone introducir estructuras de lenguaje para arrojar e interceptar excepciones. Se identificaron los diferentes tipos de errores a tratar dentro del sistema creando clases que permitan identificar cada tipo de error en el momento de ejecución.

Fábrica: provee de una interfaz para crear familias de objetos relacionados o dependientes sin especificar los tipos concretos de clases. Su uso se encuentra centrado a la creación los conectores correspondientes al acceso a datos que se esté utilizando, así como en la obtención de los servicios a utilizar. Ejemplo de utilización de este patrón en la aplicación:

```
public static class ReportRuntimeFactory
{
  public static IReportRService ReportService
  {
   get
   {
     IReportRService service = BisonRuntime.GetService<IReportRService>();
     if (service == null)
     throw new RuntimeServiceNotFoundException("IReportRService");
     return service;
   }
  }
}
```

Singleton: se asegura que solo se pueda crear una instancia de la clase y ofrece un punto global de acceso a esta instancia. El uso de este patrón permite que los servicios puedan ser creados solo una vez.

GRASP: el uso de este patrón está totalmente ligado a cada componente desarrollado en el sistema, donde cada uno de ellos posee solo las funcionalidades acorde a las particularidades que lo caracterizan.

3.2.1 Patrones de workflow

Los patrones para el diseño de *workflow* van desde los más simples como el patrón secuencial hasta los más complejos, por ejemplo, el patrón de sincronización. Varios de los patrones utilizados para el diseño de los *workflows* se engloban en las siguientes clasificaciones: Patrones de control de flujo básicos y Patrones estructurales.

Patrones de control de flujo básicos: Estos patrones están presentes en la mayoría de los lenguajes de workflow, y sirven para modelar procesos secuenciales, paralelos, o aquellos que incluyan alguna decisión.

Uno de los patrones de control de flujo básico utilizado en la solución es el patrón **Secuencia** (*Sequence*) que plantea ejecutar una actividad determinada luego de completada la actividad anterior.

Patrones estructurales: Estos patrones permiten terminar un subproceso cuando ya no haya nada que hacer, o permiten definir ciclos de forma arbitraria. Son patrones que manejan múltiples instancias. Cuando se le da seguimiento a un caso, algunas veces es necesario que el proceso sea instanciado muchas veces.

Entre los patrones estructurales que se utilizan en la aplicación se puede mencionar el patrón **Ciclos Arbitrarios** (*Arbitrary Cycles*) que plantea establecer un punto en el proceso del workflow de manera tal que una serie de actividades puedan ser ejecutadas en varias ocasiones.

La utilización de los patrones antes mencionados de puede apreciar en la Fig. 17 Diseño del workflow de Adicionar Reporte imagen 1 donde las actividades se ejecutan de forma secuencial y existe un ciclo *while* que permite la ejecución de un conjunto de actividades reiterativamente.

3.3 Especificaciones de las clases

3.3.1 Clases Controladoras

Las clases controladoras tienen la responsabilidad de manejar las entidades del negocio que son necesarias persistir en una fuente de datos. Son las encargadas de convertir la información que devuelve el conector a la entidad de negocio correspondiente. Estas clases contienen asociada una

propiedad del tipo de la interfaz que representará al conector a la que se le pasará como parámetro solo la información necesaria para que realice las operaciones o funcionalidades solicitadas.

Nombre	ReportController
Descripción	Es la encargada de mediar entre los servicios de reporte y los conectores.
Métodos	Descripción
SaveReport	Persiste una entidad reporte en la base de datos.
LoadReport	Carga la entidad reporte especificada.
SaveCategory	Persiste una entidad categoría en la base de datos.
GetCategory	Devuelve la entidad categoría especificada.
GetSourceData	Devuelve los datos del campo especificado por parámetros.
GetCategories	Devuelve las subcategorías de la categoría especificada por parámetros.
EditCategory	Actualiza la entidad categoría especificada por parámetros.
DeleteCategory	Elimina la entidad categoría especificada.
GetReports	Devuelve los reportes pertenecientes a la categoría especificada.
DeleteReport	Elimina el reporte especificado.

Tabla 4 Descripción de la clase controladora ReportController.

3.3.2 Clases Conectoras

Implementan la interfaz del conector correspondiente. Implementan cada una de las funcionalidades definidas basándose en la fuente de datos y el mecanismo de comunicación utilizado. Hacen uso del LINQ para consultar o buscar información en la fuente de datos definida.

Nombre	ReportConnectorDAL
Descripción	Es la encargada de mediar entre las clases controladoras y la base de datos.
Métodos	Descripción
SaveReport	Persiste una entidad reporte en la base de datos.
LoadReport	Carga la entidad reporte especificada.
SaveCategory	Persiste una entidad categoría en la base de datos.
GetCategory	Devuelve la entidad categoría especificada.
GetSourceData	Devuelve los datos del campo especificado por parámetros.
GetCategories	Devuelve las subcategorías de la categoría especificada por parámetros.
EditCategory	Actualiza la entidad categoría especificada por parámetros.

DeleteCategory	Elimina la entidad categoría especificada.
GetReports	Devuelve los reportes pertenecientes a la categoría especificada.
DeleteReport	Elimina el reporte especificado.

Tabla 5 Descripción de la clase controladora ReportConnectorDAL.

3.3.3 Clases Entidades

Las clases entidades tienen como principal objetivo describir el modelo de objetos relacionados con el negocio. Dentro de este proyecto se encuentran definidas tanto las entidades de negocio que pasan a la persistencia como aquellas que están constituidas con composiciones y agregaciones de las antes mencionadas y que solo tienen vida en el transcurso de un proceso. Son los contenedores de información que fluyen a través de las capas de Presentación, Procesos-Servicios y Negocio.

Nombre	Category	
Descripción	Contiene los datos correspondientes a una categoría	
Atributos	Tipo de Dato	Descripción del atributo
idCategory	Guid	Identificador de la categoría.
name	string	Nombre de la categoría
description	string	Descripción de la categoría
parent	Category	Categoría padre

Tabla 6 Descripción de la clase entidad Category.

Nombre	IdentityReport	
Descripción	Contiene los datos correspondientes a un Reporte	
Atributos	Tipo de Dato	Descripción del atributo
idReport	Guid	Identificador del reporte
name	string	Nombre del reporte
description	string	Descripción del reporte
category	Category	Categoría contenedora del reporte
reportForm	ReportForm	Formulario de captura de parámetros del reporte.
rptName	string	Nombre del archivo de Crystal Report que utiliza este reporte.

Tabla 7 Descripción de la clase entidad IdentityReport.

Nombre	ReportForm	
Descripción	Contiene los datos del formulario de captura de parámetros	
Atributos	Tipo de Dato	Descripción del atributo

reportParameters	List<ReportParameter>	Representa los parámetros que necesita el documento de Crystal Report y que no se han tratado todavía.
fields	List<ParameterField>	Parámetros asociados a un control dentro del formulario de captura de parámetro.

Tabla 8 Descripción de la clase entidad ReportForm.

Nombre	ReportParameter	
Descripción	Representa un parámetro del documento de Crystal Report	
Atributos	Tipo de Dato	Descripción del atributo
name	string	Nombre del parámetro
type	string	Control asociado al parámetro
typeValue	string	Campo a utilizar como fuente de datos
connection	string	Representa la conexión que utilizará el control

Tabla 9 Descripción de la clase entidad ReportParameter.

Nombre	ParameterField	
Descripción	Representa un parámetro del documento de Crystal Report asociado a un control	
Atributos	Tipo de Dato	Descripción del atributo
parameter	ReportParameter	Nombre del parámetro

Tabla 10 Descripción de la clase entidad ParameterField.

Nombre	TextBoxField	
Descripción	Representa un parámetro del documento de Crystal Report asociado a un control textbox	
Atributos	Tipo de Dato	Descripción del atributo
parameter	ReportParameter	Nombre del parámetro

Tabla 11 Descripción de la clase entidad TextBoxField.

Nombre	ComboBoxField	
Descripción	Representa un parámetro del documento de Crystal Report asociado a un control dropdownlist	
Atributos	Tipo de Dato	Descripción del atributo
parameter	ReportParameter	Nombre del parámetro

Tabla 12 Descripción de la clase entidad ComboBoxField.

Nombre	CheckBoxField	
Descripción	Representa un parámetro del documento de Crystal Report asociado a un control checkbox	
Atributos	Tipo de Dato	Descripción del atributo

parameter	ReportParameter	Nombre del parámetro

Tabla 13 Descripción de la clase entidad CheckBoxField.

Nombre	ReportRender	
Descripción	Componente que se utiliza para visualizar una ReportForm	
Atributos	Tipo de Dato	Descripción del atributo
parameter	ReportForm	Formulario que visualizará

Tabla 14 Descripción de la clase entidad ReportRender.

