Universidad de las Ciencias Informáticas

Facultad 1

Título: Framework para el desarrollo de aplicaciones web basadas en workflow sobre la plataforma .NET.

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

Autores: Alejandro Flores Pérez

Reisbel Machado Rosabal

Tutores: MSc. Jorge Landrián García

Ing. Reynier Blanco Zambrano

Ciudad de La Habana, Cuba

Junio 2010

"Año 52 de la Revolución"

"El único modo de hacer un gran trabajo es amar lo que haces. Si no lo has encontrado todavía, sigue buscando. No te acomodes. Como con todo lo que es propio del corazón, lo sabrás cuando lo encuentres."

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado: “Framework para el desarrollo de aplicaciones web basadas en workflow sobre la plataforma .NET”, y otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

_____	_____
Firma del autor	Firma del autor
Alejandro Flores Pérez	Reisbel Machado Rosabal
_____	_____
Firma del tutor	Firma del tutor
Ing. Reynier Blanco Zambrano	MSc. Jorge Landrián García

DATOS DE CONTACTO

Tutor: MSc. Jorge Landrián García
Ingeniero en Ciencias Informáticas, UCI 2007, Título de Oro.
Categoría Docente: Instructor.
Correo electrónico: jlandrian@uci.cu
Subdirector de Investigación y postgrado del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas.

Tutor: Ing. Reynier Blanco Zambrano
Ingeniero en Ciencias Informáticas, UCI 2009, Título de Oro.
Categoría Docente: Instructor.
Correo electrónico: rblanco@uci.cu
Arquitecto principal del proyecto Sistema Único de Identificación Nacional perteneciente al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas.

AGRADECIMIENTOS

Alejandro:

A todas personas que de una forma u otra han ayudado a la realización de este trabajo de diploma y mi desempeño en la Universidad.

A la revolución cubana por darme la oportunidad de formarme profesionalmente.

A mis tutores por toda su preocupación y entrega.

A los profesores de la facultad que me formaron durante cinco años, a todos los profesores y compañeros del proyecto Identidad Cuba.

A todos los amigos que he conocido que han estado siempre presentes en las buenas y en las malas.

A toda mi familia, por su confianza y amor.

Reisbel:

A la UCI por haberme dado mucho en todos estos años

A todos los que han contribuido de una forma u otra en mi formación personal y profesional

A mis profesores

A mis tutores

A todos mis amigos

A mis hermanos Alain, Amed, Malena, Vismar y Ale por todo

A mi familia

DEDICATORIA

Alejandro:

A mis padres, Isabel y Alfredo, por todo su esfuerzo porque yo sea alguien en la vida, nadie ha recibido más afecto y amor incondicional como el que ustedes me han dado.

A mi novia por ser mi confidente, mi amiga, por saber comprenderme y apoyarme en todo.

A mi hermano por ayudar a mi mamá durante todos estos años que no he podido estar con ella.

Reisbel:

A mi hermano

A mi papá

Al más chiquito de la familia, mi sobrinito Ernesto

A mi abuela

A la persona que más admiro y quiero en esta vida, mi mamá

RESUMEN

La tecnología de flujo de trabajo (*workflow)* se ha erigido en los últimos años como una herramienta de gran relevancia y eficiencia para llevar a cabo la necesaria coordinación de los elementos que intervienen en los procesos: usuarios, actividades y recursos. Las aplicaciones que utilizan esta tecnología permiten una automatización integral del entorno de los procesos, aportando el dinamismo necesario para gestionar adecuadamente la complejidad y heterogeneidad de los procesos de negocio en las organizaciones. El presente trabajo se propone la implementación de un *framework* en la plataforma .NET que facilite el desarrollo de aplicaciones web basadas en *workflow*. La solución ofrece un conjunto de funcionalidades, proporciona una mayor aproximación a los usuarios de negocio, brinda rapidez y flexibilidad para modelar y cambiar los procesos según las necesidades de las organizaciones, aporta escalabilidad, fortalece el puente creado por el *workflow* para la comunicación entre el analista y el desarrollador, posee actividades y servicios especializados en la orquestación de interfaces de usuario, que permiten definir su flujo de manera gráfica. La solución está desarrollada en un entorno orientado a las tecnologías de *Microsoft* [1]en la versión 3.5 de .NET *Framework*, utilizando Oracle como sistema gestor de bases de datos. Actualmente el *framework* se encuentra en uso por el Proyecto Identidad Cuba perteneciente al Centro de Identificación y Seguridad Digital.

Palabras Claves

desarrollo de aplicaciones web, framework, motor de procesos, workflow

[1] ***Microsoft*:** empresa multinacional estadounidense, fundada en 1975 por Bill Gates y Paul Allen. Dedicada al sector de la informática, con sede en Redmond, Washington, Estados Unidos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con la aparición de las nuevas tecnologías ha evolucionado la forma en que se desarrollan los sistemas de información de las organizaciones. Inicialmente surgieron los sistemas operativos y sus aplicaciones, más tarde los sistemas manejadores de bases de datos, con estos se logró separar los datos de la aplicación. Posteriormente la atención se centró en las interfaces de usuario, donde se buscaba independizar estas de las aplicaciones. Finalmente apareció la tecnología de *workflow*, que busca separar los procesos de negocio de los sistemas.

La tecnología de *workflow* se ha erigido en los últimos años como una herramienta de gran relevancia y eficiencia para llevar a cabo la necesaria coordinación de los elementos que intervienen en los procesos: usuarios, actividades y recursos. Los sistemas de *workflow* permiten una automatización integral del entorno de los procesos, aportando el dinamismo necesario para gestionar adecuadamente la complejidad y heterogeneidad de tales elementos.

La alta competitividad con la que se enfrentan las organizaciones ha hecho indispensable que estas busquen mayor flexibilidad, innovación y rapidez en su toma de decisiones. La gestión de procesos de negocio desempeña un papel crucial en el logro de estos objetivos. La gestión está dirigida a realizar procesos competitivos y capaces de reaccionar autónomamente a los cambios, mediante el control constante de la capacidad de cada uno, la mejora continua, la flexibilidad estructural y la orientación de las actividades hacia la plena satisfacción del cliente y de sus necesidades. Indudablemente este es uno de los mecanismos más efectivos para alcanzar altos niveles de eficiencia, si los procesos son identificados y delimitados adecuadamente, permiten acelerar el desarrollo de los sistemas de información de la organización, además facilitan el mantenimiento de los mismos.

La idea de tener herramientas genéricas para soportar procesos de negocio surgió en los años 70, sin embargo, no tuvieron el éxito deseado debido principalmente a que en esa época no se contaba con la infraestructura adecuada. No fue hasta los años 90 que las redes de datos se volvieron más sólidas. Adicionalmente no se tenía en mente sistemas de información que contemplaran a los procesos de negocio como una parte fundamental del mismo, ni las organizaciones les daban la importancia que merecían. No es hasta la actualidad que se ha alcanzado un punto en el desarrollo de la tecnología orientada a procesos, donde primero se diseña el *workflow* de la manera más abstracta, sin considerar la implementación, y entonces es diseñado el sistema de información junto con los recursos de la organización. Desde ese entonces el término *workflow* ha estado tradicionalmente asociado con la automatización de procesos de negocio, donde documentos, información y tareas son intercambiados

y transferidos entre diferentes participantes, de acuerdo con un conjunto definido de reglas, para conseguir o contribuir a un objetivo de negocio.

Las aplicaciones web tienen una serie de características propicias para el desarrollo de sistemas para organizaciones donde exista un ambiente de trabajo distribuido. Las mismas posibilitan un entorno multiplataforma del lado del cliente, exigiendo el mínimo de requerimientos para su uso, solo es necesario poseer una conexión a internet o a una intranet y un navegador web que cumpla con los estándares establecido. Además, debido a que la aplicación se encuentra alojada en el servidor, resulta menos costosa la actualización del sistema.

Windows Workflow Foundation[2](WWF) es una tecnología extensible, que permite el desarrollo de aplicaciones con procesos de negocio complejos. Sin embargo, en la versión actual de esta tecnología se dificulta el desarrollo de aplicaciones web orientadas a procesos, que permita la navegabilidad a nivel de interfaz, el intercambio de información entre la presentación y el *workflow* y que además soporte conceptos tales como transacciones, concurrencia, tracking, persistencia y comunicación entre servicios, todo lo anterior utilizando múltiples gestores de bases datos.

Al hacer un análisis de la situación planteada se define el siguiente **problema científico**: ¿Cómo facilitar el desarrollo de aplicaciones web basadas en *workflow* sobre la plataforma .NET?

Por tanto, este trabajo se propone como **objeto de estudio** los *frameworks* para el desarrollo de aplicaciones basadas en *workflow*. Como **campo de acción** las aplicaciones web basadas en *workflow* sobre la plataforma .NET.

Con vista a dar solución al problema mencionado anteriormente se plantea como **objetivo general** de la investigación: Desarrollar un *framework* que facilite el desarrollo de aplicaciones web basadas en *workflow* sobre la plataforma .NET. Para cumplir con el mismo se han derivado los siguientes objetivos específicos:

- Definir la arquitectura, servicios, componentes y funcionalidades para el *framework*.
- Desarrollar las funcionalidades del *framework*.
- Evaluar las funcionalidades implementadas en el *framework*.

Para dar solución a la interrogante se plantea la siguiente **hipótesis**:

[2] ***Windows Workflow Foundation*:** Es básicamente una librería constituida por artefactos de ejecución, reglas, actividades, servicios de *runtime* y un diseñador que permite el diseño gráfico de los *workflows*.

Si se implementa un *framework* se facilitará el desarrollo de aplicaciones web basadas en *workflow* sobre la plataforma .NET.

Para dar cumplimiento a los objetivos planteados se definieron las siguientes tareas:

- Estudio de los temas relacionados con el estado actual de las soluciones informáticas que utilizan tecnología *workflow*, así como los *framework* para el desarrollo de aplicaciones web basadas en *workflow* sobre la plataforma .NET.
- Análisis de las tecnologías que permitan desarrollar un *framework* adecuado para el desarrollo de aplicaciones web basadas en *workflow* sobre la plataforma .NET.
- Identificación de los requisitos funcionales y no funcionales de sistema.
- Identificación de la arquitectura, servicios y componentes para el *framework*.
- Implementación del *framework* para el uso del mismo en el desarrollo de aplicaciones web basadas en *workflow* sobre la plataforma .NET.
- Diseño de los casos de pruebas que permitan cuantificar la calidad de cada uno de los componentes y funcionalidades del *framework*.
- Prueba de las funcionalidades implementadas en el *framework*.

Para el cumplimiento de las tareas de investigación se emplearon los siguientes métodos investigativos:

Histórico-Lógico*:* se realizó la investigación con apoyo de este método investigativo debido a que en la actualidad existe un amplio desarrollo de proyectos similares, lo que da la posibilidad de analizar toda una serie de experiencias obtenidas por otros investigadores y ponerlas al servicio del propósito de esta investigación, además se utilizó para el estudio de las tecnologías utilizadas durante el desarrollo de la investigación.

Analítico-Sintético: se utilizó este método para el procesamiento de la información y arribar a las conclusiones de la investigación, así como precisar las características de la solución propuesta.

Modelación*:* se utilizó como herramienta para comprender el problema y crear un modelo abstracto del mismo que permita el diseño de la solución.

Entrevista*:* utilizado con asesores y miembros de grupos de desarrollo sobre diferentes temas relacionados con la investigación.

El **resultado esperado** de la presente investigación es un *framework* que facilite el desarrollo de aplicaciones web basadas en *workflow* sobre la plataforma .NET.

El documento consta de cuatro capítulos, desarrollados a partir del estudio realizado. La descripción de los mismos se presenta a continuación:

Capítulo I. Fundamentación teórica: en este capítulo se realiza un análisis de otras soluciones existentes. Se hace alusión a las herramientas y tecnologías propuestas en el ambiente de desarrollo definido, a los sistemas manejadores de *workflow* y a la tecnología. De esta última se detallan las características y elementos que la componen.

Capítulo II. Características del sistema: en este capítulo se exponen las principales características y requisitos del sistema que son objeto de automatización en el desarrollo de la investigación, se realiza por consiguiente una propuesta general del sistema. Se describe el negocio a través de un modelo de dominio mediante herramientas de modelación. Se analizan cada una de las funcionalidades, y procesos del sistema.

Capítulo III. Análisis y diseño del sistema: en este capítulo se describe a profundidad la construcción de la propuesta de solución mediante los diversos artefactos que especifica el proceso de desarrollo de *software* utilizado. Se realiza un análisis de las características y funcionalidades que se desean desarrollar. Se define el diseño de la base de datos, incluyendo el diagrama de clases persistentes así como el modelo de datos correspondiente.

Capítulo IV. Implementación y pruebas: este capítulo contiene el modelo propuesto para la implementación, los estándares de codificación utilizados, las diferentes interfaces y la implementación de los componentes, servicios y funcionalidades del framework, así como los resultados de las pruebas realizadas al mismo.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se realiza un estudio del marco teórico sobre las soluciones que utilizan la tecnología de *workflow* para el modelado y gestión de procesos, partiendo de los principales conceptos para lograr su entendimiento. Se caracterizan las herramientas y tecnologías que se analizaron para ser utilizadas en la confección de la solución.

1.2 Origen y evolución de la tecnología *workflow*

Esta tecnología no se originó como un método de trabajo en grupo, sino como una forma de reducir el tiempo y costo de llevar a cabo los procesos de negocios y asegurar que las tareas sean llevadas a cabo consistentemente para mejorar la calidad de las mismas. El crecimiento en el uso del *workflow* fue soportado por la introducción de las redes locales y el correo electrónico. Las primeras aproximaciones a la automatización de las tareas de oficina a través del almacenamiento de copias digitales de documentos como cartas de clientes o facturas condujeron al desarrollo de los *workflows*.

La automatización de tareas se basa en la idea de que algunas cosas son realizadas con mayor efectividad por las computadoras que por las personas. Los humanos son buenos para tomar decisiones, innovar, identificar hechos inesperados, etc. Pero usualmente no son eficientes en actividades tales como: buscar un documento entre cientos, tener presentes los vencimientos de las tareas que se tienen que realizar dentro de ciertos plazos, así como el aseguramiento de que el trabajo terminado pase de un lugar a otro respetando una secuencia definida.