3.1 Servicios del sistema

Un servicio no es más que una función, auto-contenida, que acepta llamada(s) y devuelve respuesta(s) mediante una interfaz bien definida que lo comunica con sistemas externos a él, independientemente del lenguaje de programación o tecnología con la que se trabaje. La utilización de servicios provee una mejora en los tiempos de realización de cambios en procesos, facilidad en modelos de negocios basados en colaboración con otros entes y colaboración con terceros y facilidad para reemplazar elementos en nuestro servicio sin consecuencias en el proceso de negocio. (21)

Atendiendo a las funcionalidades que implementan, los servicios del módulo de reportes han sido divididos en dos grupos: Servicios de Tiempo Real (*Runtime Services*) y Servicios de Negocio (*Business Services*).

Servicios de Tiempo Real (*Runtime Services*): Estos servicios representan un factor clave dentro de la aplicación ya que son los que permiten la interacción con las instancias de procesos creadas.

Servicios de Negocio (*Business Services*): Estos servicios son los que interactúan con los *workflows*, es decir, son los servicios que atienden las peticiones de las actividades HandleExternalEvent y las CallExternalMethod. Se encargan de comunicar el flujo de procesos con las capas inferiores y gestionan junto a las clases controladoras y conectoras la información contenida en la base de datos.

3.1.1 Descripción de los servicios del sistema

Servicios de Tiempo de Ejecución

Nombre	IReportRServices
Descripción	
Controla los procesos referentes a la gestión en el Módulo de Reportes	
Métodos	Argumentos
Load(Guid instanceId, string nameRpt)	EventHandler<StringEventArgs> OnLoad
Terminate(Guid instanceId)	EventHandler<ExternalDataEventArgs> OnTerminate
AcceptLoad(Guid instanceId)	EventHandler<ExternalDataEventArgs> OnAcceptLoad
Asociate(Guid instanceId, string control,string parameter)	EventHandler<FieldEventArgs> OnAsociate
Delete(Guid instanceId, string parameterName)	EventHandler<StringEventArgs> OnDelete
Save(Guid instanceId)	EventHandler<ExternalDataEventArgs> OnSave
SaveReport(IdentityReport report)	
ReportForm LoadRpt(string nameRpt)	

Tabla 15 Descripción del servicio de Tiempo de Ejecución IReportRServices

Servicios de Negocio

Nombre	IReportServices
Descripción	
Se encarga de llevar a cabo la persistencia de las entidades del Módulo de Reportes	
Métodos	
SaveReport(IdentityReport report)	
LoadReport(Guid idReport)	
DeleteReport(string idReport)	
SaveCategory(Category category)	
EditCategory(Category category)	
DeleteCategory(Category category)	
GetReports(Guid idCategory)	
GetCategories(Guid idCategory)	
GetCategory(Guid idCategory)	

Tabla 16 Descripción del servicio de negocio IReportServices

3.1.2 Diagrama de clases del diseño

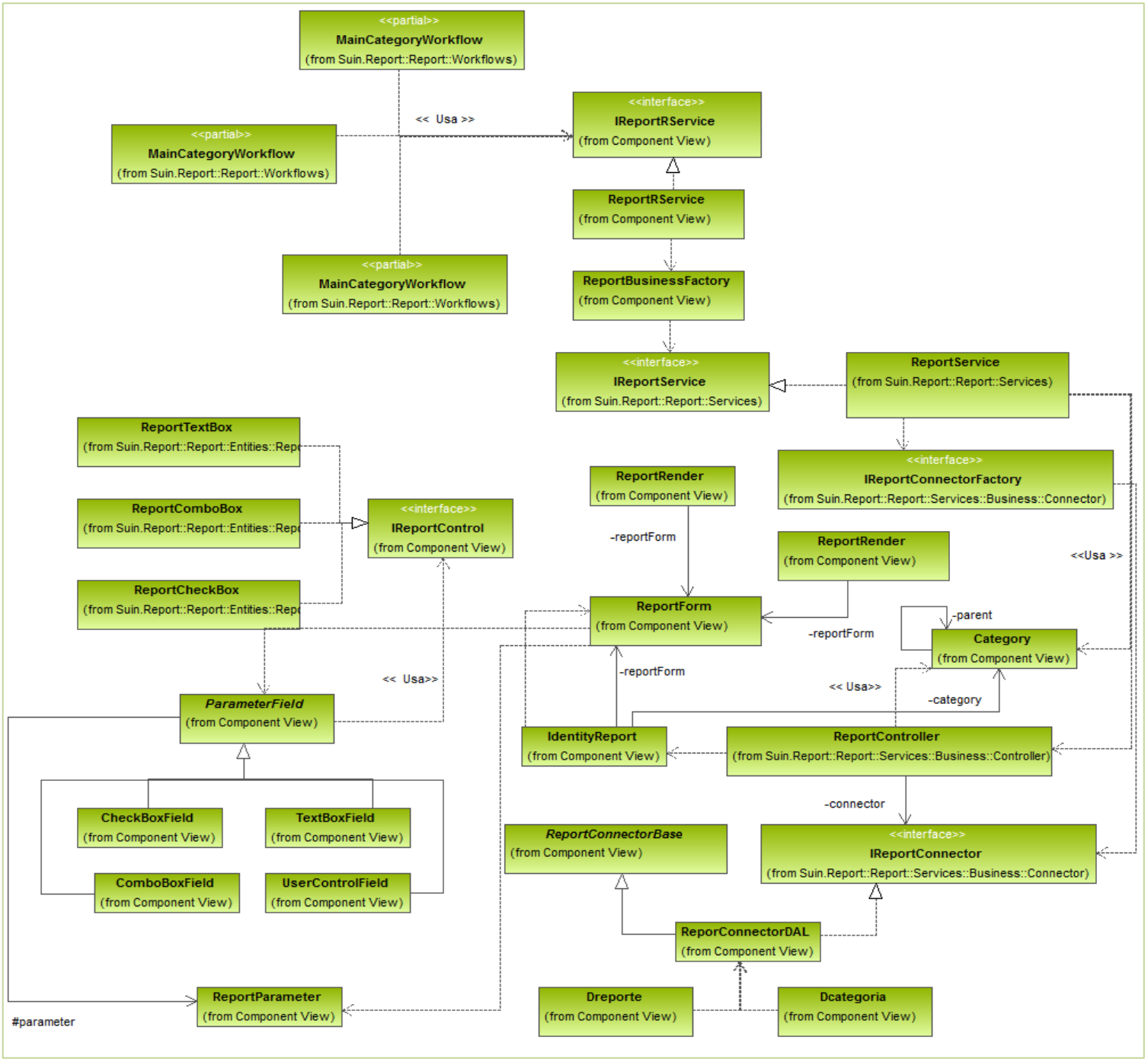


Fig. 16 Diagrama de clases del diseño

3.2 Diseño de Workflow

Windows Workflow Foundation es un marco que permite a los usuarios crear flujos de trabajo de sistemas o humanos en sus aplicaciones. Contiene un conjunto predeterminado de actividades que proporcionan la funcionalidad para el flujo de control, las condiciones, el control de eventos, la administración de estados, y la comunicación con aplicaciones y servicios. Al diseñar flujos de trabajo, se pueden utilizar actividades proporcionadas por Windows Workflow Foundation o crear actividades personalizadas.

Algunas de las actividades proporcionadas por Windows Workflow Foundation son:

- Condicionales (IfElseActivity, Clases Policy Activity).
- Bucles *While Repeat – Until*(Clases WhileActivity, ReplicatorActivity)
- Disparo y Captura de excepciones(Clases ThrowActivity, FaultHandlerActivity)
- Manejo de Eventos(HandleExternalEventActivity, EventDrivenActivity)
- Ámbito de Ejecución(Clase SincronizationScopeActivity) (22)

Las actividades son las unidades de creación fundamentales de los flujos de trabajo. Un flujo de trabajo no es más que conjunto de actividades que están organizadas jerárquicamente en una estructura de árbol. Una actividad representa una acción en un flujo de trabajo. Puede ser una acción simple, como un retraso, o una actividad compuesta que se compone de varias actividades secundarias. (22)

3.2.1 Actividades de WF utilizadas en el Módulo de Reportes

IfElseActivity

La actividad IfElseActivity ejecuta condicionalmente una de varias bifurcaciones alternativas. Coloca una condición en cada actividad IfElseBranchActivity. Si la condición se evalúa como true, se ejecutan las actividades contenidas en la actividad IfElseBranchActivity; de lo contrario, se evalúa la condición IfElseBranchActivity siguiente, y así sucesivamente. No tiene que colocar una condición en la última actividad IfElseBranchActivity porque se trata como la bifurcación Else. (25)

IfElseBranchActivity

La actividad IfElseBranchActivity representa una bifurcación de una actividad IfElseActivity que se ejecuta si la propiedad Condition de la actividad IfElseBranchActivity está establecida como true. Puede

agregar tantas actividades IfElseBranchActivity como desee a una actividad IfElseActivity, y puede agregar tantas actividades como desee a cada actividad IfElseBranchActivity. (25)

WhileActivity

La actividad WhileActivity ejecuta de manera iterativa una actividad secundaria única siempre que su propiedad Condition sea true. Una condición de regla o una condición de código que está asociada a la propiedad Condition se evalúa antes de cada iteración del bucle WhileActivity. Si la condición se evalúa como true, se ejecuta la actividad secundaria. Si la condición se evalúa como false, no se ejecuta la actividad secundaria y se finalizan las actividades WhileActivity. (25)

FaultHandlerActivity

La actividad FaultHandlerActivity atiende una excepción del tipo especificado. La actividad FaultHandlerActivity es un contenedor para otras actividades que realizan el trabajo que se hace cuando se produce la excepción especificada. (25)

CallExternalMethodActivity

La actividad CallExternalMethodActivity y la actividad HandleExternalEventActivity se pueden usar para comunicaciones de entrada y salida con un servicio local. Estas actividades se utilizan para comunicaciones genéricas o para crear actividades que estén enlazadas estrictamente a eventos y métodos concretos en una interfaz que se atribuye con el atributo ExternalDataExchangeAttribute. (25)

HandleExternalEventActivity

La actividad HandleExternalEventActivity se usa en conjunción con la actividad CallExternalMethodActivity para las comunicaciones de entrada y salida con un servicio local. Estas actividades se utilizan para comunicaciones genéricas o para crear actividades que estén enlazadas estrictamente a eventos y métodos concretos en una interfaz que se atribuye con el atributo ExternalDataExchangeAttribute. (25)

CodeActivity

La actividad CodeActivity es un forma sencilla de actividad personalizada que permite agregar fácilmente código de C# o de Visual Basic a cualquier parte de un flujo de trabajo. El código introducido en la

actividad CodeActivity reside en el archivo con separación de código que se compila con el flujo de trabajo. (25)

ClientActivity

Permite definir las interacciones del proceso con los usuarios de la aplicación. Estas pantallas pueden ser de diferentes tipos. Para su uso es necesario especificarle la url del componente representado. (26)

RegisterWorkflowAsQueryableActivity

Permite activar una puerta de entrada para obtener la información necesaria del workflow. Esta actividad tiene como base las colas utilizadas por el workflow para realizar sus acciones. Una vez que la instancia ha ejecutado la actividad, se queda registrado dentro de la cola de esta un objeto que responde a las peticiones de información enviadas al workflow. (26)

AssociationActivity

Representa la asociación de un valor a una propiedad dentro del workflow. El valor de origen puede ser una variable de entrada o bien un valor string que sea pasado en la propiedad value. Además de representar la asociación de valores de manera gráfica también se encarga de dejar una traza de los valores que han sido modificados a lo largo del proceso siempre que la propiedad IsTreaceble tenga valor true. (26)

3.2.2 Transformación del modelo de proceso mejorado a Workflow Foundation

En la transformación del modelo de proceso de mejora del módulo de reportes al diseño del workflow, para cada acción del proceso, se pudieran desencadenar una o varias actividades, en dependencia de la complejidad del proceso en el que se esté trabajando y la cantidad de entidades que estemos manejando. A continuación se muestra el diseño del workflow para el proceso adicionar reporte.