Al igual que la evolución de la informática en general, la evolución del *workflow* está ligada con el cambio en los paradigmas informáticos y organizacionales de cada época. Si se resume la evolución de la informática en las últimas cuatro décadas, se verá como han cambiado los objetivos a seguir de cada época. En la década de los 60’ y 70’ el objetivo era resolver grandes cantidades de cálculo de manera eficiente. En los 80’ se buscaba mejorar el manejo y administración de las bases de datos y en los 90’ surge la necesidad de entender y poder manejar eficientemente los procesos, a manera de poder sacarle el mayor provecho posible. Observando la actuación del *workflow* dentro de estas tres etapas, se puede identificar lo que sería un *workflow* manual en la primera etapa, el automatizado dentro de la segunda, y lo que ofrece en la actualidad. En el primer caso se puede apreciar que antes de que la informática se integrara al trabajo cotidiano, éste era realizado manualmente combinando toda la información en distintas ficheros. En este ambiente era bastante difícil determinar el estado de

un determinado fichero, así como el hecho de determinar el proceso a seguir. Se manejaban grandes cantidades de documentos en forma manual, con los consiguientes errores humanos que traían aparejados dichos manejos. Es posible entonces identificar un *workflow* manual inmerso en las tareas cotidianas de esa época. Surge a partir de ese momento la necesidad de remplazar las actividades manuales por actividades automáticas. Se busca tener un mayor control y coordinación sobre toda la información que se maneja para llevar a cabo las tareas de las empresas.

En los comienzos de la década de los 80' se aprecia la existencia de diversos sistemas de información, donde se manejaba y administraba toda la información necesaria para llevar a cabo la producción. Se logró automatizar ciertas tareas, que antes se realizaban manualmente, se puede hablar por lo tanto de un *workflow* automatizado. A fines de esta década se busca mejorar el flujo de la información, el desafío que se plantea es obtener la información rápida y eficientemente. Surgen las necesidades de incrementar la eficiencia, optimizar la productividad, acortar los tiempos de los procesos, tener un control sobre estos, así como reducir los costos y mejorar la gestión, todo esto como consecuencia del incremento de la competitividad y de la exigencia de mejores productos, dentro de un mercado que avanza a gran velocidad.

1.2.1 Ventajas de la tecnología workflow

Se pueden identificar algunas de las ventajas de utilizar esta tecnología:

- La estandarización de los procesos de negocio lleva a tener un mayor conocimiento de los mismos, lo que a su vez conduce a obtener una calidad superior.
- Utilizando esta tecnología es posible monitorear el estado actual de las tareas así como observar la forma en que evolucionan los planes de trabajo realizados. Permite ver cuáles son los embotellamientos dentro del sistema, aquellas tareas o decisiones que están necesitando de tiempo no planificado y se tornan en tareas o decisiones críticas.
- La asignación de tareas se realiza mediante la definición de roles dentro de la empresa, eliminando la tediosa tarea de asignar los trabajos caso por caso.
- Se asegura que los recursos de información (aplicaciones y datos) van a estar disponibles para los trabajadores cuando ellos los requieran.
- Se fomenta pensar los procesos de una manera distinta de la tradicional forma jerárquica que se utiliza para diseñarlos.

1.3 Elementos de la tecnología workflow

A continuación se hace un breve recorrido por los conceptos más importantes de la tecnología de *workflow* , estos sirven para entender el desarrollo de esta tecnología.

Workflow

Workflow se puede definir como uno de los mecanismos usados por los negocios para expresar los procesos como series de actividades auto-contenidas. La organización *Workflow Management Coalition*[3] define a los *workflow* como "*La automatización de un proceso de negocio, total o parcial, en la cual documentos, información o tareas son pasadas de un participante a otro para efectos de su procesamiento, de acuerdo con un conjunto de reglas establecidas.*" (1)

Un *workflow* es generalmente expresado usando máquina de estado finito o diagramas de flujo. El trabajo pasa a través del modelo desde el principio hasta el final, y las actividades pueden ser ejecutadas por personas o por funciones del sistema. Además, provee una forma de describir el orden de ejecución y la dependencia de las relaciones entre las piezas de corta o larga duración.

La tecnología *workflow* está vinculada con la automatización de aquellos procedimientos donde las tareas son ejecutadas entre participantes de acuerdo con un conjunto definido de reglas. En la práctica la mayoría de los *workflow* están organizados dentro del contexto tecnológico para tener soporte de tecnologías de información y computadoras para la automatización de sus procedimientos.

Proceso de negocio

Se puede definir un proceso de negocio como un conjunto de actividades estructurales relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Implica un fuerte énfasis en cómo se ejecuta el trabajo dentro de una organización. Cada proceso de negocio tiene sus entradas, funciones y salidas. Las entradas son requisitos que deben tenerse antes de que una función pueda ser aplicada. Cuando una función es aplicada a las entradas de un método, se tienen ciertas salidas resultantes.

Un proceso de negocio puede ser parte de un proceso mayor que lo abarque o bien puede incluir otros procesos que deban ser incluidos en su función. En este contexto un proceso puede ser visto a varios

[3] ***Workflow Management Coalition*****:** Es la coalición o grupo de trabajo que se encarga de estandarizar los procedimientos para el diseño, desarrollo e implantación de una herramienta de workflow en una organización. Se dedica un apartado completo a ella y el significado de las siglas se puede consultar en el apartado.

niveles. El enlace entre los procesos y la generación de valor lleva a algunas personas a ver los procesos como los flujos de trabajo que efectúan las tareas de una organización (2).
Dentro de estos se pueden definir las siguientes características:

- Ser medidos y están orientados al rendimiento.
- Tienen resultados específicos.
- Entregan resultados a clientes o *stakeholders*[4].
- Responden a alguna acción o evento específico.
- Las actividades deben agregar valor a las entradas del proceso.

De esta forma, un proceso de negocio puede ser definido como una colección de actividades estructurales relacionadas que producen un valor para la organización, para sus inversores o para sus clientes. A su vez estas actividades se definen como la unidad lógica de trabajo, es indivisible y por esta razón debe ser llevada a cabo completamente. Sin embargo, la naturaleza indivisible de la tarea depende del contexto en que se encuentra definida.

Orden

Existen cuatro instrucciones básicas para describir el orden en la ejecución de actividades dentro de un proceso. La más simple es la ejecución secuencial de las tareas, en otras palabras las tareas son llevadas a cabo una detrás de la otra. Esto usualmente significa que hay una dependencia entre ellas. Por ejemplo, el resultado de una de estas tareas es la entrada de la otra.

Si dos tareas tienen que ser realizadas de manera simultánea, se está en presencia de un tipo de orden paralelo. En este caso, hay dos tareas que necesitan ser llevadas a cabo sin que el resultado de una afecte al resultado de la otra, tienen que ser ejecutadas simultáneamente.

Otra forma de orden es el selectivo, y se presenta cuando se tiene que escoger entre dos o más tareas. En una situación ideal una tarea no se ejecuta más de una vez en un mismo proceso. Sin embargo, algunas veces es necesario ejecutar una tarea muchas veces, a esta forma de orden se le llama iteración (2).

Agente

Los agentes en el contexto de la tecnología de *workflow* se pueden definir como entidades ya sean humanas o computacionales que llevan a cabo las actividades de un proceso. De acuerdo con la forma

[4] **Stakeholder:** persona que es afectada por las actividades de una organización

en la que se definen los procesos y las actividades, los agentes pueden contar con ciertas características que permiten al comportamiento ir escogiendo uno a uno para ir ejecutando cada actividad.

Modelo de comportamiento

El modelo de comportamiento describe los pasos que se siguen durante la ejecución del *workflow* y que caracterizan su comportamiento, incluye el momento de ejecución de cada una de las actividades, las acciones que se deben realizar en caso de error y el criterio para la elección de agentes. De esta forma, una tarea puede ser ejecutada de cuatro formas básicas (3):

- Manual: El *workflow* es activado mediante la acción de un agente humano.Tiene como parámetro opcional el agente que lo activa.
- Temporal: El *workflow* se activa bajo una condición temporal, como parámetro se puede indicar en qué condición temporal se ejecuta el *workflow*.
- Sistema: Es la ejecución automática del *workfow*. Tiene como parámetro el sistema que ha activado el *workflow*.
- Evento: El *workflow* se activa mediante un evento externo como un mensaje.

Estándares de *workflow*

La diversidad de técnicas para modelar procesos de negocio es un problema notorio en la actualidad. Aunque la estandarización se ha discutido por más de 10 años, sigue haciendo falta un formato de intercambio común. La razón es un problema, que puede ser atribuido a las diferentes perspectivas que tienen los analistas y diseñadores de procesos de negocio. Actualmente, el modelado de procesos de negocio está siendo sujeto de varios esfuerzos de estandarización. Diferentes organizaciones incluyendo el Grupo para el Manejo de Objetos (OMG) (4), la Organización para el Avance de Estándares de Información Estructurados (OASIS) (5) y la Coalición para el Manejo de *Workflows* (*WfMC*) (1), así como compañías de *software* y grupos de académicos han propuesto meta-modelos y formatos de intercambio para el modelado de procesos de negocio.

Modelo de referencia de la Coalición de Manejadores de *Workflow* (*WfMC*)

El modelo de referencia de *workflow* fue desarrollado desde estructuras génericas de aplicaciones de *workflow*, identificando las interfaces con estas estructuras, las cuales permiten a los productos

comunicarse a distintos niveles. Todos los sistemas de *workflow* contienen componentes genéricos que interactúan de forma definida. Para poder tener cierto nivel de interoperabilidad[5] entre los diversos productos de *workflow*, es necesario definir un conjunto de interfaces y formatos para el intercambio de datos entre dichos componentes. El modelo propuesto por la *WfMC* incluye 5 interfaces. Estas interfaces cubren (1):

- Especificaciones para los datos de la definición del proceso y su intercambio.
- Interfaz para soportar interoperabilidad entre diferentes sistemas de *workflow* .
- Interfaz para soportar la interacción con una variedad de aplicaciones.
- Interfaz para soportar las funciones de interfaz con el usuario.
- Interfaz para proveer de monitoreo del sistema y funciones de medición para facilitar el manejo de ambientes mixtos de *workflow* .

La Figura 1. Modelo de referencia de workflow (Interfaces y Componentes). ilustra los componentes principales e interfaces dentro de la arquitectura *workflow* propuesto por la WfWC.

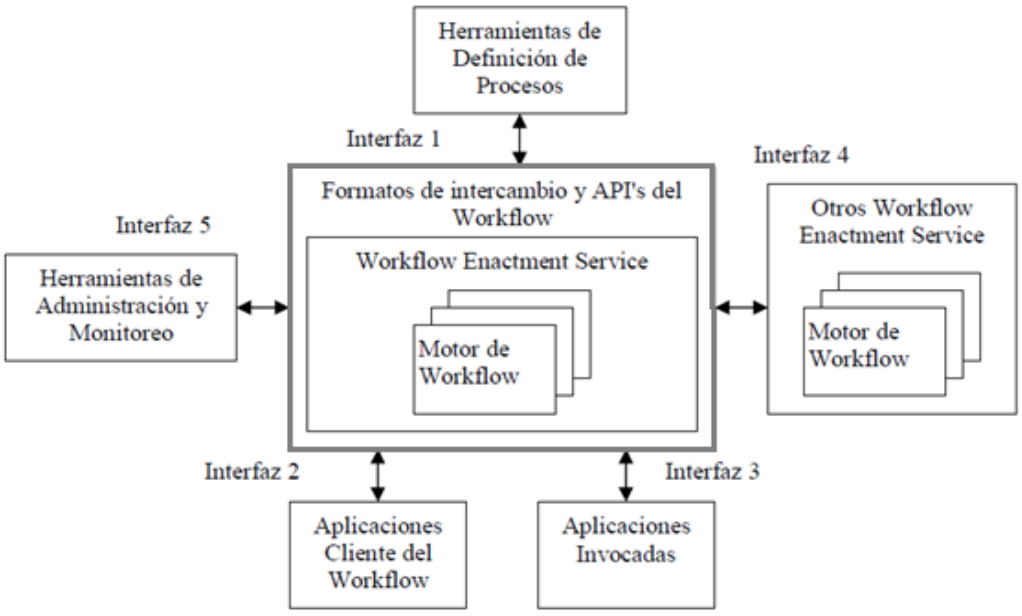


Figura 1. Modelo de referencia de workflow (Interfaces y Componentes). **(1)**

[5] **Interoperabilidad:** en este contexto se entiende como la cooperación de sistemas diferentes que colaboran para llevar a cabo un objetivo.

Estructura de los sistemas manejadores de workflow

Un sistema manejador de *workflow* (*WFMS*) es una parte fundamental en el desarrollo de la tecnología *workflow*. Es una herramienta que permite la definición, la instanciación y ejecución de *workflow* a través de *software* (1). Estos sistemas integran y coordinan aplicaciones heterogéneas distribuidas que colaboran entre sí para llevar a cabo procesos. Permiten dar soporte y agilizar procesos de negocio complejos independientemente del tiempo y lugar. Se puede resumir que los sistemas de *workflow* cubren todos los aspectos del proceso, actividades, información y organización.

En general todos los sistemas de *workflow* deben tener características específicas para proveer soporte en tres áreas funcionales:

- Funciones de construcción
- Funciones de control del proceso en tiempo de ejecución
- Interacción en tiempo de ejecución

En la Figura 2. Características de un sistema de workflow. se muestran las características de estos sistemas.

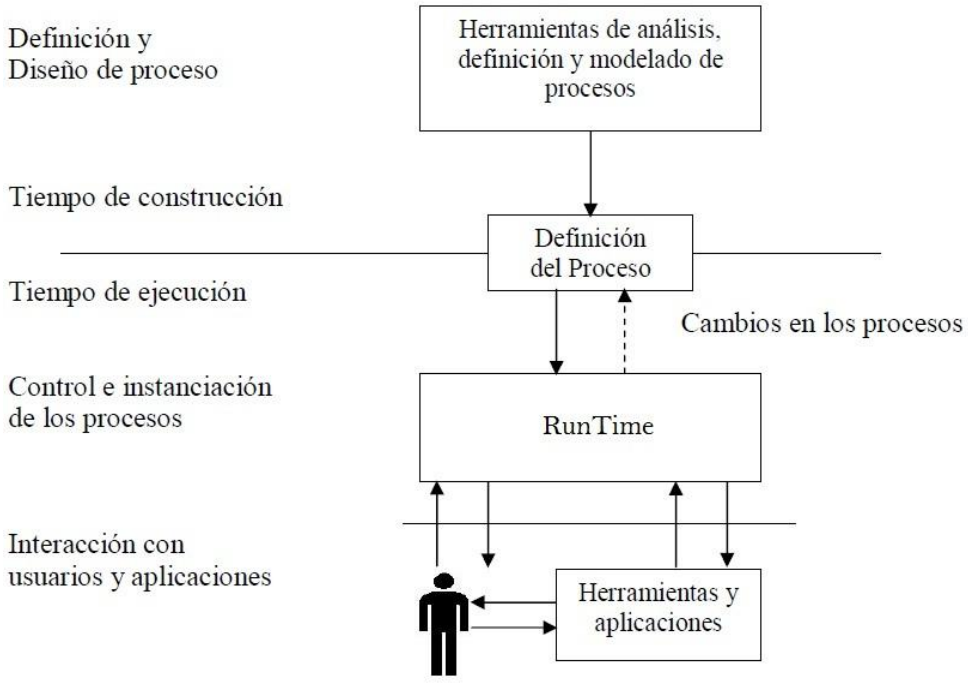


Figura 2. Características de un sistema de workflow. **(2)**

Los sistemas de *workflow* son frecuentemente asociados con la reingeniería de procesos. La misma contempla el análisis, modelado, definición y posteriormente la implementación operacional de los procesos de negocio. Sin embargo, no todas las actividades de la reingeniería de procesos terminan

en una implementación de *workflow*. La tecnología de *workflow* es frecuentemente una solución apropiada que proporciona la separación de la lógica de los procesos de la empresa y su soporte operacional de la tecnología de información, permitiendo que los cambios subsecuentes, sean incorporados dentro de la definición de las reglas definidas en los procesos. Un sistema manejador de *workflow* es aquel que provee la automatización de un proceso de negocio a través del manejo de las actividades subsecuentes y la invocación de un recurso apropiado (humano o tecnológico), asociado a cada actividad del proceso.

Funciones de construcción

Las funciones de construcción corresponden a la definición de un proceso de negocio. El proceso se traslada del mundo real a una definición computacional a través de un *workflow*, procesable mediante una o más técnicas de modelado. La definición de un proceso está asociada a operaciones realizadas por humanos o por computadoras. Esta se puede representar gráficamente o mediante un lenguaje semiformal. Algunos sistemas lo hacen mediante una interfaz gráfica otros prefieren definir los procesos a través de documentos XML (*Extensible Markup Language*)[6]. Dentro de estos sistemas pueden existir alteraciones dinámicas a la definición del proceso durante el tiempo de ejecución.

Funciones de ejecución

En tiempo de ejecución, la definición de un proceso se interpreta por el *software* responsable de crear y controlar la operación de las instancias de los procesos, la secuencia de las actividades dentro de los procesos e invocar los recursos. Estas funciones de control de procesos en tiempo ejecución actúan como vínculo entre el modelado del proceso con el mundo real, reflejado en las interacciones de los usuarios y las aplicaciones. El principal componente es el *software* de control (motor de *workflow*), responsable de la creación y eliminación de procesos, del control de la secuencia de actividades dentro de un proceso y de la interacción con los recursos humanos y/o tecnológicos.

Un motor de proceso *workflow* puede controlar la ejecución de un conjunto de procesos, subprocesos o instancias con un alcance definido determinado por el rango de tipos de objetos y sus atributos, el cual puede ser interpretado dentro de la definición del proceso. El motor de *workflow* es el responsable del control del ambiente de tiempo de ejecución en un servicio de lanzamiento de workflow (*Workflow Enactment Service)*.

[6] **XML:** es un metalenguaje extensible de etiquetas.

El servicio de lanzamiento de *workflow* consiste de múltiples motores de *worklfow*. Hay una partición de la ejecución del proceso a través de los motores que lo forman. Este puede ser por tipo de proceso, con un motor particular controlando un tipo particular de proceso en su totalidad; por la distribución de funcionalidad, con un motor particular controlando aquellas partes de un proceso que requieran la intervención del usuario o del uso de recurso dentro de su control de dominio, o algún otro mecanismo de partición.

Facilidades que provee el motor de *workflow*.

- Interpretación de la definición de procesos.
- Control de las instancias de los procesos: creación, activación, suspensión, terminación.
- Navegación entre actividades, las cuales pueden invocar operaciones secuenciales o paralelas, calendarización de plazos, interpretación de datos relevantes.
- Conexión o desconexión de participantes específicos.
- Identificación de actividades para la atención de algún usuario y una interfaz para soportar la interacción con el usuario.
- Mantenimiento de los datos de control y datos relevantes del workflow, así como del intercambio de datos relevantes del workflow hacia o desde aplicaciones o usuarios.
- Una interfaz para invocar aplicaciones externas y el vínculo con algún dato relevante del *workflow*.
- Acciones de supervisión para el control y administración.

Funciones de interacción

El proceso representado por un *workflow* normalmente está relacionado a operaciones humanas que se realizan en conjunto con herramientas computacionales. La interacción de estas herramientas con el *software* de ejecución es fundamental para poder llevar a cabo el proceso.

Patrones de *workflow*

En el desarrollo de *software*, los patrones son descritos como una solución bien definida y entendida a un tipo particular de problema. Los patrones ofrecen mayor flexibilidad a los programas. Similar a los patrones de programación se pueden identificar algunos patrones de *workflow*, mediante los cuales se ha acelerado la estandarización de procesos en estos sistemas.

Desde 1999 se han venido elaborando patrones de *workflow* . Los patrones van desde los más simples como el patrón secuencial hasta los complejos, por ejemplo, el patrón de sincronización. Los patrones de *workflow* pueden ser clasificados en las siguientes categorías (6):

- **Patrones de control de flujos básicos.** Estos patrones están presentes en la mayoría de los lenguajes de *workflow*, y sirven para modelar procesos secuenciales, paralelos y aquellos que incluyan alguna decisión.
- **Patrones de ramificación avanzada y sincronización.** Estos patrones superan a los patrones de control de flujo básico al permitir tipos avanzados de bifurcación y sincronización.
- **Patrones estructurales.** Estos patrones permiten terminar un subproceso cuando ya no haya nada que hacer, o permiten definir ciclos de forma arbitraria.
- **Patrones que manejan múltiples instancias.** Cuando se le da seguimiento a un caso, algunas veces es necesario que el proceso sea instanciado muchas veces.

Tipos de sistemas de workflow

Existen diferentes clasificaciones para aplicaciones de *workflow* en dependencia del objetivo para el que fueron desarrolladas:

- *Workflow* de producción
- Workflow colaborativo
- Workflow administrativo

***Workflow* de producción**

En las aplicaciones de *workflow* de producción, el mismo es la tarea principal de los participantes. Dicho personal puede tener actividades adicionales en su trabajo diario, pero fundamentalmente la realización de *workflow*. Debe notarse además que el *workflow* de producción se suele circunscribir a un sólo departamento de la organización. Debido a la naturaleza de producción, dichas aplicaciones deben cumplir con algunos de los siguientes atributos (2):

- Velocidad de transferencia, o sea, la velocidad con que las tareas pasan de un paso a otro. Es improductivo que un miembro del equipo no haga nada mientras espera a que le llegue trabajo.
- La flexibilidad de poder cambiar el proceso no suele ser importante. Una vez establecido el flujo, este permanece sin cambio por largo tiempo. El *workflow* de producción suele estar

circunscrito a un sólo departamento, la escalabilidad[7], o capacidad de crecer no es importante. Este tipo de soluciones está optimizado para trasladar grandes volúmenes de información e imágenes a lo largo de rutas preestablecidas.

El *workflow* de producción fue el primer tipo de *workflow* desarrollado y mercadeado, porque generalmente no se requería de una base distribuida de usuarios a lo largo de la compañía para lo que es indispensable contar con una red de área local. En general automatizan procesos de negocios que tienden a ser repetitivos, bien estructurados y con gran manejo de datos.

***Workflow* colaborativo**

Involucra procesos estructurados o semi-estructurados que permiten a varias personas participar en un grupo de trabajo. Estos procesos involucran típicamente un documento que hace las veces de contenedor de la información, viajando de paso en paso y en cada uno de ellos el partícipe realiza una tarea o acción sobre el documento. Por tanto, las características esenciales de workflow colaborativo son las siguientes (2):

- El documento y el proceso son claves. Es importante para la aplicación preservar la integridad tanto del documento como del proceso. Está restringido a ciertos grupos creativos dentro de la organización. Es importante que una buena solución no sea intrusiva ya que el trabajo de conocimiento es un proceso mental que involucra la creatividad, la que no se desea restringir.
- El *workflow* colaborativo debe ser muy flexible ya que el trabajo creativo puede tomar rumbos inesperados. Las soluciones de *workflow* colaborativo suelen estar centradas en el documento.

***Workflow* administrativo**

Involucra procesos administrativos tales como órdenes de compra, hojas de tiempos y movimientos, reportes de gastos, cambios de órdenes, reportes de calidad y muchas otras actividades que traspasan las barreras departamentales e inclusive de la empresa misma.

Existen un gran número de procesos administrativos en cada organización, por ello la solución debe ser capaz de manejar muchos procesos diferentes. Casi cualquier persona es un participante potencial, de ahí que la escalabilidad de la solución sea de mucha importancia. (2)

[7] **Escalabilidad:** propiedad deseable de un sistema que indica su habilidad para extender el margen de operaciones sin perder calidad.

El *workflow* administrativo es diferente para cada organización; de ahí la importancia de poder cambiar los procesos fácilmente. Ya que cualquiera en la empresa es un participante potencial, es necesario poder distribuir el *software* al mayor número de usuarios con la menor carga logística posible.

Perspectiva de workflow

Los procesos de *workflow* son manejados por casos, es decir, las tareas son ejecutadas para casos específicos, los cuales son usualmente soportados por sistemas manejadores de *workflow* .

En los procesos manejados por casos, se pueden distinguir tres perspectivas de modelado: la perspectiva de control del flujo del proceso, la perspectiva de recursos del proceso, y la perspectiva del caso. La perspectiva del control del flujo del proceso se refiere al orden en que se deben ejecutar las tareas de un proceso. Las tareas que necesitan ser ejecutadas son determinadas por estructuras de distribución especificadas en el proceso. Las estructuras de distribución comunes son: la secuencia, la condición, el paralelismo y la iteración. Las tareas de un proceso son además ejecutadas por recursos, estos pueden ser humanos o técnicos.

En la perspectiva de recursos, estos son agrupados en roles. Estas dos perspectivas, la perspectiva de control del flujo del proceso y la de recurso, son genéricas, es decir, no están ligadas a un caso específico. La tercera perspectiva se refiere a los casos individuales que son ejecutados de acuerdo con el proceso definido (primera perspectiva) por los recursos (segunda perspectiva) (3).

Generalidades

Uno de los aspectos que se deben precisar es que la reingeniería[8] y automatización de procesos de negocio no es la misma acción. La automatización de *workflow* es puramente una tecnología de *software* que provee los medios para automatizar procesos de negocio o administrativos. Reingeniería es el análisis de procesos de una organización con la finalidad de mejorarlos de alguna manera.

Las organizaciones pueden automatizar sus procesos de negocio o administrativos, utilizando *software* de automatización sin reingeniería. A su vez, se puede hacer reingeniería sin realizar automatización. Se puede hacer reingeniería y como parte de la actividad instalar una solución de *workflow*. La automatización de *workflow* puede beneficiarse de un proceso de reingeniería y viceversa, pero no hay obligación alguna de acoplar dichos esfuerzos o igualarlos (7).

[8] **Reingeniería:** en este contexto se entiende la re-concepción y el rediseño de los procesos de negocios

Workflow no es lo mismo que automatización de *workflow*. Cualquier aplicación que puede enviar un documento utilizando correo electrónico u otro medio puede reclamar que soporta *workflow*, porque la aplicación en efecto hace fluir trabajo de un individuo a otro. Sin embargo, poder enviar un documento está muy alejado de la automatización. Se debe diferenciar entre habilitado para *workflow* y *software* para la automatización de *workflow*. El *software* de automatización debe proveer un número de características y capacidades para automatizar procesos que van más allá de simple enrutamiento de documentos.

1.4 Análisis de soluciones existentes

La tecnología *workflow* ha sido el foco de una intensa actividad en término de productos, estándares y trabajos de investigación. Actualmente existen compañías y grupos de investigación que desarrollan productos y realizan investigaciones relacionadas con los *workflows*.

Entre los sistemas manejadores de *workflow* de código abierto, se encuentran dos que merecen más atención debido a que, el *Yet Another Workflow Language* (YAWL) se inició y continúa como proyecto de investigación sobre patrones de *workflow* y el segundo, *Java Business Process Management* 2.0 (jBPM) es un sistema flexible y extensible programado en Java, que forma parte importante de JBoss, el servidor de aplicaciones de *software* libre más utilizado para aplicaciones J2EE (*Edition Enterprise Java 2*) y por ende su distribución e integración es transparente.

Yet Another Workflow Languaje (YAWL)

El *YAWL* surge en respuesta a las limitaciones de los sistemas manejadores de *workflow* y lenguajes de *workflow* para modelar patrones de *workflow* existentes. Está basado en Redes de *Petri*[9], pero extiende algunas características para facilitar la implementación de patrones que involucran múltiples instancias, sincronización avanzada y cancelación. *YAWL* es de hecho, un nuevo lenguaje con su propia semántica y específicamente diseñado para *workflow* (8).

La implementación del lenguaje se encuentra limitada en algunos aspectos, ya que la ejecución de Redes de *Petri* de alto nivel es complicada debido su complejidad. Además, a pesar de ser un modelo altamente expresivo, su mayor desventaja es que su definición gráfica está basada en Redes de Petri que tienen menos legibilidad. Otra desventaja significativa es que no genera trazas suficientemente potentes.

[9] **Redes Petri:** permiten expresar eventos concurrentes en sistema de evolución en paralelo.

Java Business Process Management (jBPM)

El objetivo de jBPM es proveer un sistema manejador de *workflow* en *Java* que sea flexible y extensible. Está ideado para proveer de un mecanismo muy simple, para empezar con una máquina de estados, que permite a los desarrolladores en Java agregar jBPM en sus proyectos. Por otro lado, permite escalar a los procesos de *workflow* más complejos y patrones de *workflow*. En octubre de 2004 el proyecto jBPM unió sus esfuerzos con JBoss para que fuera una pieza crítica de su plataforma de *middleware*[10] empresarial.

En la documentación de jBPM el concepto de actividad es remplazado por un estado y una acción. Un estado es un proceso que especifica una dependencia con un actor externo. En tiempo de ejecución del proceso, significa que el motor de *workflow* tiene que esperar hasta que el actor notifique al sistema manejador de *workflow* que el estado ha terminado.

Una acción es una pieza de código que debe ser ejecutada por el sistema manejador de *workflow* a través de un evento especificado que ocurre durante la ejecución del proceso. El sistema manejador de *workflow* inicia la ejecución de la acción en un evento especificado durante la ejecución del proceso.

El modelo de estados de jBPM está basado en grafos con nodos y transiciones. Un estado es un ejemplo de un nodo. El grafo de estado, provee la estructura del proceso. Las acciones son piezas de código que pueden ser ejecutadas por eventos en los procesos. (9)

Dentro de las desventajas que se pueden identificar se encuentra que la documentación no se actualiza a la par con la liberación de cada versión. Además, el diseño gráfico no se asemeja tanto a un diagrama de actividades, sino que tiende más a parecerse a un diagrama de grafos. Otra limitante que presenta es que está orientada a la programación más que al diseño gráfico y que la definición del flujo en XML es muy sencilla pero no es estándar.

Windows Workflow Foundation

Windows Workflow Foundation es una tecnología extensible para desarrollar soluciones de *workflow* sobre la plataforma .NET. Provee una Interfaz de programación de aplicaciones (API)[11] y herramientas para el desarrollo y la ejecución de aplicaciones basadas en *workflow*.