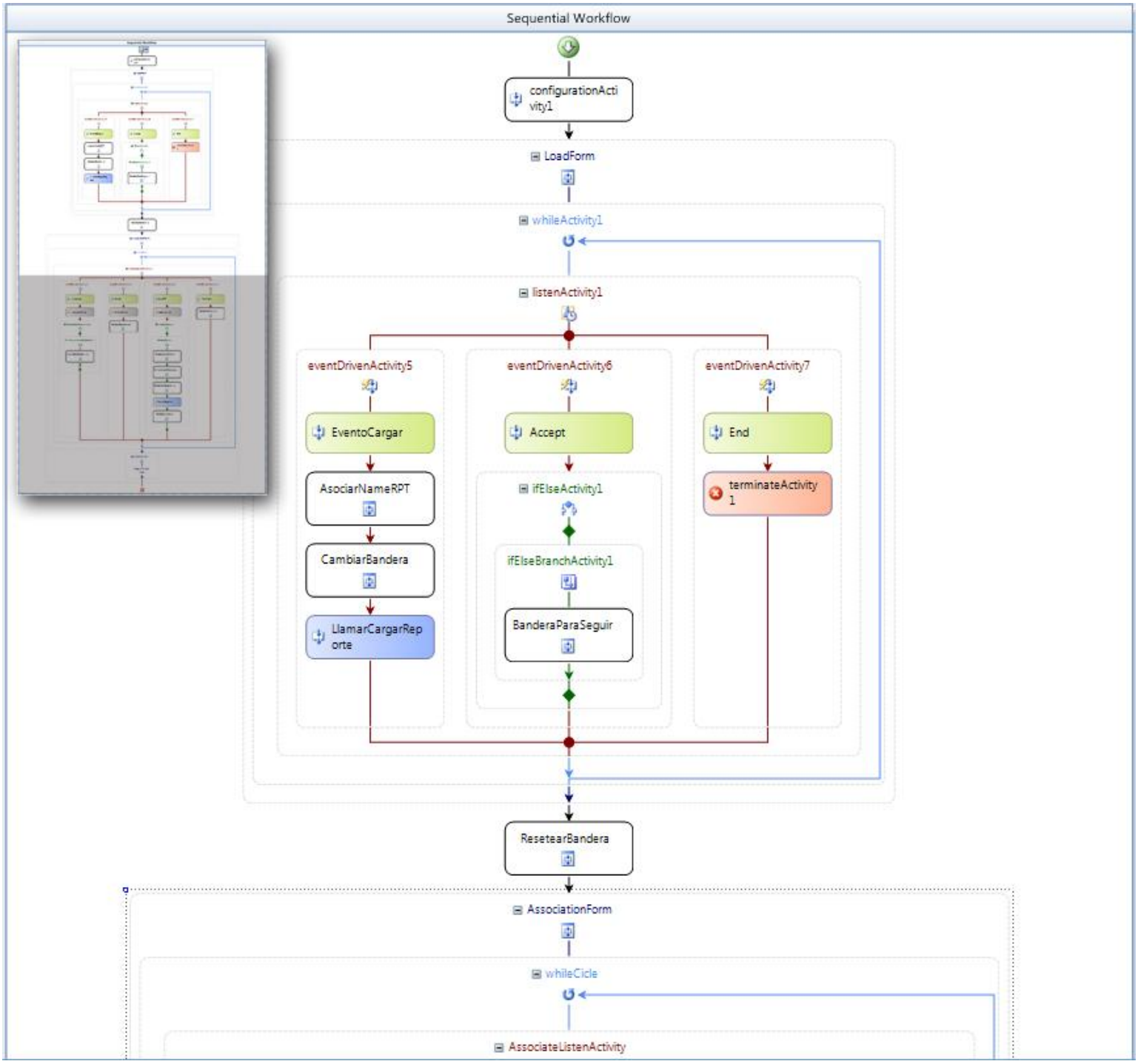


Fig. 17 Diseño del *workflow* de Adicionar Reporte imagen 1

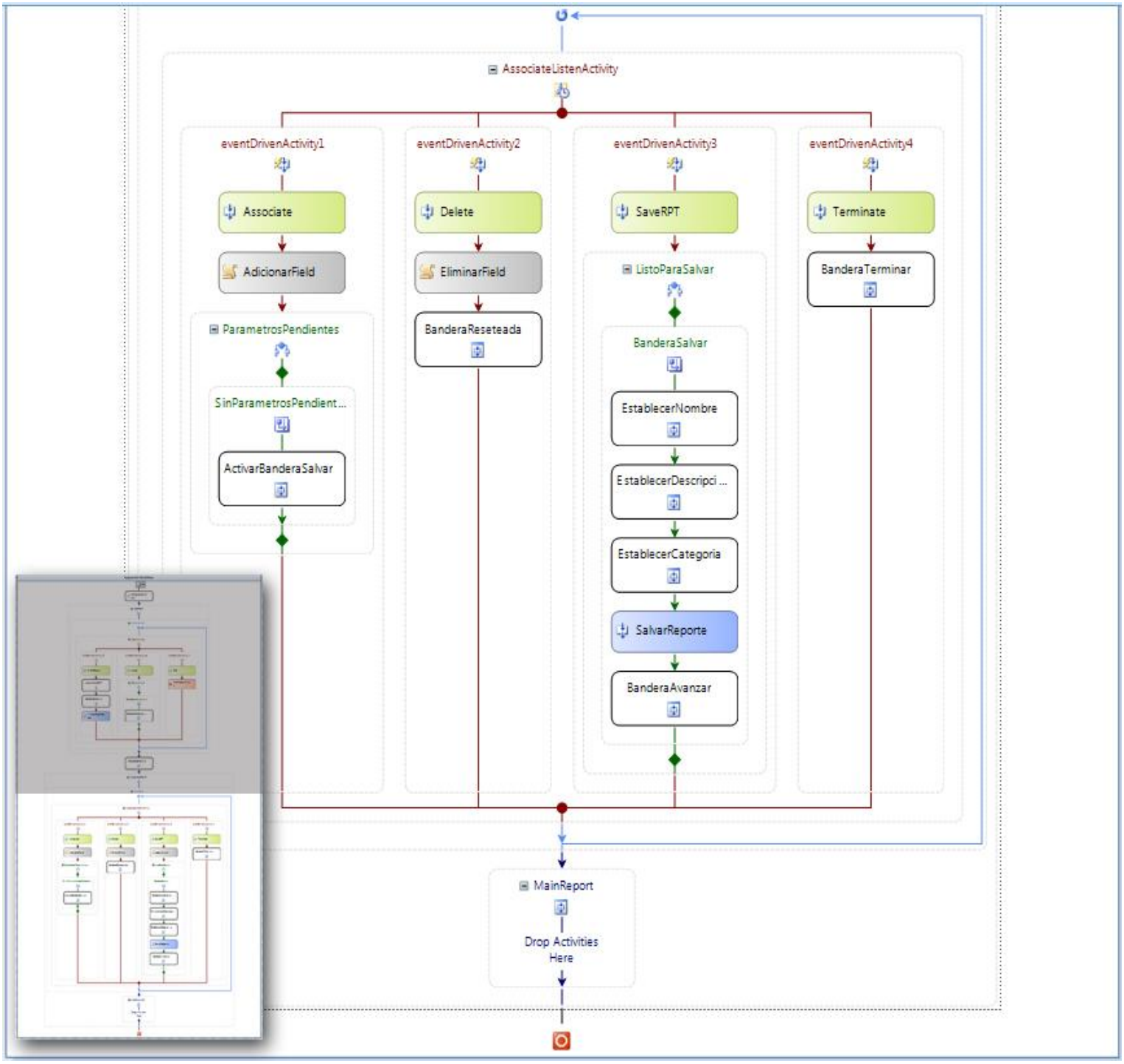


Fig. 18 Diseño del *workflow* de Adicionar Reporte imagen 2

3.3 Modelo de datos

Un modelo de datos es un lenguaje orientado a describir una Base de Datos. Típicamente un modelo de datos permite describir:

- Las estructuras de datos: El tipo de los datos que hay en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- Operaciones de manipulación de los datos: Operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

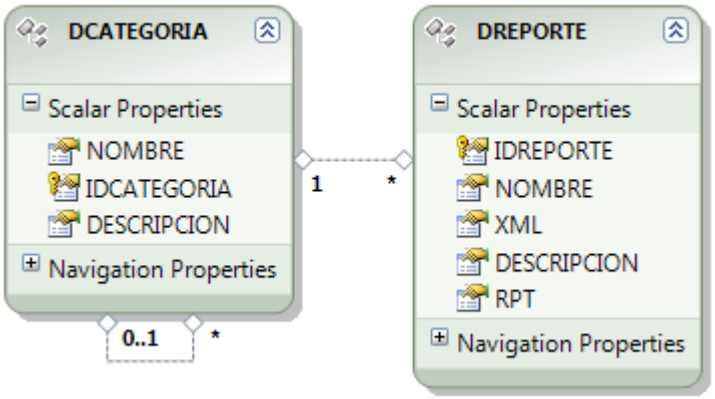


Fig. 19 Entidades más relevantes del modelo de datos del Módulo de Reportes del SUIN

En la Fig. 19 Entidades más relevantes del modelo de datos del Módulo de Reportes del SUIN se muestran las entidades más importantes con la que interactúa la aplicación. El mismo fue elaborado a partir del modelo de entidades conceptuales presentado en el capítulo 2 y que muestra la relación entre las entidades DREPORTE y DCATEGORIA, entidades que establecen una relación directa con la aplicación.

3.3.1 Descripción de las entidades fundamentales

Tabla	DREPORTE
Descripción	Representa a los reportes almacenados en la aplicación
Atributos	IdReporte, Nombre, Descripción, RPT, XML

Tabla 17 Descripción de la entidad DREPORTE

Tabla	DCATEGORIA
Descripción	Representa a las diferentes categorías a las que pertenecerán los reportes de la aplicación.
Atributos	IdCategoría, Nombre, Descripción

Tabla 18 Descripción de la entidad DCATEGORIA

3.4 Conclusiones

❖ El estudio de este capítulo permite entender la estructura organizativa de la arquitectura de la aplicación así como los módulos que lo conforman y que unidos, integran el sistema con sus capacidades.

❖ Se describen las clases en las cuales se soporta el desarrollo del sistema propuesto especificando su tipo (entidades, controladoras y gestoras) y funciones especificas dentro de la aplicación. Se detallan los servicios de negocio y de tiempo real que sirven de sustento al cumplimiento de las exigencias planteadas por el cliente.

❖ Se muestra el diseño del *workflow* obtenido a partir del diagrama de mejora de procesos donde cada actividad da cumplimiento a requerimientos planteados en el capítulo Características del Sistema.

❖ Se obtiene el modelo de datos real de la aplicación, observándose pocos cambios a partir del modelo de entidades conceptuales mostrado en el Capítulo 2.

Capítulo 4: Implementación y Prueba

Introducción

El presente capítulo muestra los estándares de codificación utilizados en el desarrollo del sistema, así como una representación de las dependencias existentes entre elementos de implementación y los correspondientes elementos de diseños que son implementados. Muestra también las relaciones físicas de los distintos nodos que componen el sistema y el reparto de los componentes sobre dichos nodos, conformando de esta forma un modelo de implementación que permita describir los componentes a construir, sus organizaciones y dependencias entre los diferentes nodos físicos en los que la aplicación funcionará. Se lleva a cabo además, la tarea de validación de las especificaciones del cliente a través de las pruebas del sistema, lo que permite verificar el cumplimiento de los requerimientos definidos y garantizar la calidad del software desarrollado.

4.1 Estándares de codificación y tratamiento de errores

Un patrón de nomenclatura coherente es uno de los elementos más importantes de la previsibilidad y el descubrimiento de una biblioteca de clases administradas. El uso generalizado y la comprensión de estas directrices de denominación deberían eliminar muchas de las preguntas más comunes. Varios de los estándares de codificación que se aplican al Módulo de Reportes son:

4.1.1 Estilos para la capitalización

Se podrán utilizar los siguientes tres convenios para la capitalización de los identificadores:

- **Pascal**

La primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan. Puede utilizar los identificadores de Pascal en caso de tres o más caracteres. Por ejemplo:

```
SaveReport
```

- **Camello**

La primera letra en el identificador está en minúscula y la primera letra de cada subsiguiente palabra concatenada es mayúscula. Por ejemplo:

```
reportParameters
```

- **Mayúscula**

Todas las letras en el identificador se capitalizan. Esta convención se utilizará solo para los identificadores que constan de dos o menos letras. Por ejemplo:

```
System.Web.UI

System.IO
```

4.1.2 Sensibilidad a mayúsculas

- No se deberá utilizar nombres o identificadores que requieran ser *case sensitivity*[11]
- No se deberá crear dos *namespaces*[12] que se diferencien solo en el uso de las mayúsculas. Por ejemplo:

```
Report.entities.Reportcontrols

Report.Entities.ReportControls
```

- No crear funciones con nombres de parámetros que se diferencian solo en el uso de la mayúscula. Por ejemplo:

```
EditCategory(Category Category, Category category)
```

- No se deberá crear *namespaces* con nombres de clases que se diferencien solo en el uso de las mayúsculas. Por ejemplo:

```
Report.Entities

Report.ENTITIES
```

- No crear clases con propiedades que se diferencien solo en el uso de las mayúsculas. Por ejemplo:

```
string Description{get, set}

string DESCRIPTION{get, set}
```

[11] sensible a las mayúsculas o minúsculas
[12] espacio de nombres

- No crear clases con métodos que se diferencien solo en el uso de las mayúsculas.

4.1.3 Evitando confusión de nombre y tipo

- Utiliza nombres que describan a sus identificadores en vez de nombres que describen el tipo de identificador. En determinados casos donde el parámetro no tiene un significado semántico distinto al de su tipo se usa un nombre genérico. Por ejemplo, una clase que soportará la escritura de gran variedad de tipos de datos debe contener los siguientes métodos. Por ejemplo:

```
Void Write (double value);

Void Write (float value);

Void Write (long value);

Void Write (int value);

Void Write (short value);
```

4.1.4 Tratamiento de excepciones

El diseño de un sistema no solo debe tener en cuenta lo que debe ocurrir, sino que además debe realizar un análisis profundo de las diferentes situaciones que se puedan presentar y que constituyen algún tipo de violación o de situación en particular que provocaría un error dentro del sistema.

Las excepciones como se menciona en el párrafo anterior son parte del diseño aun cuando la gran mayoría de ellas son identificadas en el proceso de implementación, no obstante es necesario definir con antelación un mecanismo efectivo para su tratamiento. En el caso del sistema que se está implementando, es necesario llevar una traza de todas aquellas excepciones que se lancen, y en muchos casos se requiere que usuario que esta interactuando con la aplicación en ese momento especifique alguna información que tribute a la futura validación o corrección de la excepción lanzada.

Para el tratamiento de excepciones se definirá una interfaz base que permite dejar una traza clara de todas las excepciones lanzadas dentro del sistema. Además se pueden considerar las excepciones como parte del flujo de negocio en muchos procesos por lo cual es necesario lograr identificar cada tipo

de excepción para darle un tratamiento en particular a cada una, para ello se debe definir una clase en particular para aquellas excepciones que sean de interés para el negocio. (23)

4.2 Diagrama de componentes

Un diagrama de componentes describe los elementos físicos del sistema y sus relaciones. Se utiliza para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados. (24)

De acuerdo con la arquitectura que brinda soporte a la aplicación, se elaboró el diagrama de componentes que se muestra en la Fig. 20 Diagrama de componentes del Módulo de Reportes. En él se puede apreciar la interrelación entre cada uno de los componentes utilizados en solución a las funcionalidades del Módulo de Reportes.

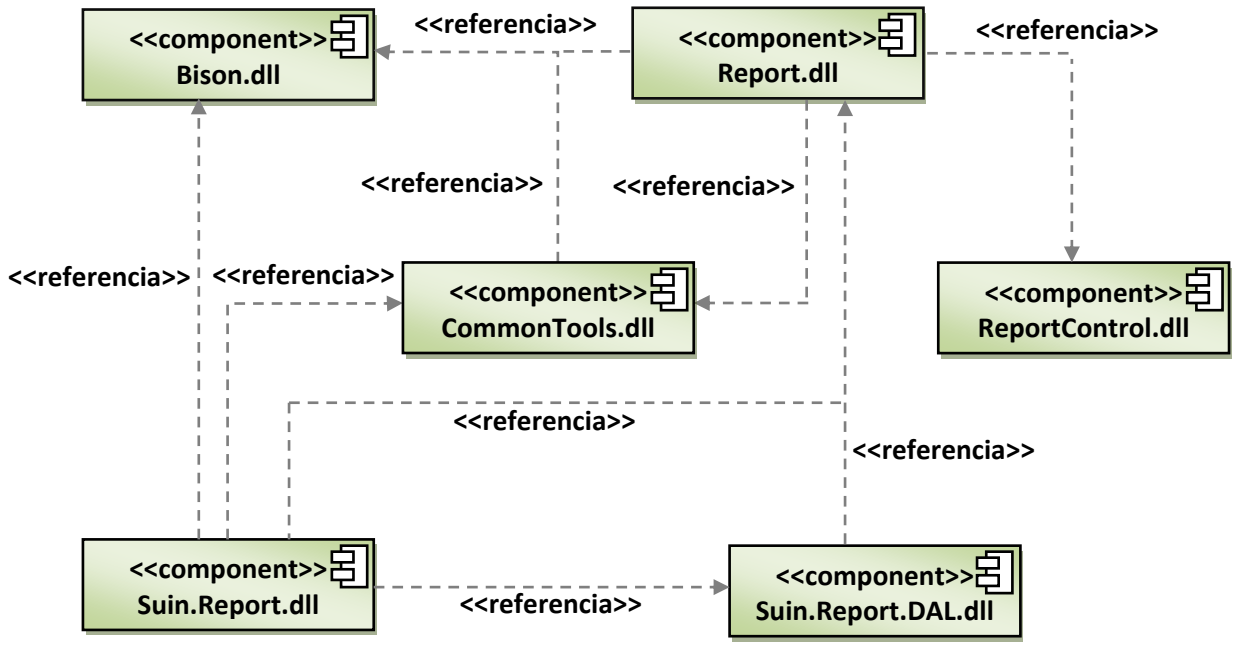


Fig. 20 Diagrama de componentes del Módulo de Reportes

4.3 Diagrama de despliegue

Un Diagrama de Despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Estos diagramas describen la arquitectura física del sistema durante la ejecución en términos de procesadores, dispositivos y componentes de software. (24)

En la Fig. 21 Diagrama de despliegue del Módulo de Reportes, se muestra el diagrama de despliegue de la aplicación. Aquí se puede apreciar la topología del sistema, o sea, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos, orientada de la siguiente manera. La máquina cliente se conectará al servidor de aplicaciones, encargado de manejar las funcionalidades del sistema y a una impresora desde donde se podrán imprimir los diferentes reportes que se visualicen desde el sistema, el servidor de aplicaciones se conectará al servidor de bases de datos provincial, de donde obtendrá la información necesaria para el cumplimiento de las funcionalidades y el servidor de bases de datos provincial se conectará al servidor nacional al que solicitará los datos que no sean de su dominio.

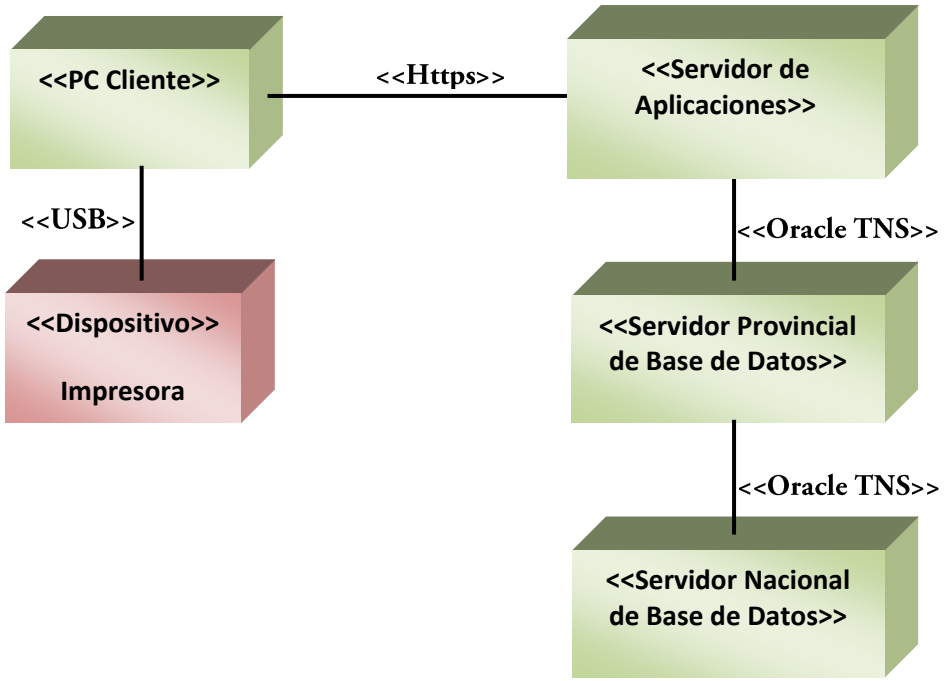


Fig. 21 Diagrama de despliegue del Módulo de Reportes

4.4 Interfaces del sistema

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema. La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca a un sistema al éxito o al fracaso. Para desarrollar una interfaz correcta para un programa específico se deben valorar y tener en cuenta determinados elementos que ayudaran a definir parámetros del uso de signos, colores, fuentes, imágenes y demás componentes que esta requiera (12). A continuación se muestra la interfaz adicionar reporte, una de las más importantes del Módulo de Reportes para el Sistema Único de Identificación Nacional.