[10] **Middleware:** software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

[11] **API**: conjunto de funciones residentes en sistema operativo en que permiten que una aplicación se ejecute.

Esta tecnología complementa al *framework* .NET con un grupo de componentes basados en *workflow* que brindan a los desarrolladores la habilidad de definirlos, compilarlos, instanciarlos, depurarlos y rastrearlos. Los desarrolladores pueden incorporar conceptos tales como planificación, coordinación de tareas y escalabilidad en sus aplicaciones existentes con mínimo costo. WWF provee la plataforma base donde se pueden desarrollar aplicaciones con disímiles procesos (13).

El principal beneficio de este *framework* es que trae un modelo unificado de *workflow* y un conjunto de herramientas en reemplazo de muchas librerías propietarias. Usándolo es posible el desarrollo de aplicaciones con procesos de negocio complejos que soporten: transacciones, concurrencia, compensación, *tracking* y comunicaciones.

Dentro de las principales desventajas que esta tecnología presenta se puede citar que el desarrollo con la misma tiene una curva de aprendizaje alta, además tiene complicados pasos de configuración, especialmente en lo que respecta a la implementación. Otra limitante es que se debe desarrollar la tecnología en torno a los objetos en tiempo de ejecución a fin de ejecutar varias instancias al mismo tiempo. Por otra, parte cualquier cambio implica una modificación en el *workflow*, por lo que incluso los *workflows* mas sencillos llevan cierto tiempo para su desarrollo. Es importante citar que solamente tiene soporte para el sistema gestor de base de datos *SQL Server*, aunque brinda la estructura necesaria para posibles implementaciones personalizadas. Además, en el desarrollo de aplicaciones web no provee una estructura para la definición de las interfaces desde el propio diseño del *workflow*.

Luego de analizar las características, ventajas y desventajas de las soluciones existentes más importantes se decide utilizar WWF como base para el desarrollo de la solución, por las posibilidades que tiene esta tecnología de poder extender sus funcionalidades y por los beneficios que presenta. Además, la misma cumple con las políticas de desarrollo del Proyecto Identidad Cuba, entorno donde se encuentra enmarcada la investigación.

1.5 Ambiente de desarrollo

El ambiente de desarrollo es algo imprescindible en la producción de *software*. Es donde se definen el conjunto de herramientas y tecnologías que intervienen en un proceso de desarrollo de *software*, así como versiones a usar y su integración.

A continuación se presentan un conjunto de herramientas utilizadas en el desarrollo de este trabajo como son: un ambiente de desarrollo integrado, un control de versiones, un gestor de base de datos, herramientas de modelado y *frameworks*. Se exponen sus características y ventajas para poder tener

un mayor conocimiento de sus prestaciones y valorar su utilidad. Es preciso aclaran que el uso de dichas herramientas es dispuesto por las políticas de desarrollo que se establecen en el Proyecto Identidad Cuba.

1.5.1 Ambiente de desarrollo integrado

Microsoft Visual Studio Team System 2008 Service Pack 1

El Visual Studio 2008 es un entorno de desarrollo que ofrece un conjunto de herramientas avanzadas y características de alta productividad, tales como: diseñadores visuales, funciones de *debugging*[12], funciones para bases de datos y funciones innovadoras que permiten crear rápidamente aplicaciones futuras para distintas plataformas. Esta última versión incluye numerosas mejoras como diseñadores visuales para un desarrollo más rápido con *.NET Framework 3.5*, mejoras sustanciales en las herramientas de desarrollo web y mejoras de programación que aceleran el desarrollo a partir de todo tipo de dato.

Brinda funciones de programación y de datos mejoradas, como *Language Integrated Query*[13] (LINQ), que facilita el armado de soluciones capaces de analizar información y de actuar en consecuencia. Presenta características de diseño de arquitecturas, un explorador de servidores para obtener acceso visual a bases de datos, contadores de rendimiento y componentes de aplicaciones del lado del servidor (13).

1.5.2 Sistema gestor de bases de datos

Oracle 11g

Oracle es un sistema de gestión de base datos relacional desarrollado por *Oracle Corporation*. Se considera a como uno de los sistemas de bases de datos más completos, destacándose por las siguientes características (14):

- Soporte de transacciones.
- Estabilidad.

[12] **Debugging:** es el proceso de identificar y corregir errores de programación.

[13] **LINQ:** conjunto de características que agregan capacidades de consulta a los lenguajes C# y Visual Basic.

- Escalabilidad.
- Soporte multiplataforma.

La base de datos Oracle en *Windows* ha evolucionado desde un nivel básico de integración del sistema operativo hasta utilizar servicios más avanzados en esta plataforma.

Brinda una plataforma segura y escalable para el acceso confiable y rápido a todo tipo de información utilizando las interfaces estándar del sector. Permite una sólida administración del contenido para tipos avanzados de datos, esta versión incluye nuevas mejoras y características como son (15):

- La recopilación automática para PL/SQL y Java en la base de datos.
- *Triggers*[14] más rápidos, con inclusión de invocaciones más eficientes de *triggers* por fila.
- Operaciones SQL más rápidas y fáciles de usar.
- Replicación *Oracle Data Guard* y *Oracle Streams* más rápida.
- Conexiones directas, más rápidas y confiables, a los dispositivos de almacenamiento del sistema de archivo de red.
- Actualizaciones más rápidas.
- Recuperación más rápida de archivos de gran tamaño.

1.5.3 Herramientas de modelado

Altova UModel 2009 Services Pack 1 Enterprise Architects: Es una herramienta de modelado, se encuentra entre las más simples herramientas de dibujo que no necesitan mayor comprensión para estructurar diagramas y las más complejas herramientas de apoyo que poseen hasta el último matiz de especificaciones. Sin embargo, no implementa todos los tipos de diagramas definidos en la especificación de UML 2.1, en cambio, permite construir nueve de los más comunes y útiles tipos de diagramas UML.

Los diagramas son organizados en una jerarquía de paquetes y se pueden manejar de dos formas, ya sea en la vista de modelo (que muestra la jerarquía entera), o en la vista de diagrama (donde son

[14] **Trigger**: procedimiento que se ejecuta cuando se cumple una condición al realizar una operación en base de datos.

agrupados por tipo de diagrama). También se pueden compartir proyectos entre uno o más de UModel. (16).

Se puede utilizar para generar código en Java o C#; o bien, importar clases de Java o C# para construir los diagramas UML correspondientes. La generación de código está basada en plantillas, que son totalmente modificables y así permite un alto nivel de control sobre el código generado. En general, es una manera muy servicial de generar desde un simple diagrama UML hasta una aplicación muy personalizada junto con las plantillas de código en C# y Java.

1.5.4 Tecnología

***Oracle Data Provider* para .NET 10.2.0.2**

Oracle Data Provider ofrece un mejor acceso a la información de la base de datos de Oracle desde un entorno .NET. Permite a los desarrolladores aprovechar la funcionalidad avanzadas de la base de datos de Oracle, las optimizaciones de desempeño y las características avanzadas de seguridad. Ofrece también un mejor ambiente, flexibilidad y alternativas para sus aplicaciones. Con ella, los desarrolladores pueden utilizar .NET, pero sin sacrificar las capacidades de administración de datos que Oracle brinda (17).

***Microsoft .NET Framework* 3.5**

NET Framework 3.5 contiene muchas y nuevas características que se agregan a las versiones anteriores del *framework*. Algunas de estas características son (13):

- Integración total de LINQ y del reconocimiento de los datos. Esta nueva característica permite escribir código en idiomas habilitados para LINQ para filtrar, enumerar y crear proyecciones de varios tipos de datos SQL, colecciones, XML y conjuntos de datos usando la misma sintaxis.
- ASP.NET incorpora AJAX que permite crear experiencias web más eficaces, más interactivas y con un gran índice de personalización que funcionan con los exploradores más usados.

Es neutral en cuanto al lenguaje de programación y funciona sobre la base de librerías. Se puede programar utilizando *Visual Basic.NET*, C++.NET, C#, J# y otros más. Admite la creación y ejecución de la siguiente generación de aplicaciones y servicios web XML. El diseño está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.

- 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, pero distribuida en Internet o ejecutar de forma remota.
- 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 escritorio 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.

Contiene dos componentes principales: *Common Language Runtime* (CLR) y la biblioteca de clases de *.NET Framework*. CLR es el fundamento de la tecnología. El motor de tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. El código destinado al motor de tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado. La biblioteca de clases es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario.

Lenguaje de programación C# 3.0

C# un lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que carece de elementos heredados innecesariamente en .NET. Por esta razón, se suele decir que es el lenguaje nativo de .NET.

La sintaxis y estructura es muy parecida al lenguaje de programación C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su

aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son comparables con los de *Visual Basic* (13).

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues *Microsoft* ha escrito la mayor parte de su biblioteca de clases base (BCL en inglés) usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el *.NET Framework SDK.*

***Extensible Markup Language* (XML)**

Lenguaje de marcado extensible (*XML*) es un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. Es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructuradas. Se pueden definir algunas características:

- Es una arquitectura abierta y extensible. No se necesita versiones para que puedan funcionar en futuros navegadores. Los identificadores pueden crearse de manera simple y ser adaptados en el acto en internet/intranet por medio de un validador de documentos.
- Mayor consistencia, homogeneidad y amplitud de los identificadores descriptivos del documento con XML.
- Integración de los datos de las fuentes más dispares. Permite hacer el intercambio de documentos entre las aplicaciones tanto en el propio PC como en una red local o extensa.
- Datos compuestos de múltiples aplicaciones. La extensibilidad y flexibilidad de este lenguaje permite agrupar una amplia variedad de aplicaciones, desde páginas web hasta bases de datos.
- Gestión y manipulación de los datos desde el propio cliente web.

Lenguaje de programación PL/SQL

PL/SQL es un lenguaje de programación incorporado en Oracle que permite el manejo de variables, el uso de estructuras modulares, soporta estructuras de control de flujo y toma de decisiones y controla excepciones.

Este lenguaje está basado en ADA[15], por lo que incluye todas las características de los lenguajes de tercera generación. Esto permite manejar las variables, tener una estructura modular (procedimientos y funciones) y controlar las excepciones. Además, incorpora un completo soporte para la programación orientada a objetos. (19)

Los programas creados con PL/SQL pueden ser almacenados en bases de datos como cualquier otro objeto quedando disponibles para los usuarios. El uso del lenguaje PL/SQL es imprescindible para construir disparadores de bases de datos.

PL/SQL está incluido en el servidor y en algunas herramientas de cliente. Soporta todos los comandos de consulta y manipulación de datos, aportando al lenguaje SQL las estructuras de control y otros elementos propios de los lenguajes de programación de tercera generación.

1.5.5 Control de versiones

Microsoft Visual Studio Team System 2008 Team Explorer

La herramienta *Team Explorer* se puede usar como cliente completo e independiente para obtener acceso a *Visual Studio Team System 2008 Team Foundation Server*. Permite a los usuarios unir el ciclo de vida de desarrollo de *software* con compatibilidad integrada para control de código fuente y de elementos de trabajo, administración de compilaciones e informes detallados que muestran tendencias de errores, velocidad del proyecto e indicadores de calidad (13).

1.5.6 Metodología de desarrollo de software

***Microsoft Solution Framework[16] for Capability Maturity Model Integration[17]* (MSF for CMMI)**

MSF for CMMI es una metodología para el desarrollo de *software* para la planificación, desarrollo y gestión de proyectos tecnológicos. Se centra en el modelo de procesos y de equipo dejando los demás aspectos en segundo plano. Se compone de varios modelos que se encargan de cada una de las fases del desarrollo de un proyecto: modelo de arquitectura del proyecto, modelo de equipo, modelo de

[15] **Ada:** lenguaje de programación orientado a objetos, fuertemente tipado y concurrente.

[16] **Microsoft Solution Framework:** es un conjunto de principios, modelos, disciplinas, conceptos y directrices para el desarrollo de soluciones informáticas de Microsoft.

[17] ***Capability Maturity Model Integration:*** modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

procesos, modelo de gestión de riesgo, modelo de diseño de procesos y modelo de aplicación. Esta metodología no es rígida pues no existe una sola estructura que se pueda acoplar a todos los tipos de proyectos (13).

Es una metodología integrada y productiva, ya que combina muchos elementos y características. Se enfoca más en las habilidades y cualidades de las personas que en la eficacia de los modelos de procesos. *MSF for CMMI* está basado en mejores prácticas del mundo real. El mismo contiene un meta-modelo que está basado en la filosofía de que un proceso está creado para ayudar y no para estorbar, este modelo es compartido para procesos ágiles y maduros. Consta de cinco fases: Inicio, Planificación, Desarrollo, Estabilización e Implementación y es continuamente refinado por clientes, consultores y grupos de desarrollo de *Microsoft*.

1.6 Conclusiones

- El estudio del estado del arte permite afirmar que la tecnología de automatización de *workflow* constituye una herramienta de gran relevancia para llevar a cabo la necesaria coordinación de los elementos que intervienen en los procesos.
- El análisis de las diferentes soluciones existentes demostró la necesidad de comenzar el desarrollo de un *framework* capaz de resolver una serie de limitantes en el desarrollo de aplicaciones web orientadas a procesos.
- Fue seleccionada la tecnología *Windows Workflow Foundation* como base para el desarrollo de la solución, por los beneficios que la misma presenta.
- El estudio de las tecnologías permitió escoger las herramientas, la metodología y los lenguajes más adecuados para el desarrollo de la solución.

CAPÍTULO II. CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se especifica con detenimiento el problema que se pretende resolver, lo que constituye la base para la identificación del objeto de automatización. Se realiza una propuesta inicial del sistema, estableciendo y describiendo el ambiente en el que se enmarca el mismo, así como los requisitos funcionales y no funcionales.

2.2 Modelo de dominio

No se lograron identificar procesos de negocios claros en el marco de la investigación, solo elementos conceptuales; por lo que se propone un modelo de dominio. Con este modelo se pretende contribuir a la comprensión del contexto del sistema, y por lo tanto también contribuir a la comprensión de los requerimientos que se desglosan del mismo.

En la **Figura 1. Modelo de referencia de workflow (Interfaces y Componentes).**3 se muestra el modelo de dominio en el que se especifican las relaciones que existen entre los principales conceptos relacionados con el desarrollo de soluciones orientadas a procesos sobre la plataforma .NET.