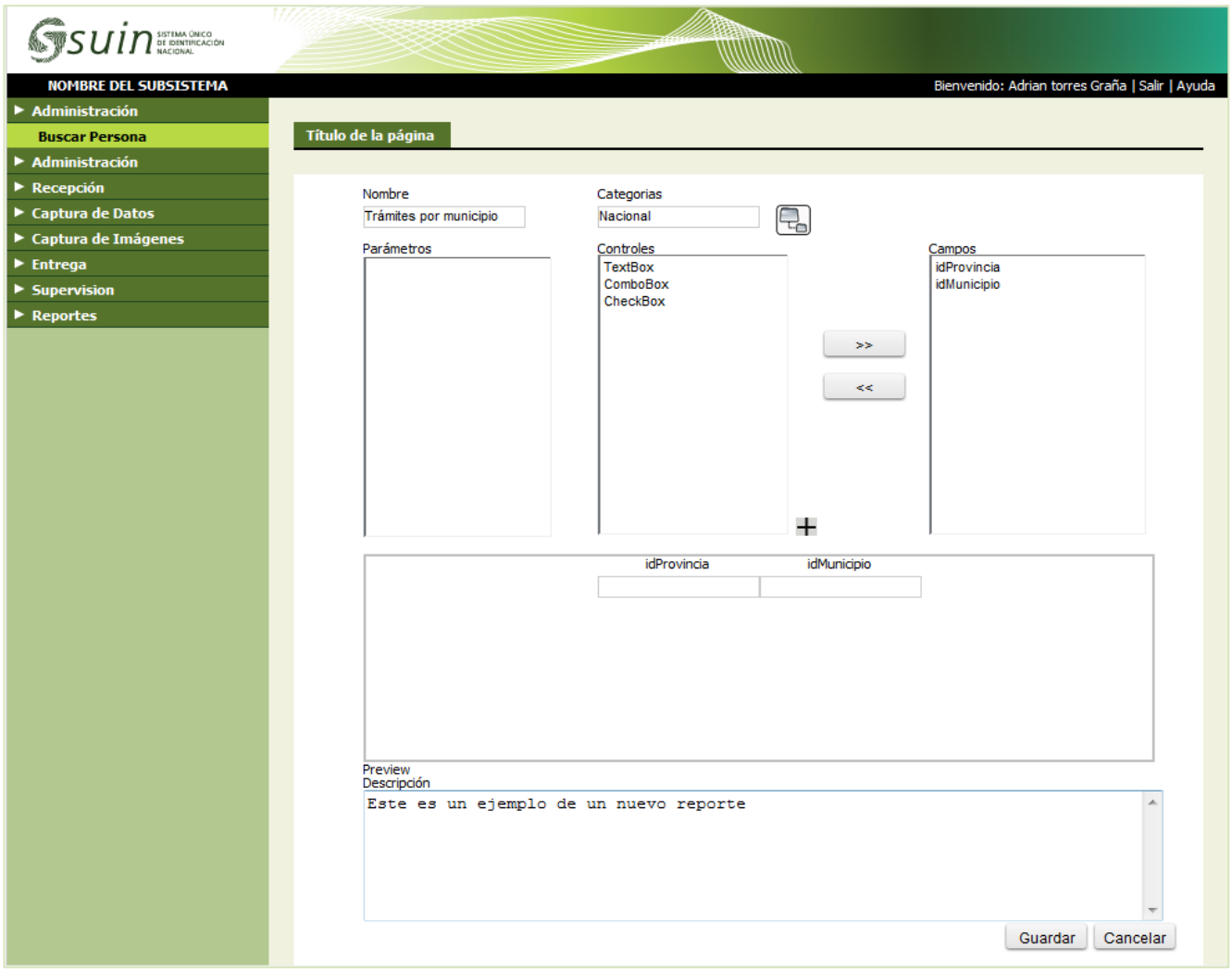


Fig. 22 Interfaz Adicionar Reporte

La interfaz de adicionar reporte, que se muestra en la Fig. 22 Interfaz Adicionar Reporte, permitirá a un usuario con los permisos necesarios adicionar un nuevo reporte al sistema. Para ello el usuario primero tendrá la opción de seleccionar el archivo de Cristal Report (.rpt) de donde el sistema extraerá las cadenas de conexión del reporte y los parámetros necesarios. Una vez seleccionado el archivo de Crystal Report, el usuario será capaz de diseñar el formulario de captura de parámetros del reporte, mediante la asociación de cada uno de los parámetros del reporte con un componente visual que capturará el valor del parámetro, para esto el usuario se puede auxiliar de los componentes propios del módulo. Una vez asociados el control y el parámetro, se selecciona una categoría para el reporte, se le realiza una breve descripción y se procede a almacenarlo en la base de datos.

4.5 Pruebas

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. Dentro de cada una de las etapas de desarrollo de un software las pruebas son fundamentales ya que a partir de ellas es posible controlar que los productos cumplan requisitos mínimos de operabilidad además de garantizar la calidad de estos productos (25).

Con el objetivo verificar el grado de satisfacción de los requerimientos de software del Módulo de Reportes se llevó a cabo el diseño de diferentes casos de prueba capaces de probar el software a lo ancho y en profundidad, de manera tal que se facilitara una evaluación suficiente de los elementos objetivos de las pruebas.

4.5.1 Diseño de Casos de Prueba

El diseño de casos de prueba consiste en la confección de distintos escenarios de prueba según la técnica o técnicas identificadas previamente. El diseño de cada caso de prueba debe ir acompañado del resultado que ha de producir el software al ejecutar dicho caso para detectar un posible fallo en el programa. Los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba (29).

Con el fin de probar el módulo en su totalidad fueron aplicados los dos métodos de prueba fundamentales: el método de caja blanca o caja de cristal y el método de caja negra.

4.5.1.1 Caja Blanca (Pruebas unitarias)

La prueba de caja blanca se centra en la estructura interna del programa para elegir los casos de prueba. Esta prueba, también llamada caja de cristal, ocupa un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para realizar sus casos de prueba. (25)

Para llevar a cabo las pruebas de caja blanca el equipo de desarrollo se apoyó en el conjunto de herramientas de prueba de Visual Studio Team System Test, específicamente en las pruebas unitarias (*unit test*).

Pruebas unitarias (*unit test*)

Las pruebas unitarias son una herramienta muy importante para el personal de pruebas y sobre todo para los desarrolladores. Estas pruebas utilizan el código, lo ejecutan y examinan los resultados obtenidos determinando si la prueba fue satisfactoria o no. Las pruebas unitarias son generadas por Visual Studio Team System para métodos específicos, incluidos los métodos privados.

Las pruebas unitarias se utilizan para ejecutar otro código fuente llamando directamente a los métodos de una clase, pasando los parámetros apropiados. A continuación, si incluye instrucciones *ASSERT*, éstas pueden probar los valores que se generan con respecto a los valores esperados. Los métodos de pruebas unitarias residen en clases Test, que se almacenan en archivos de código fuente. (16)

A continuación se muestra la prueba unitaria realizada al método *GetCategories* de la clase ReporConnectorDAL que se encarga de devolver las subcategorías de una categoría especificada. Para la realización de la prueba se creó un nuevo proyecto de prueba, ReportTest, desde el que, a través del método de prueba *GetCategoriesTest*, se probó la funcionalidad del método *GetCategories*.

Para el primer caso de prueba se le especificó al método *GetCategoriesTest* que ejecutara el método *GetCategories* con un identificador vacio, como se muestra en la Fig. 23 Método *GetCategoriesTest* para el primer caso de prueba, lo que significa que el método deberá devolver todas aquellas categorías que son subcategorías.

```
[TestMethod()]
public void GetCategoriesTest()
{
 ReporConnectorDAL target = new ReporConnectorDAL
 Guid idCategory = Guid.Empty
 List<Hashtable> expected = new List<Hashtable> ();
 List<Hashtable> actual;
 actual = target.GetCategories(idCategory);
 Assert.AreNotEqual(expected, actual);

}
```

Fig. 23 Método GetCategoriesTest para el primer caso de prueba

Para el segundo caso de prueba se probó el método GetCategories introduciendo el identificador de la categoría Nacional, la cual no cuenta con subcategorías, por lo que el resultado para este caso sería una lista vacía. En esta ocasión el método GetCategoriesTest quedó implementado como se muestra en la

Fig. 24 Método GetCategoriesTest para el segundo caso de prueba.

```
[TestMethod()]
public void GetCategoriesTest()
{
 ReporConnectorDAL target = new ReporConnectorDAL();
 Guid idCategory = new Guid("96885bc3-4ec8-433e-b69f-fc3e7a8a633a");
 List<Hashtable> expected = new List<Hashtable>();
 List<Hashtable> actual;
 actual = target.GetCategories(idCategory);
 Assert.AreEqual(expected, actual);
}
```

Fig. 24 Método GetCategoriesTest para el segundo caso de prueba

4.5.1.2 Caja Negra (Pruebas de sistema)

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (25).

A continuación se muestran los diseños de prueba de caja negra correspondientes al Módulo de Reportes para los requisitos Adicionar Reporte y Mostrar Reporte. Para mayor información ver Anexo 2: Diseño de Casos de Prueba.

CPR 6: Adicionar Reporte.

Condiciones de Ejecución:
El administrador de reportes debe estar autenticado en el sistema.

Nombre del Flujo	Escenarios del flujo	Descripción de la funcionalidad	Flujo central
Adicionar reporte	1.1 Adicionar reporte sin parámetros	Permite adicionar un reporte	1. Mostrar pantalla para adicionar reporte. 2. Permitir seleccionar reporte. 2.1 Si no tiene parámetros. 2.2 Permitir adicionar reporte.
	1.1.a Adicionar reporte con parámetros	Permite adicionar un reporte	1. Si tiene parámetros. 1.1 Diseñar reporte. 1.2 Adicionar reporte

Tabla 19 Descripción del caso de prueba de Caja Negra para el requisito funcional Adicionar Reporte

CPR 7: Mostrar Reporte.

Condiciones de Ejecución:
El administrador de reportes debe estar autenticado en el sistema.
El funcionario de reportes debe estar autenticado en el sistema.

Nombre del Flujo	Escenarios del flujo	Descripción de la funcionalidad	Flujo central
1. Mostrar reporte	1.1 Mostrar reporte sin parámetros	Permite mostrar un reporte	1.Mostrar reportes existentes 2.Permitir seleccionar reporte 3.Si no tiene parámetros 3.1 Mostrar reporte
	1.1.a Mostrar reporte con parámetros	Permite mostrar un reporte	1. Mostrar reportes existentes 2. Permitir seleccionar reporte 3. Si tiene parámetros 3.1 Permitir introducir parámetros

			3.2 Mostrar reporte

Tabla 20 Descripción del caso de prueba de Caja Negra para el requisito funcional Mostrar Reporte

4.5.2 Iteraciones

Para comprobar la calidad del software se realizaron tres iteraciones de prueba donde se validó la funcionalidad del sistema de acuerdo a los requisitos definidos. El objetivo fundamental de estas iteraciones es encontrar no conformidades y registrarlas para posteriormente satisfacerlas.

4.5.2.1 Resultados de las pruebas

El primer caso de prueba de caja blanca se ejecutó satisfactoriamente arrojando como resultado que la prueba había pasado, tal y como se muestra en la Fig. 25 Ejecución del primer caso de prueba.

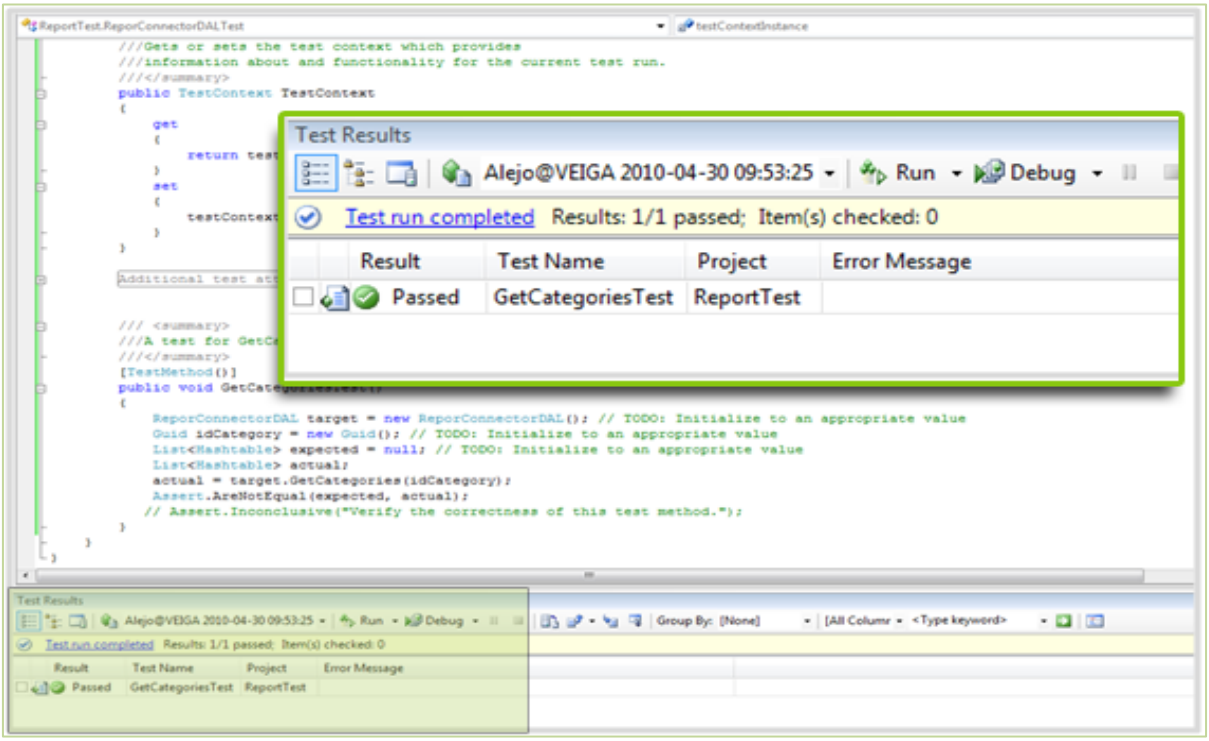


Fig. 25 Ejecución del primer caso de prueba

Como se puede apreciar en la Fig. 26 Ejecución del segundo caso de prueba, el caso de prueba dos de caja blanca también pasó, comprobando que con esos parámetros el método se había ejecutado correctamente.

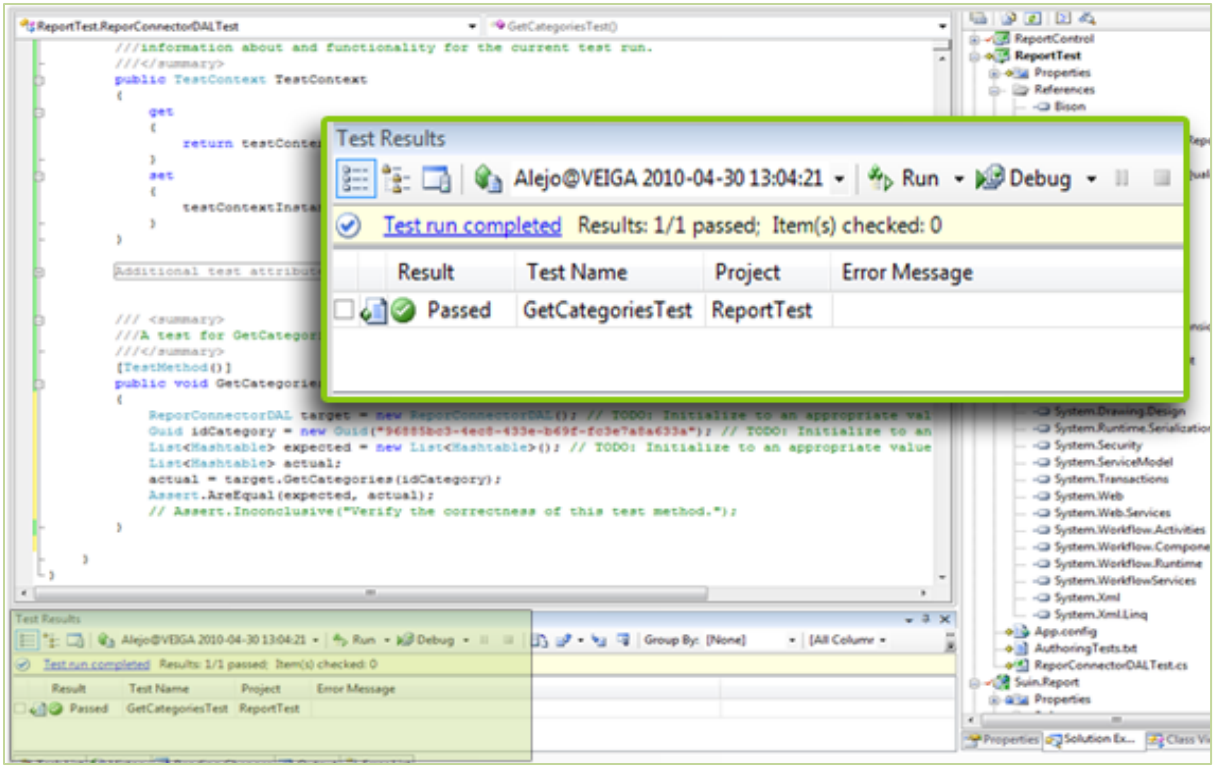


Fig. 26 Ejecución del segundo caso de prueba

A continuación se muestran los resultados de los casos de prueba de caja negra.

ID del Escenario	Tipos de Clases	Clases	Resultado Esperado	Resultado de la Prueba	Observaciones
1.1 Adicionar reporte sin parámetros	Clases Válidas	Debe permitir seleccionar un reporte.	Permite seleccionar un reporte para adicionarlo.	Permite adicionar un reporte luego de haber sido seleccionado.	-
1.1.a Adicionar reporte con parámetros		Debe permitir seleccionar un reporte Debe permitir diseñar el reporte	Permite seleccionar y diseñar un reporte para adicionarlo	Permite seleccionar y diseñar un reporte.	-

1.1 Adicionar reporte sin parámetros	**Clases Inválidas**	No permite seleccionar un reporte	No permite seleccionar un reporte para adicionarlo.		-
1.1.a Adicionar reporte con parámetros		No permite seleccionar un reporte No permite diseñar el reporte	No permite seleccionar y diseñar un reporte para adicionarlo		-

Tabla 21 Resultado de la descripción de la prueba Adicionar Reporte

ID del Escenario	Tipos de Clases	Clases	Resultado Esperado	Resultado de la Prueba	Observaciones
1.1 Mostrar reporte sin parámetros	**Clases Válidas**	Debe mostrar los reportes existentes.	Permite seleccionar un reporte para mostrarlo.	Permite seleccionar y mostrar el reporte.	-
1.1.a Mostrar reporte con parámetros		Debe mostrar los reportes existentes. Debe permitir introducir parámetros.	Permite seleccionar un reporte para mostrarlo.	Permite mostrar el reporte seleccionado e introducir los parámetros que requiere.	-
1.1 Mostrar reporte sin parámetros	**Clases Inválidas**	No muestra los reportes existentes.	No permite seleccionar un reporte para mostrarlo.		-
1.1.a Mostrar reporte con parámetros		No muestra los reportes existentes. No permite introducir parámetros.	No permite seleccionar un reporte para mostrarlo.		-

Tabla 22 Resultado de la descripción de la prueba Mostrar Reporte

4.6 Conclusiones

- En el presente capítulo se abordaron temas referentes a los estándares de codificación y tratamientos de errores utilizados para la implementación del sistema.
- Se especificaron las dependencias existentes entre los distintos componentes que conforman el sistema, así como las relaciones físicas de los nodos que integran el Módulo de Reportes.
- Se evaluó el cumplimiento de las funcionalidades del sistema de acuerdo a los diseños de casos de prueba definidos.
- Se mostraron las interfaces finales con las que trabajará el usuario final de la aplicación luego de haber culminado el proceso de desarrollo del software.

Conclusiones Generales

En la actualidad la manipulación de la información y su presentación de forma dinámica constituye un elemento fundamental en la proyección y desarrollo de las empresas que utilizan sistemas gestores de información. Con la realización del presente trabajo de diploma se ha desarrollado una primera versión del Módulo de Reportes del Sistema Único de Identificación Nacional, el cual brinda soporte a los restantes módulos que integran el SUIN y permite el manejo y visualización de los datos almacenados.