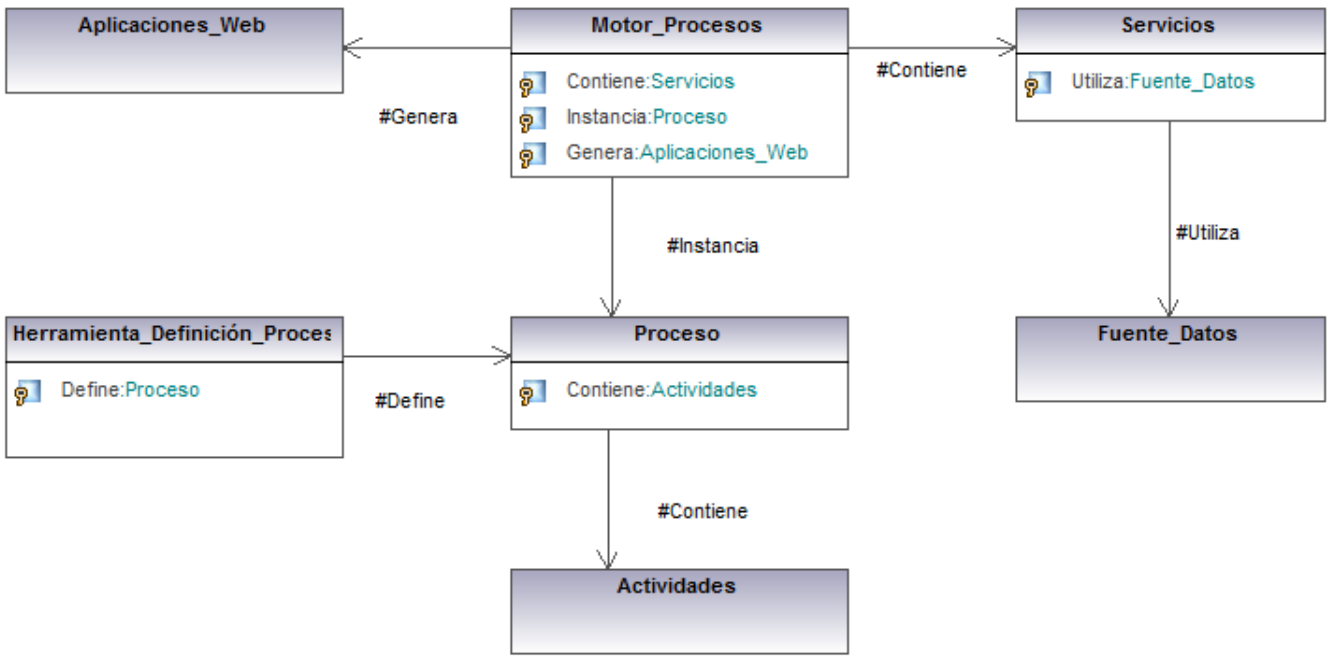


Figura 3. Descripción del modelo de dominio

Para obtener correctamente los requerimientos y poder construir un sistema se necesita tener un firme conocimiento del objeto de estudio del mismo. En un modelo de dominio se capturan los tipos más importantes de objetos. Estos objetos o clases, como también se les puede llamar, se obtienen de una especificación de requisitos o mediante la entrevista con expertos del dominio. Este modelo deja bien claro cómo funciona el entorno en el cual está enmarcada la herramienta. (20)

A continuación se definen todos los conceptos utilizados en el diagrama, mediante un glosario de términos.

Actividades: Las actividades son las unidades de creación fundamentales de los flujos de trabajo. Estas representan un paso dentro de la ejecución de una instancia de *workflow*. Se diseñan para servir un único objetivo y agrupan la lógica necesaria para cumplir ese propósito. Cada actividad tiene un contexto de ejecución que representa el entorno de ejecución de la misma.

Aplicaciones web: Es una solución *software* que permite que los usuarios puedan acceder a recursos, que automatiza la secuencia de acciones, actividades o tareas utilizadas para la ejecución del proceso. Específicamente son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador. Es una aplicación de *software* que se codifica en un lenguaje soportado por los navegadores web.

Definición de proceso: Contiene toda la información necesaria acerca de los procesos, incluye información de comienzo de actividades, condiciones, y reglas de navegación. Define el orden en que las tareas son ejecutadas, puede variar dependiendo de las características de cada una de ellas.

Fuentes de datos: Es conjunto de información almacenada en memoria auxiliar a la que se puede acceder mediante un conjunto de programas que manipulan esos datos. Fundamentalmente se usan sistemas de base de datos relacional por su evolución hacia nuevas tecnologías. Las bases datos son un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación.

Herramienta para la definición de proceso: Forma parte de los componentes de *software* del *workflow*. Es utilizada para crear una descripción de los procesos en una forma entendible para una computadora. Esta herramienta podría estar basada en un lenguaje de definición de procesos formal, en un modelo de interacción entre objetos, o simplemente en un conjunto de reglas para transferir información entre los participantes. Además, puede ser proporcionada como parte de un producto de software orientado a *workflow*, o podría simplemente existir por sí sola y tener integración con diferentes productos de *workflow*.

Motor de procesos (*RuntimeEngine*): Este componente provee el ambiente de ejecución para cada una de las instancias de *workflow*. Es una herramienta que permite automatizar los procesos de negocios.

Servicios (*RuntimeServices*): Estos servicios son instancias de clases que son creadas y registradas en el *runtime* durante el arranque de la aplicación. Cada servicio cumple un propósito específico para el cual se definió. Estos pueden existir en dos formas: servicios internos del *runtime* y servicios externos.

Instancia de flujo de trabajo (*WorkflowInstance*): Es una instancia ejecutable del flujo de trabajo (*workflow*). Es una ejecución o realización concreta de un proceso.

2.3 Propuesta del sistema

La propuesta en su vista más abstracta está compuesta por diferentes módulos relacionados entre sí. Cada módulo está diseñado siguiendo un patrón de capas bien definidas y diseñadas para reducir al mínimo el acoplamiento y aumentar la reutilización entre las mismas.

2.3.1 Descripción de la propuesta del sistema

En el modelo propuesto se definen una serie de componentes que en su conjunto conforman el *framework*, estos componentes interactúan entre ellos brindando una serie de funcionalidades. Se pueden identificar algunos conceptos.

Actividades: En este paquete se definen todas las actividades que por su comportamiento son necesarias en la arquitectura base. Son todas aquellas actividades que por su contenido lógico son utilizadas en más de una ocasión, ya sea dentro de un mismo proceso o en procesos distintos.

***Hosting*:** 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*. 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*.

Entidades: En este paquete se agrupan los objetos que se utilizan en el *framework*, el término entidad refiere a aquella representación de un objeto o concepto del mundo real.

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

Excepciones: En este paquete se encuentran las clases que permiten el tratamiento de excepciones. Además de realizar una traza de todas las excepciones que se lancen. Consta de una serie de clases que permiten identificar cada tipo de excepción para darle un tratamiento en particular a cada una.

***Framework*:** Forma parte de la capa de procesos y servicios. Su principal objetivo es encapsular en un componente todas aquellas actividades y servicios que son de vital importancia para la arquitectura.

2.4 Especificación de requisitos funcionales del sistema

El propósito fundamental del flujo de trabajo de los requisitos es guiar de manera correcta el desarrollo del sistema. Esto se consigue mediante una descripción de los requisitos del sistema (es decir, las condiciones o capacidades que el sistema debe cumplir) suficientemente buena como para que pueda llegarse a un acuerdo entre el cliente y los desarrolladores sobre qué debe y qué no debe hacer el sistema. (21). A continuación, en la tabla 1 se muestran la descripción de los roles.

2.4.1 Descripción de los roles

Rol	**Objetivo**
Desarrollador	Se beneficia de las funcionalidades del *framework*, puede habilitar o deshabilitar las características que sean necesarias para el desarrollo. Se encarga de realizar todas las configuraciones requeridas por las funcionalidades que habilite.
Sistema	Realiza las acciones que sean necesarias de acuerdo con las configuraciones realizadas por el desarrollador. Garantiza el correcto funcionamiento de todas las funcionalidades habilitadas por el desarrollador para su solución.

Tabla 1. Descripción de roles

2.4.2 Catálogo de requisitos

RF 1 Permitir la gestión de estados de una instancia de *workflow*.

RF1.1 Permitir habilitar el mecanismo de gestión de estados de instancias de *workflow*.

RF1.2 Permitir deshabilitar el mecanismo de gestión de estados de instancias de *workflow*.

RF1.3 Permitir configurar las propiedades del mecanismo de gestión de estados de instancias de *workflow*.

a) Medio persistente que será utilizado.

RF1.4 Permitir definir en tiempo de diseño los estados en que se puede encontrar una instancia de *workflow*.

RF1.5 Permitir buscar instancias de *workflow* según criterios.

a) Identificador de la instancia de *workflow*.

b) Nombre del estado.

RF1.6 Gestionar los estados de una instancia de *workflow*.

RF1.6.1 Permitir salvar el estado de una instancia de *workflow*.

RF1.6.2 Permitir eliminar el estado de una instancia de *workflow*.

RF 2 Permitir salvar las propiedades de una instancia de *workflow*.

RF 2.1 Permitir definir en tiempo de diseño las propiedades que se desean salvar de la instancia de *workflow*.

RF 2.2 Permitir salvar las propiedades de la instancia de *workflow*.

RF 2.3 Permitir eliminar las propiedades de la instancia de *workflow*.

2.4.3 Descripción de requisitos del sistema

Para profundizar en el análisis de los requisitos funcionales del sistema ver **¡Error! No se encuentra el rigen de la referencia.**

RF 1 Permitir la gestión de estados de una instancia de *workflow*.

RF1.1 Permitir habilitar el mecanismo de gestión de estados de instancias de *workflow*.

Propósito	Habilitar el mecanismo de gestión de estados de instancias de *workflow*.	
Roles	Desarrollador	
Precondiciones	El mecanismo de gestión de estados de instancias de *workflow* debe estar deshabilitado.	
Conceptos tratados	**Concepto**	**Atributos**
	StateWorkflowService	
Descripción	1. Añadir la entrada de activación del mecanismo de gestión de estados de instancias de *workflow* en el archivo de configuración en la sección de configuración de los componentes del *framework*. 2. Leer la configuración.	
Validaciones	Comprobar que exista la sección de configuración de los componentes del *framework*.	
Postcondiciones	Se habilitó el mecanismo de gestión de estados de instancias de *workflow*.	
Prototipo	------	

Tabla 2. Descripción del requisito funcional 1.1

RF1.2 Permitir deshabilitar el mecanismo de gestión de estados de instancias de *workflow*.

Propósito	Deshabilitar el mecanismo de gestión de estados de instancias de *workflow*.	
Roles	Desarrollador	
Precondiciones	El mecanismo de gestión de estados de instancias de *workflow* debe estar habilitado.	
Conceptos tratados	**Concepto**	**Atributos**
	StateWorkflowService	
Descripción	1. Suprimir la entrada de activación del mecanismo de gestión de estados de instancias de *workflow* en el archivo de configuración en la sección de configuración de los componentes del *framework*. 2. Leer la configuración.	
Validaciones	Comprobar que exista la sección de configuración de los componentes del *framework*.	
Postcondiciones	Se deshabilitó el mecanismo de gestión de estados de instancias de *workflow*.	

Prototipo	------

Tabla 3. Descripción del requisito funcional 1.2

RF1.3 Permitir configurar las propiedades del mecanismo de gestión de estados de instancias de *workflow*.

Propósito	Configurar el mecanismo de gestión de estados de instancias de *workflow*.	
Roles	Desarrollador	
Precondiciones	El mecanismo de gestión de estados de instancias de *workflow* debe estar habilitado. El mecanismo de gestión de estados de instancias de *workflow* debe estar configurado.	
Conceptos tratados	**Concepto**	**Atributos**
	StateWFManagerServiceSection	• *name* • *type*. • *connectionString*
Descripción	1. Añadir la sección para gestión de estados de instancias de *workflow* en el archivo de configuración de la solución. • *name*. • *type*. 2. Inicializar sección de configuración del mecanismo de gestión de estados de instancias de workflow. • *connectionString*. 3. Leer la configuración.	
Validaciones	Comprobar que exista la sección de configuración de los componentes del *framework*.	
Postcondiciones	Se configuró el mecanismo de gestión de estados de instancias de *workflow*.	
Prototipo	------	

Tabla 4. Descripción del requisito funcional 1.3.

RF1.4 Permitir definir en tiempo de diseño los estados en que se puede encontrar una instancia de *workflow*.

Propósito	Definir en tiempo de diseño los estados en que se puede encontrar

	una instancia de *workflow*.	
Roles	Desarrollador	
Precondiciones	El mecanismo de gestión de estados de instancias de *workflow* debe estar habilitado. El mecanismo de gestión de estados de instancias de *workflow* debe estar configurado.	
Conceptos tratados	**Concepto**	**Atributos**
	StateActivity	• *state*
Descripción	1. Declarar una variable de tipo *enum* con cada uno de los estados en que se encuentra. 2. Especificar en cada *stateActivity* que se utilice en el diseño del *workflow* el estado en que se encuentra el *workflow*.	
Validaciones	Debe escoger un estado en cada *stateActivity* que se utilice en el diseño del *workflow*.	
Postcondiciones	Se definió en tiempo de diseño los estados en que se puede encontrar una instancia de *workflow*.	
Prototipo	------	

Tabla 5. Descripción del requisito funcional 1.4

RF1.5 Permitir buscar instancias de workflow según criterios.

Propósito	Buscar instancias de workflow según criterios.	
Roles	Desarrollador.	
Precondiciones	El mecanismo de gestión de estados de instancias de workflow debe estar habilitado. El mecanismo de gestión de estados de instancias de workflow debe estar configurado.	
Conceptos tratados	**Concepto**	**Atributos**
	SearchWorkflowService	
Descripción	1. Especificar el criterio por el cual desea buscar la instancia de *workflow*. 2. Estado de la instancia de *workflow* 3. Tipo de la instancia de *workflow*	
Validaciones	Comprobar que exista conexión con la base de datos.	

Postcondiciones	Se buscó una instancia de *workflow* por un criterio específico.
Prototipo	------

Tabla 6. Descripción del requisito funcional 3.5

RF1.6 Gestionar los estados de una instancia de *workflow*.

Propósito	Guardar las propiedades del *workflow* en un estado específico de una instancia de *workflow*.	
Roles	Desarrollador	
Precondiciones	El mecanismo de gestión de estados de instancias de *workflow* debe estar habilitado. El mecanismo de gestión de estados de instancias de *workflow* debe estar configurado.	
Conceptos tratados	**Concepto**	**Atributos**
	StateWorkflowService	
Descripción	1. Identificar la instancia de *workflow*. 2. Eliminar el estado actual de la instancia de *workflow*. 3. Salvar el estado de la actual de la instancia de *workflow*.	
Validaciones	Comprobar que exista conexión con la base de datos.	
Postcondiciones	Se actualizó de estado de una instancia de *workflow*.	
Prototipo	------	

Tabla 7. Descripción del requisito funcional 1.6

RF 2 Permitir salvar las propiedades de una instancia de *workflow*.

RF 2.1 Permitir definir en tiempo de diseño las propiedades que se desean salvar de la instancia de *workflow.*

Propósito	Definir en tiempo de diseño las propiedades de la instancia de *workflow* que se desean salvar.	
Roles	Desarrollador	
Precondiciones	El mecanismo para salvar las propiedades de la instancia de *workflow* debe estar habilitado.	
Conceptos tratados	**Concepto**	**Atributos**
	propertyWorkflow	• *path*

		• *name* • *valueInfo*
Descripción	1. Agregar la actividad *ConfigurationActivity*. 2. Definir las propiedades de la instancia de workflow que se desean salvar.	
Validaciones	Comprobar que se exista la actividad *ConfigurationActivity* en el diseño del *workflow*.	
Postcondiciones	Se definieron las propiedades de la instancia de *workflow* que se desean salvar.	
Prototipo		

Tabla 8. Descripción del requisito funcional 2.1

RF 2.1 Permitir salvar las propiedades de la instancia de *workflow*

Propósito	Salvar las propiedades de la instancia de *workflow*.	
Roles	Desarrollador	
Precondiciones	Las propiedades de la instancia de *workflow* que se desean salvar deben estar definidas.	
Conceptos tratados	**Concepto**	**Atributos**
	propertyWorkflow	• *path* • *name* • *valueInfo*
Descripción	1. Agregar la actividad *SavePropertyWorkflow*	
Validaciones	Comprobar que exista conexión con la base datos.	
Postcondiciones	Se salvó las propiedades de la instancia de *workflow* en la base datos.	
Prototipo		

Tabla 9. Descripción del requisito funcional 2.2

RF 2.2 Permitir eliminar las propiedades de la instancia de *workflow*.

Propósito	Eliminar las propiedades de la instancia de *workflow*.	
Roles	Desarrollador.	
Precondiciones	Las propiedades de la instancia de *workflow* han sido salvadas con anterioridad en la base datos.	
Conceptos tratados	**Concepto**	**Atributos**

	propertyWorkflow	• *path* • *name* • *valueInfo*
Descripción	1. Las propiedades de la instancia de *workflow* se eliminan de la base de datos automáticamente cuando se completa la instancia de *workflow*.	
Validaciones	Comprobar que exista conexión con la base datos.	
Postcondiciones	Se eliminó las propiedades de la instancia de *workflow* de la base datos.	
Prototipo		

Tabla 10. Descripción del requisito funcional 2.3

Para profundizar en la descripción de requisitos funcionales del sistema ver **¡Error! No se encuentra l origen de la referencia.**

2.5 Especificación de requisitos no funcionales

Los requerimientos no funcionales se pueden definir como propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. En este caso los requisitos están enfocados a identificar una serie de características que no están comprendidas en las funcionalidades pero a su vez garantizan el óptimo rendimiento del *framework* y faciliten su uso por los desarrolladores.

Requisitos de *software*

RnF1 El sistema podrá ser utilizado bajo el sistema operativo Windows XP o superior.

RnF2 El sistema requiere de .NET Framework v3.5 o superior.

RnF3 El sistema requiere la librería de Oracle que permita conectarse al servidor de aplicaciones.

Requisitos de diseño y de implementación.

RnF4 El sistema requiere utilizar las tecnologías que brinda el *Framework* .NET sobre la programación orientada a objetos.

Requisitos de seguridad

RnF5 El sistema garantizará un fuerte tratamiento de excepciones.

RnF6 El sistema tendrá mecanismos de tolerancia ante fallos que permitan recuperarse antes errores.

RnF7 La conexión a los servidores de bases de datos y de aplicaciones es requerida para el correcto funcionamiento.

RnF8 El sistema garantizará la mayor exactitud en los datos.

a) Deberán realizarse validaciones y comprobaciones automáticas en todos los casos posibles para garantizar la consistencia de los datos.

Requisitos de Usabilidad

RnF9 El sistema facilitará el uso para variedad de desarrolladores de *software*, incluyendo personas con pocos conocimientos en el uso del *framework*.

RnF10 El sistema será distribuido en idioma español, aunque estará preparado para soporte multilenguaje.

RnF11 El proceso de desarrollo del *framework* será documentado.

RnF12 Proveer un manual de ayuda a los usuarios que utilicen el *framework*.

Requisitos de Soporte

RnF13 EL sistema facilitará la adaptación a diferentes ambientes sin necesidad de usar otros medios.

Requisito de Rendimiento

RnF14 El sistema debe garantizar la mayor eficiencia posible en cuanto al tratamiento de la información, de manera que la velocidad de procesamiento de la misma sea la mejor.

RnF15 Se debe garantizar la consistencia y disponibilidad de la información en todo momento, por lo que se requiere además un tiempo de recuperación mínimo.

RnF16 El sistema debe hacer un uso racional de los recursos de *hardware*.

2.6 Conclusiones

- El modelado del dominio permitió diseñar una propuesta de solución del sistema, obteniéndose además los requisitos funcionales y no funcionales del mismo.
- La descripción de los requisitos funcionales y no funcionales del sistema proporcionó un conocimiento detallado de las funcionalidades que deben desarrollarse para cumplir con los objetivos de la investigación.

CAPÍTULO III. ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se realiza un modelado del sistema propuesto y se define la arquitectura del mismo. Además, se especifica el comportamiento de la solución en cada uno de los servicios y componentes que se proponen, así como los patrones de diseño utilizados. Se realiza el diseño de la base de datos, teniendo en cuenta el diagrama de clases persistentes y el modelo de datos.

3.2 Arquitectura de la solución

La solución es un *framework* para el desarrollo de aplicaciones web basadas en *workflow*, que permite la orquestación de procesos de negocio con *Windows Workflow Foundation*. Su principal objetivo es proporcionar un componente que permita gestionar las instancias de *workflow*. Además, agrupa un conjunto de actividades y servicios que le dan mayor dinamismo al desarrollo de sistemas centrado en la orquestación de procesos de negocio, específicamente para un ambiente web.

3.2.1 Estructura

El *framework* se ha desarrollado con una estructura que permite el encapsulamiento en paquetes de las actividades, servicios y entidades que se encuentran relacionadas para dar solución a un requerimiento en particular. Se caracteriza por contener elementos que permiten una implementación de forma gráfica lo más cercana posible al modelo del proceso de negocio. Las actividades y servicios que se encuentran en este surgen como complemento de las funcionalidades que brinda el *framework* de la plataforma .NET, centrándolo principalmente en un ambiente de desarrollo web. En la figura 4 se puede observar la estructura del *framework*.

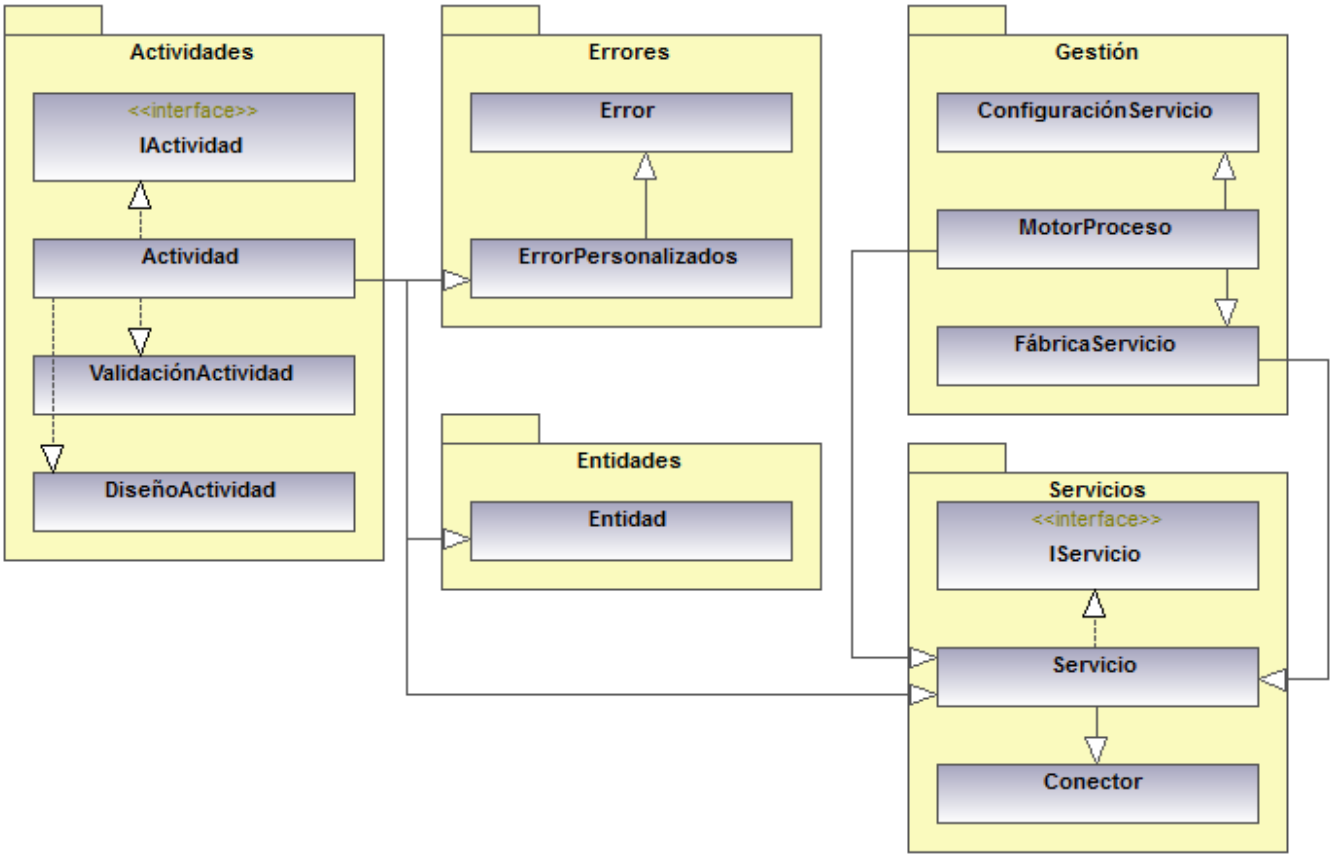


Figura 4. Estructura del *framework*

Actividades (*activities*)

En el paquete actividades se definen todas las actividades que por su comportamiento son necesarias en la arquitectura base que propone el uso del *framework* específicamente para un ambiente web. Las actividades son funciones o acciones que cumplen una funcionalidad en particular permitiendo extender las capacidades gráficas del *workflow*. Uno de los principales objetivos del *framework* es lograr modelar la mayor parte de la lógica de negocio de forma gráfica en un *workflow*, incluyendo las pantallas de interacción de usuario que se deben mostrar.

Servicios (*services*)

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.

- Implementan los controladores que atenderán las funcionalidades.

- Representan las operaciones referentes a una entidad de negocio y aquellas que por concepto de composición se encuentran estrictamente relacionadas con esta.
- Definen las interfaces que tienen que ser implementados por los conectores que vinculan el servicio con la fuente de datos correspondientes.
- Define la interfaz que ha de ser implementada por la fábrica de los conectores según la fuente de datos a utilizar.

Gestión (Hosting)

En este paquete se encuentran las clases 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 runtime.

Entidades (*Entities*)

En este paquete se agrupan los objetos que se utilizan en el *framework*, el término entidad refiere a aquella representación de un objeto o concepto del mundo real.

Errores (*Exceptions*)

En este paquete se encuentran las clases que permiten el tratamiento de excepciones. Además de realizar una traza de todas las excepciones que se lancen. Consta de una serie de clases que permiten identificar cada tipo de excepción para darle un tratamiento en particular a cada una.

Marco de trabajo (*Framework*)

Forma parte de la capa de procesos y servicios. Su principal objetivo es el de encapsular en un componente todas aquellas actividades y servicios que son de vital importancia para la arquitectura.

3.3 Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia y que se ha demostrado que funcionan (21). Existen problemas durante el desarrollo de *software* que se repiten o que son análogos, que responden a un cierto patrón. Por lo que es deseable tener una colección de dichos patrones con soluciones óptimas para cada caso a modo de buenas prácticas. Durante el desarrollo del *framework* se utilizaron un conjunto de patrones entre los que se destacan:

Encapsulamiento: Propone esconder algunos componentes, permitiendo sólo 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 que son exclusivamente necesarios. En la figura 5 se observa la utilización de este patrón.

```
public class BisonRuntime
{
    private static BisonRuntime bisonHost;

    public static BisonRuntime BisonInstance...

    private void ExternalMethodCall(object sender, ExternalDataEventArgs e)...
```

Figura 5. Utilización del patrón encapsulamiento

Subclase: Este patrón 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. En la figura 6 se observa la utilización de dicho patrón.

```
[Serializable]
public class SerializerException : Exception
{
    public SerializerException()
        : base("No se puede serializar el objeto")
    { }

    public SerializerException(string message)
        : base(message)
    { }
```

Figura 6. Utilización del patrón subclase

Excepciones: Propone introducir estructuras de lenguaje para lanzar 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. En la figura 7 se observa la utilización del patrón fábrica.

```
public static class BisonFactory
{
    /// <summary>
    /// Servicio de Navegación
    /// </summary>
    public static IWebNavigatorService WebNavigatorService
    {
        get
        {
            return BisonRuntime.GetService<IWebNavigatorService>();
        }
    }
```

Figura 7. Utilización del patrón fábrica

Instancia única: 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: Son un conjunto de patrones para la asignación de responsabilidades, el uso de este patrón está totalmente ligado a cada componente desarrollado en el sistema, donde cada uno de ellos posee solo las funcionalidades acordes con las particularidades que lo caracterizan.

Patrones de *workflow*

Patrones de control de flujos básicos: Estos patrones están presentes en la mayoría de los lenguajes de *workflow*, y sirven para modelar procesos secuenciales, paralelos, y aquellos que incluyan alguna decisión. En la figura 8 se observa la utilización de este patrón de *workflow*.

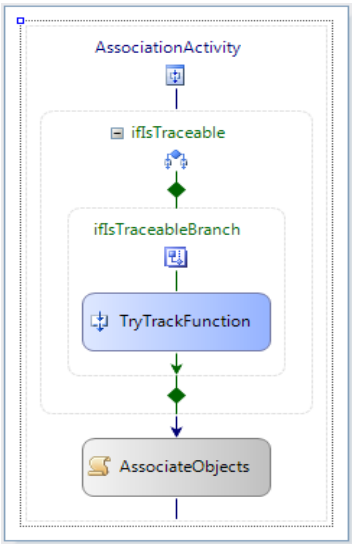


Figura 8. Utilización del patrón de *workflow* control de flujo básico

Patrones de ramificación avanzada y sincronización: Estos patrones superan a los patrones de control de flujo básico al permitir tipos avanzados de bifurcación y sincronización. En la figura 9 se observa la utilización del patrón de *workflow* ramificación avanzada y sincronización.