- Se realizó un extenso estudio de los sistemas encargados de gestionar reportes, a partir de lo cual se llegó a la conclusión de la necesidad de implementación de un Módulo de Reportes.
- Se fundamentaron las tecnologías utilizadas para el desarrollo de la aplicación basando su estudio en las facilidades que ofrecían a un desarrollo integrado.
- Se obtuvo el modelo de dominio correspondiente a los procesos que se realizan en las oficinas de Carné de Identidad y Registro de la Población referente a los reportes, lo que permitió una mejor comprensión de los conceptos con que trabajan los usuarios y con los que trabajará la aplicación.
- Se obtuvo el catálogo de requisitos funcionales y no funcionales a partir de las entrevistas con el cliente, con el objetivo de lograr un mayor entendimiento entre clientes y desarrolladores como premisa para lograr un producto con la calidad requerida.
- Se definió la arquitectura del Módulo de Reportes así como los elementos de construcción del software que dan soporte al desarrollo.
- Se probaron las funcionalidades del Módulo de Reportes a través de los diseños de casos de prueba y pruebas unitarias, con el objetivo verificar el grado de satisfacción de los requerimientos de software.
- Se obtuvo como resultado una primera versión del Módulo de Reportes para el SUIN que facilita el análisis de la información almacenada.

Recomendaciones

Como resultado del sistema implementado se hacen las siguientes recomendaciones:

- Continuar con la implementación del sistema a fin de concluir las funcionalidades identificadas que no se satisfacen con esta primera versión del Módulo de Reportes para el SUIN.
- Seguir trabajando en el diseñador del formulario de captura de parámetros de manera tal que permita la inserción de nuevos controles implementados por el usuario.
- Trabajar en la asignación de la fuente de datos del control, de forma que pueda ser definida dinámicamente.
- Eliminar las dependencias del módulo con el SUIN y lograr una aplicación totalmente portable que pueda ser utilizada para el procesamiento de datos en diferentes ambientes.

Bibliografía

1. **Morales Ramírez, Yanet y Peró Chibás, Vladimir.** *Ficha técnica del sistema Info@tletas.* Ciudad Habana : s.n.

2. Atta. *Atta.* [En línea] Instituto Nacional de Biodiversidad, 2001. [Citado el: 25 de Octubre de 2009.] http://atta.inbio.ac.cr/.

3. **Extranjería, Equipo de Desarrollo. Servicio Administritativo de Identificación Migración y.** *Manual de Usuario del Módulo Reportes del SAIME .*

4. **Pérez, Agdarys Gonsalez.** *Metodología y funcionamiento del Sistema Automatizado del Carné de Identidad y Registro de la Población.* Ciudad Habana : s.n.

5. *Metodología y funcionamiento del Sistema Automatizado del Carné de Identidad y.* Ciudad Habana : s.n.

6. **de Laurentiis Gianni, Renato.** *Workflow-Tecnología para la Integración y Orquestación de Procesos, Sistemas y Organización.* Madrid : IBERICA IT Group, 2002.

7. **Morales, Ing. Pablo.** *Arquitectura de procesos para modelos de Workflow.* Montevideo : Lithium Software.

8. **services, Centers for medicare and medicaid.** *SELECTING A DEVELOPMENT APPROACH.* 2008.

9. Microsoft Colombia. *Microsoft Colombia.* [En línea] 2000. [Citado el: 29 de 10 de 2009.] http://www.microsoft.com/colombia/portafolio/msf.htm.

10. **Huayta García, Madeleine L.** *CMMI: ASEGURAMIENTO DE LA CALIDAD.* Ciudad Habana : IV Simposio de Ingeniería Industrial, Informática y Afines, 2005.

11. Danysoft. *Danysoft.* [En línea] [Citado el: 5 de Enero de 2010.] http://shop.danysoft.com/Altova-UModel..

12. Monografías. *Monografías.* [En línea] 10 de Febrero de 2008. [Citado el: 6 de Enero de 2010.] http://www.monografias.com/trabajos56/sistemas-bases-de-datos/sistemas-bases-de-datos.shtml.

13. Oracle Latinoamérica. *Oracle Latinoamérica.* [En línea] http://www.oracle.com/global/lad/database/standard_edition.html.

14. **Nourie, Dana.** *Getting Started with an Integrated Development Environment.* s.l. : Sun Microsystems, 2008.

15. *Key Benefits of Microsoft Visual Studio Team System.* s.l. : Microsoft, 2007.

16. MSDN Library. *MSDN Library.* [En línea] Microsoft Corporation, 2010. [Citado el: 15 de Enero de 2010.] http://msdn.microsoft.com/es-es/library.

17. SAP BusinessObject. *SAP BusinessObject.* [En línea] [Citado el: 15 de Enero de 2010.] http://www.sap.com/solutions/sapbusinessobjects/sme/reporting-dashboarding/reporting/crystalreports/index.epx.

18. Grape City. *Grape City.* [En línea] [Citado el: 20 de Enero de 2010.] http://www.datadynamics.com/Products/ActiveReports/Overview.aspx.

19. Stimulsoft. *Stimulsoft.* [En línea] 2010. [Citado el: 20 de Enero de 2010.] http://www.stimulsoft.com/es/ReportsNet.aspx.

20. *Manual de Usuario del Sistema Generador de Reportes Dinámico.* Ciudad Habana : s.n., 2009.

21. **Standard, OASIS.** *Reference Model for Service Oriented Architecture 1.0.* s.l. : Adobe Systems Incorporated;MITRE Corporation;Fujitsu Laboratories of America Limited;Booz Allen Hamilton, 2006.

22. **Katrib, Miguel, y otros.** *Visual Studio .Net 2008 desafía todos los retos.* Ciudad Habana, Cuba : Capitán San Luis, 2008.

23. **Identidad-Cuba, Grupo de Desarrollo Proyecto.** *Estándares de codificación.* Ciudad Habana, Cuba : s.n., 2010.

24. **Guerrero, Luis A.** *Arquitectura física.* Santiago, Chile : s.n.

25. *Ingeniería de Software. Un enfoque práctico.*

Glosario de Términos

ADO.NET: es un conjunto de componentes del software que pueden ser usados por los programadores para acceder a datos y a servicios de datos. Es una parte de la biblioteca de clases base que están incluidas en el Microsoft .NET Framework., 34

API (Application Program Interface): conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación., 26

BPMN(Bussiness Process Management Notation): Notación para el Modelado de procesos de Negocio. Permite el modelado de asuntos de negocio donde se presentan gráficamente las diferentes etapas del proceso del mismo. Esta notación coordina la secuencia de eventos que fluyen entre los diferentes procesos y participantes., 26

COM(Common Object Model): es una plataforma de Microsoft para componentes de software utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos., 33

Common Language Runtime(CLR):es el componente de máquina virtual de la plataforma .Net de Microsoft. Es la implementación del estándar Common Language Infrastructure (CLI) que define un ambiente de ejecución para los codigos de los programas., 30

Eclipse: es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar proyectos., 26

Framework: es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto., 29

HTML: es un lenguaje de programación muy sencillo que se utiliza para crear los textos y las páginas web., 31

IEEE(Instituto de Ingenieros Electricistas y Electrónicos): es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros electricistas, ingenieros en electrónica, científicos de la computación, ingenieros en informática, ingenieros en biomédica, ingenieros en telecomunicación e Ingenieros en Mecatrónica, 43

ISO-9001: norma que especifica los requerimientos mínimos de sistemas de gestión de la calidad y requisitos., 44

JScript: es un lenguaje interpretado orientado a las páginas web, con una sintaxis semejante a la del lenguaje Java., 28

OLEDB: es una arquitectura de base de datos de componentes que implanta un acceso eficiente a red e Internet para llegar a muchos tipos de orígenes de datos, entre los que se incluyen datos relacionales, archivos de correo, archivos simples y hojas de cálculo., 37

SQL(Structured Query Language): es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas., 35

UML(Unified Modeling Language): Lenguaje Unificado de Modelado es un lenguaje de modelado de sistemas de software., 26

Visual Basic .NET: es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET., 26

Visual C++ 2008: es un entorno integrado de desarrollo que permite la programación orientada a objetos (POO) conjuntamente con el sistema de desarrollo SDK (también denominado API) de Windows, 28

Windows CE: sistema operativo de Microsoft incrustado modular de tiempo real para dispositivos móviles de 32-bits inteligentes y conectados., 28

Windows Presentation Foundation: es el subsistema de Windows (librerías integradas en el sistema operativo), orientado a unificar los mecanismos de creación y gestión de interfaces de usuario., 32

Workflow: sistemas que, de manera completa, definen, gestionan, controlan y ejecutan flujos de trabajo en el contexto de procesos de negocio, a través de la ejecución de software, cuyo orden de ejecución es controlado por una representación computarizada del proceso de negocio., 23

XML(Extensible Markup Language): Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una manera de definir lenguajes para diferentes necesidades. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para comrtir la información de una manera segura, fiable y fáci, 26