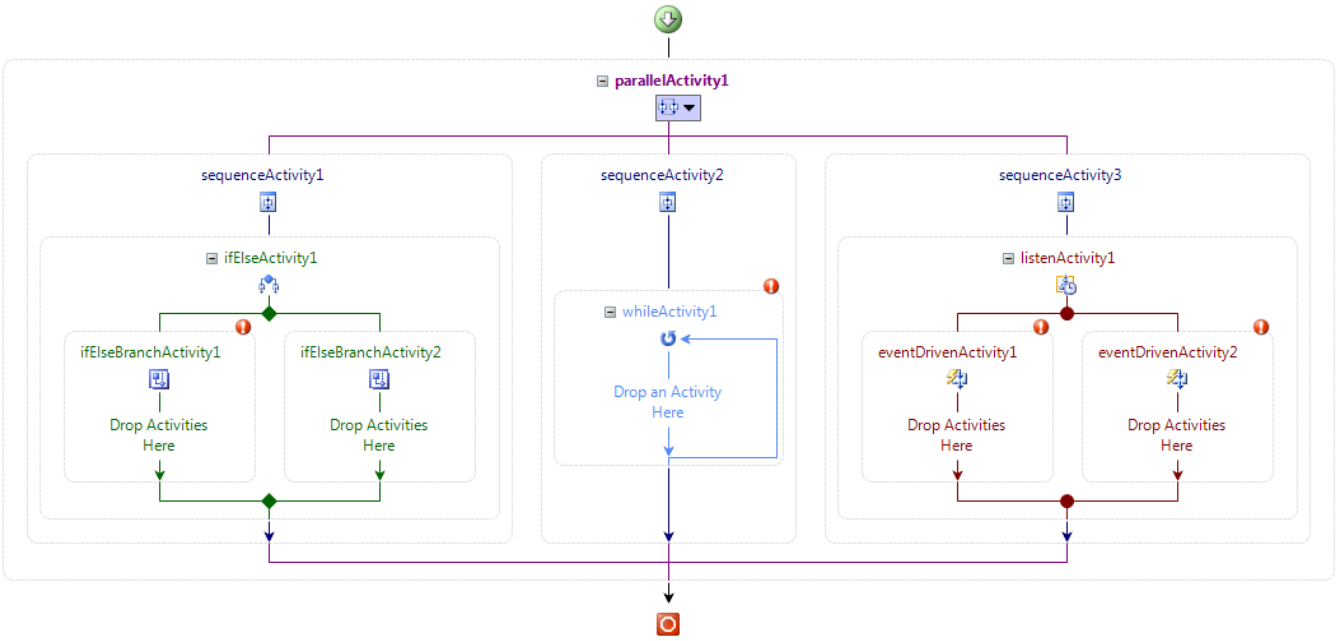


Figura 9. Utilización del patrón de *workflow* ramificación avanzada y sincronización

3.4 Diagramas de clases del diseño

En la figura 10 se observa el diagrama de clases del diseño del servicio *StateWorkflowService y el servicio SearchWorkflowService*.

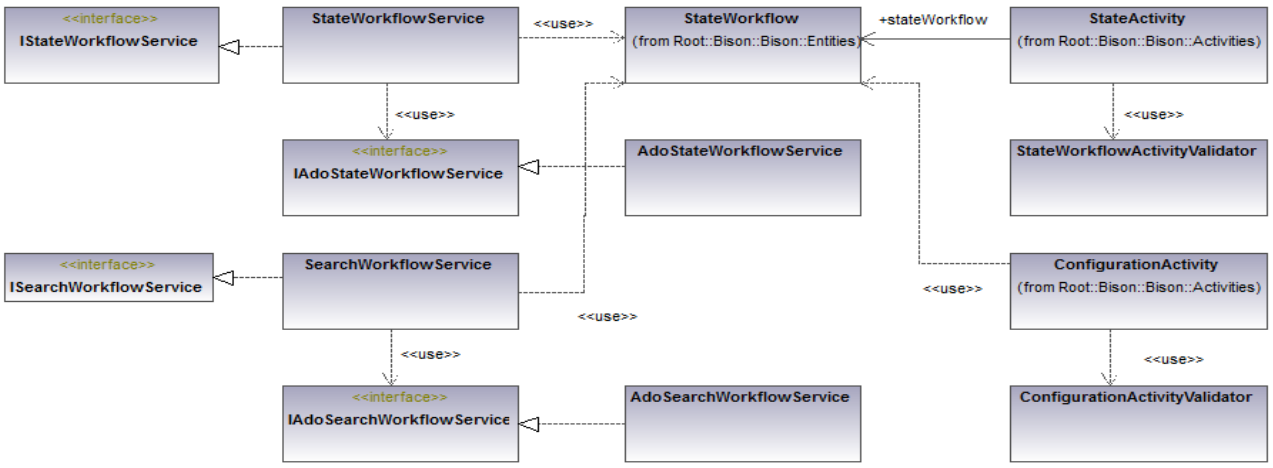


Figura 10. Diagrama de clases del diseño del servicio *StateWorkflowService*

Para ver el diagrama de clases del diseño en su totalidad ver **¡Error! No se encuentra el origen de la eferencia.**

3.5 Especificación de las clases del diseño

A continuación se realiza la especificación de las clases del diseño del servicio para salvar los estados del *workflow*. Este servicio salva el estado de la instancia de *workflow* en cualquier punto dentro de la ejecución según la necesidad.

Conectores

Los conectores en el *framework* tienen la responsabilidad de manejar las entidades del negocio que es necesario persistir en base de datos. Permiten establecer una conexión entre las entidades del negocio y la base datos. En la tabla 8 se puede observar una descripción del conector utilizado por el servicio para salvar los estados del *workflow*. Para ver una descripción más detallada de los conectores ver **¡Error! No se encuentra el origen de la referencia.**

Nombre del Conector: AdoStateWorkflowService		
Descripción: Acceso a datos del servicio de estados del *workflow*. Conexión de las entidades del negocio que utiliza el servicio de estados del *workflow* con la base de datos Oracle.		
Nombre del método	**Parámetros (tipo de dato)**	**Descripción**
SaveStateWorkflow	stateworkflow (*StateWorkflow*)	Salva en la base de datos el estado de la instancia de *workflow*.
DeleteStateWorkflow	workflowInstanciaId (Guid) estado (string)	Elimina de la base el estado de una instancia de *workflow* específica dado el identificador de la instancia de *workflow* y el estado en que se encuentra.

Tabla 11. Descripción del conector *AdoStateWorkflowService*

Entidades

Las entidades tienen como principal objetivo describir el modelo de objetos relacionadas en el negocio. Estas son contenedoras de información que fluyen a través de las capas de presentación, procesos, servicios y negocio. En la tabla 9 se puede observar una descripción de la entidad utilizada por el

servicio para salvar los estados del *workflow*. Para ver una descripción más detallada de las entidades utilizadas ver **¡Error! No se encuentra el origen de la referencia.**

Nombre de la Entidad: StateWorkflow		
Descripción: Representa el estado en que se encuentra una instancia de *workflow* en un momento determinado.		
Nombre del atributo	**Tipo de dato**	**Descripción**
workflowInstanceId	string	Identificador de la instancia de *workflow*.
state	string	Nombre del estado en que se encuentra la instancia de *workflow*.
nameWorkflow	string	Nombre del *workflow*.

Tabla 12. Descripción de la entidad StateWorkflow

Servicios

Los servicios en el *framework* representan un factor clave dentro de la aplicación ya que son los que permiten la interacción con las instancias creadas. Estos son utilizados para intercambiar datos con aplicaciones externas, permiten además la sincronización de instancias de procesos en memoria con instancias en bases de datos.

Servicio de estados del workflow (*StateWorkflowService*): Este servicio salva el estado de la instancia de *workflow* en cualquier punto dentro de la ejecución de la instancia de *workflow* según la necesidad. En la tabla 10 se puede observar una descripción de los métodos utilizados por el servicio para salvar los estados del *workflow*.

Nombre del Servicio: StateWorkflowService.		
Descripción: Servicio para salvar los estados de la instancia de *workflow*.		
Nombre del método	**Parámetros(tipo de dato)**	**Descripción**
SaveStateWorkflow	stateworkflow (StateWorkflow)	Salva en la base de datos el estado de la instancia de *workflow*.
DeleteStateWorkflow	workflowInstanceId (Guid)	Elimina de la base de datos el estado de una instancia

	estado (string)	de *workflow* específica dado el identificador de la instancia de *workflow* y el estado en que se encuentra.

Tabla 13. Descripción del servicio para salvar los estados del workflow

Servicio de persistencia (*GenericPersistenceService*): Este provee un mecanismo para salvar las instancias de *workflow* en la base de datos y cargarlas en memoria cuando sea necesario. El servicio debe ser inicializado una sola vez, cuando se inicializa el *workflow runtime*. Este no requiere ninguna intervención manual, todo el proceso es manejando por el *runtime*. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el origen de la eferencia.**

Servicio de trazas (*GenericTrackingService*): Este servicio provee un mecanismo para realizar trazas de todos los eventos y cambios ocurridos durante la ejecución de una instancia de *workflow*. Se puede agregar tantos servicios de trazas como se desee y que utilicen diferentes bases de datos. Una vez que el servicio es añadido el *runtime* le pasa automáticamente los datos de las trazas. Por defecto el servicio de trazas guarda los cambios y eventos ocurridos en todos los posibles tipos de *workflow* y actividades. El servicio brinda la posibilidad de configurar múltiples perfiles de trazas. Siendo flexible a la hora de decidir a qué tipos de *workflow* y eventos se le desea realizar trazas. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el rigen de la referencia.**

Servicio de navegación (*WebNavigatorService)*: Este servicio permite mostrar una interfaz web. El mismo tiene la responsabilidad de navegación dentro de una aplicación web. Está capacitado para identificar de las actividades que exponen una interfaz de usuario y se encuentran pendientes de ejecución dentro de una instancia de *workflow*. El mismo hace uso del servicio de autorización para definir cuáles de esas actividades le corresponden a un usuario en particular. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el origen de la eferencia.**

Servicio para salvar las propiedades del workflow (*SaveWorkflowPropertyService*): El servicio para salvar las propiedades del *workflow* permite salvar los valores de las propiedades en cualquier punto dentro de la ejecución de la instancia según la necesidad. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el origen de la referencia.**

Servicio para guardar los cambios de los datos (*RecordTrackingChanges*): El servicio permite guardar los cambios de los datos, permite realizar una traza de los valores que cambian a lo largo de

un proceso. Este servicio brinda la posibilidad de crear una traza en base de datos con las propiedades que han sido modificadas, tanto el valor anterior como el nuevo. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el origen de la eferencia.**

Servicio de valores por referencia (*ReferenceValueService*): Cuando se trabaja con *workflow* es conocido que la persistencia se realiza serializando la instancia de *workflow*, y de esta manera se mantiene la integridad de los datos, además de que la información se persiste al mismo tiempo que lo hace la actividad. Para evitar que el tamaño de esta información sea demasiado grande se ha creado el servicio de valores por referencia, que permite guardar en la fuente de datos un objeto que consuma mucha memoria como es el caso de las imágenes. A través de este servicio se puede obtener, salvar y modificar el valor de un objeto, quedando de manera transparente la realidad de que no se posee el objeto sino una referencia al lugar donde se encuentra en la fuente de datos. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el origen de la eferencia.**

Servicio para invocar workflow (*InvokeWorkflowService*): Este servicio tiene la responsabilidad de ejecutar una acción cada cierto tiempo con el objetivo de verificar si existen nuevas solicitudes en un estado en específico. Dependiendo del estado se decide cual es la acción a realizar para esta solicitud. En una instancia de *workflow* se realiza la solicitud para invocar una instancia de otro *workflow*, donde se guardan las propiedades que se desean pasen a la nueva instancia, así como la solicitud. Más tarde la solicitud es atendida por un servicio que creará la nueva instancia asignándole la información especificada, al terminarse la ejecución de este último se guarda la solicitud y los datos que fueron especificados se deseaban obtener de dicha instancia. Por último, la solicitud es atendida y se le avisa a la instancia invocadora que ya se ha terminado la acción, esta una vez es avisada continua su ejecución ya que se había quedado en estado de espera al realizar la solicitud. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el rigen de la referencia.**

Servicio para serializar información (*SerializarService*): Para lograr un correcto funcionamiento y almacenamiento de información desconocida en muchos casos se ha hecho uso de la serialización de objetos. El servicio para serializar información permite estandarizar este comportamiento. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el rigen de la referencia.**

Servicio para obtener valores del workflow (*DefaultWorkflowQueryService*): El trabajo con *workflow* permite desarrollar aplicaciones que contengan la mayor parte de la información correspondiente al proceso dentro de este, por lo que para poder visualizar u obtener la información hay que desarrollar un mecanismo que se encargue de proveer los datos que se encuentran dentro de una instancia en particular de un *workflow*. El servicio para obtener valores del *workflow* permite acceder a los valores que se encuentren dentro de una instancia particular. Para ver una descripción más detallada del servicio para obtener valores del *workflow* ver **¡Error! No se encuentra el origen de la referencia.**

Servicio de control de tiempo (*SmartTimerService*): Dentro de un servidor de aplicaciones en muchas ocasiones se pueden encontrar servicios que tienen que estar en constante ejecución, que se inicien cada cierto tiempo. Este servicio sirve de base para crear servicios similares, que al mismo tiempo permite darle una configuración en particular para cada uno de sus especializaciones. A partir de esta clase se puede estandarizar el comportamiento y la configuración de estos servicios dentro del servidor de aplicaciones. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el origen de la referencia.**

Servicio para la búsqueda de workflow (*SearchWorkflowService*): Este servicio da soporte a las acciones realizadas por la actividad *StateActivity*. Su principal objetivo es buscar los estados por los que puede haber transitado una instancia de *workflow*. Permite obtener todas las instancias de un *workflow* que se encuentran en un estado en particular, guardar y obtener información relevante en un estado específico. Para ver una descripción más detallada de los métodos incluidos en el servicio ver **¡Error! No se encuentra el origen de la referencia.**

3.6 Diseño de actividades

Las actividades son funciones o acciones que cumplen una funcionalidad en particular permitiendo extender las capacidades gráficas y funcionalidades del *workflow*. Permiten modelar la mayor parte de la lógica de negocio de forma gráfica en un *workflow*, incluyendo las interfaces gráficas de interacción con el usuario que se deben mostrar.

Actividad para representar la interacción con usuarios (*ClientActivity, TabControlActivity*): Esta actividad permite definir las interacciones del proceso con los usuarios de la aplicación. Estas interfaces gráficas pueden ser de diferentes tipos (*Page*, *UserControl*, *Tab*, *TabControl*). Para su uso es necesario especificarle la dirección del componente representado y el tipo de interfaz que se desea mostrar. En la figura 11 se puede observar el diseño de la actividad.

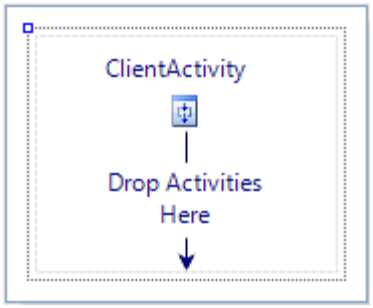


Figura 11. Actividad *ClientActivity* utilizada en el framework

Actividad para gestionar interfaces gráficas (*TabControlActivity*): Esta actividad permite mostrar un conjunto interfaces gráficas separadas por categoría, funcionalidades y que al mismo tiempo se puedan restringir según el usuario correspondiente, este componente se puede modelar igual que el resto de las actividades del framework, los *TabControlActivity* tienen la capacidad de representar un *TabActivity*, donde cada *TabActivity* tiene asociado una interfaz gráfica. En la figura 12 se puede observar el diseño de la actividad.

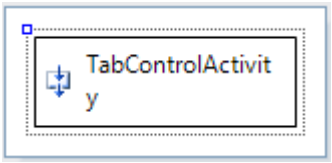


Figura 12. Actividad *TabControlActivity* utilizada en el framework

Actividad para representar interfaces gráficas (*TabActivity*): Esta actividad permite reconocer que *ClientActivity* que se está intentando cargar corresponde a un *TabControlActivity* creando un *TabControl* en la página a mostrar. Este componente se crea de manera dinámica según la información aportada por los servicios del *framework*, y de esta forma permite el acceso a todos los elementos que se tienen permiso. En la figura 13 se puede observar el diseño de la actividad.

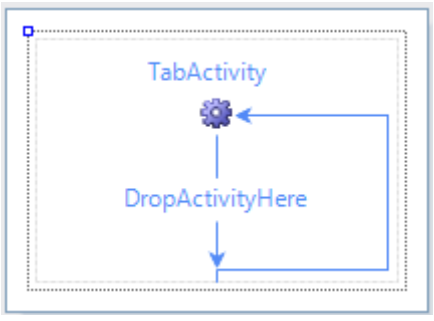


Figura 13. Actividad *TabActivity* utilizada en el framework

Actividad para gestionar de estados (*StateActivity*): Esta actividad permite gestionar el estado del negocio de la instancia de *workflow*. La misma tiene la responsabilidad de salvar el estado del negocio de la instancia de *workflow*, así como su actualización y eliminación. El estado del negocio de la instancia que es manejado por esta actividad debe ser definido anteriormente en la actividad *ConfigurationActivity*. En la figura 14 se puede observar el diseño de la actividad.

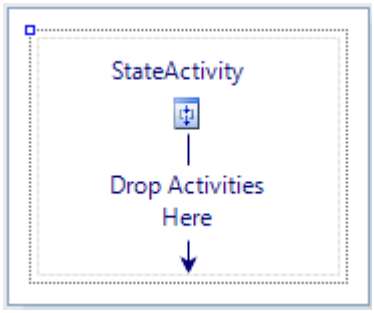


Figura 14. Actividad StateActivity utilizada en el framework

Actividad para salvar las propiedades (*SavePropertyWorkflow*): Esta actividad permite salvar y persistir los valores de las propiedades en cualquier punto dentro de la ejecución de la instancia de *workflow* según la necesidad. Estos valores son especificados con anterioridad en la actividad ConfigurationActivity. En la figura 15 se puede observar el diseño de la actividad.

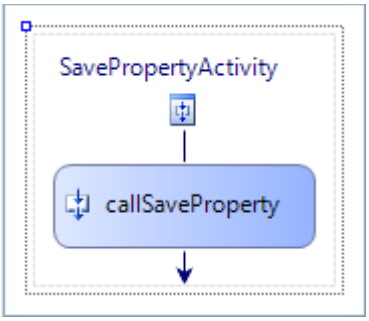


Figura 15. Actividad *SavePropertyWorkflow* utilizada en el framework

Actividad encargada de asociar valores (*AssociationActivity*): Esta actividad permite la asociación de un valor a una propiedad dentro del *workflow*. El valor origen puede ser una variable de entrada o bien un valor string que sea pasado en una propiedad de la actividad. Además de representar la asociación de valores de manera gráfica también se encarga de dejar una traza de los valores que han sido modificados durante la ejecución de la instancia de *workflow* a lo largo del proceso. En la figura 16 se puede observar el diseño de la actividad.

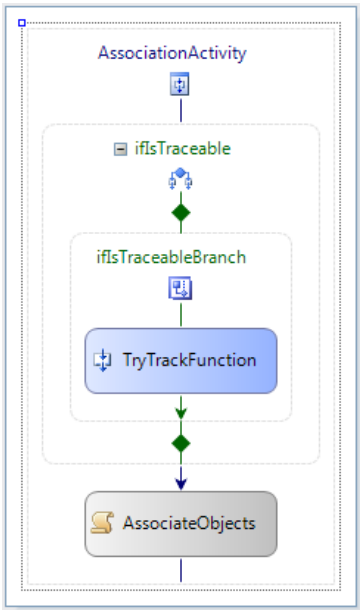


Figura 16. Actividad *AssociationActivity* utilizada en el framework

Actividad para invocar un workflow (*InvokeWorkflowActivity*): Esta actividad permite dentro de la ejecución de una instancia de *workflow* iniciar una nueva instancia de un *workflow*, el cual reciba valores iniciales que se encuentran solo activos dentro del proceso que lo manda a iniciar. Este tiene la particularidad que permite definir el intervalo de tiempo en el cual se iniciará la nueva instancia de workflow. En la figura 17 se puede observar el diseño de la actividad.

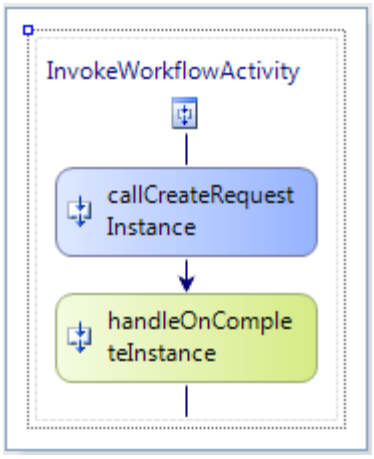


Figura 17. Actividad InvokeWorkflowActivity utilizada en el framework

Actividad para configurar el workflow (*ConfigurationActivity*): Esta actividad permite obtener o consultar la información de una instancia de *workflow*, tiene como base las colas utilizadas por el *workflow* para realizar sus acciones. Una que vez la instancia de *workflow* ha ejecutado la actividad,

queda registrada en la cola de la misma un objeto que responde a las peticiones de información enviadas. Este objeto recibe la petición y en el campo *response* de esta pone la información correspondiente a la propiedad que se hace referencia en una propiedad de dicha solicitud. Además, es posible especificar cuáles son los valores que se desean guardar y persistir para que sea posible realizar una búsqueda por dicha información o mostrarla según sea la necesidad. Permite definir los estados de negocio del *workflow*. Es importante hacer notar que los estados definidos para una instancia de *workflow* son diferentes de estos estados. En la figura 18 se puede observar el diseño de la actividad.

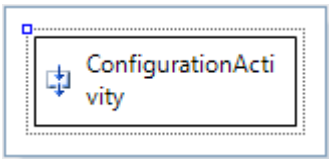


Figura 18. Actividad *ConfigurationActivity* utilizada en el framework

3.7 Modelo de datos

A continuación se presenta el modelo de datos utilizado por el servicio de estados del *workflow* el cual es una fracción del modelo de datos utilizado en el *framework*, este modelo es utilizado por el servicio de persistencia, el servicio de estados y el servicio de invocación de *workflow*. En la figura 19 se puede observar una imagen del modelo de datos. Para analizar el modelo de datos en su totalidad ver **¡Error! o se encuentra el origen de la referencia.**

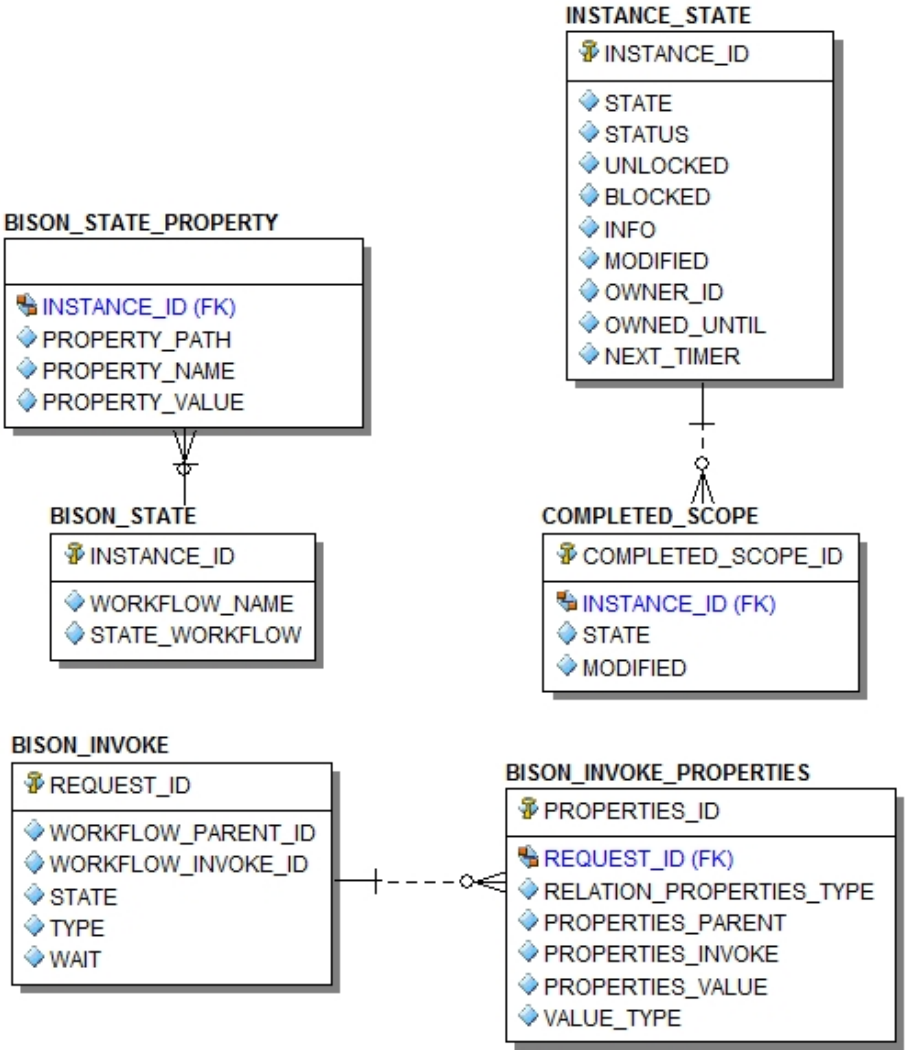


Figura 19. Modelo de datos utilizado por los servicios del framework

3.7.1 Descripción de las entidades del modelo de datos

A continuación se describen las entidades utilizadas por el servicio de estados del *workflow*. En la tabla 11 se puede observar una descripción de la entidad Bison_State_Property del modelo de datos.

Nombre de la Entidad: Bison_State_Property.			
Descripción de la Entidad: Representa las propiedades del *workflow* en un estado específico.			
Servicio: StateWorkflowService.			
Atributo	**Tipo de Dato**	**Nulo**	**Descripción**

Instance_Id	Char (36)	No	Identificador del estado en que se encuentra una instancia de *workflow*.
Property_Path	Nclob	No	Dirección de la propiedad dentro del *workflow*.
Property_Name	Nclob	Sí	Nombre de la propiedad.
Property_Value	Nclob	Sí	Valor de la propiedad.

Tabla 14. Descripción de la tabla Bison_State_Property

En la tabla 12 se puede observar una descripción de la entidad Bison_State del modelo de datos.

Nombre de la Entidad: Bison_State.			
Descripción de la Entidad: Representa el estado en que se encuentra una instancia de *workflow*.			
Servicio: StateWorkflowService.			
Atributo	**Tipo de Dato**	**Nulo**	**Descripción**
Instance_Id	Char (36)	No	Identificador del estado en que se encuentra una instancia de *workflow*.
Workflow_Name	Nvarchar (100)	No	Nombre del *workflow*.
State_Workflow	Nvarchar (100)	No	Estado en que se encuentra la instancia de *workflow*.

Tabla 15. Descripción de la tabla Bison_State

Para ver una descripción más detallada de las entidades del modelo de datos ver **¡Error! No se ncuentra el origen de la referencia.**

3.7.2 Descripción de los procedimientos almacenados

Estructura de paquetes que encapsulan los procedimientos almacenados utilizados en el *framework*. En la figura 20 se puede observar una imagen con los paquetes de procedimientos almacenados utilizados en el *framework*.

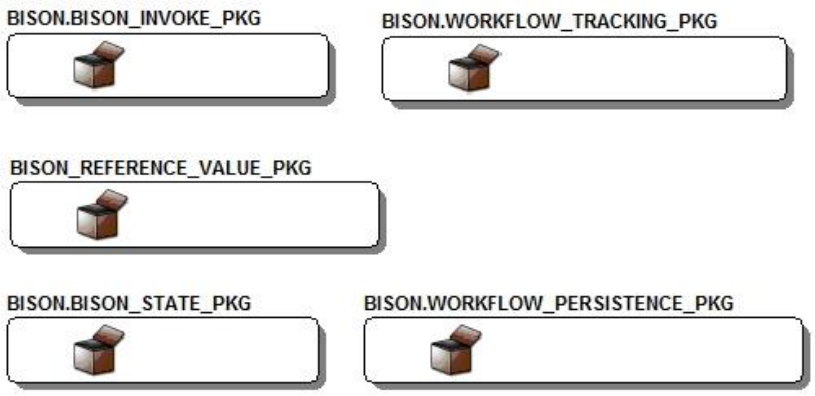


Figura 20. Paquetes de procedimientos almacenados utilizados en el *framework*

Paquete de procedimientos almacenados utilizados por el servicio de estados de *workflow* del *framework*. En la tabla 13 se puede observar una descripción de los procedimientos contenidos en el paquete Bison_State_PKG.

Nombre	Descripción
InsertState.	Inserta el estado de una instancia de *workflow*.
DeleteState.	Elimina el estado de una instancia de *workflow*.
FindStateByInstanceId.	Localiza el estado de una instancia de *workflow* dado su identificador.
FindInstancByState.	Localiza una instancia de *workflow* dado su estado.
InsertPropertyWorkflow.	Inserta las propiedades de una instancia de *workflow* por las que se puede localizar el *workflow*.
DeletePropertyWorkflow.	Elimina las propiedades de una instancia de *workflow* por las que se puede localizar el *workflow*.
FindPropertyByInstanceId.	Localiza las propiedades de una instancia de *workflow* dado su identificador.
FindProperty.	Localiza las propiedades de una instancia de *workflow* dado una cadena, en el cual alguna de estas propiedades contenga la cadena.

Tabla 16. Descripción del paquete Bison_State_PKG

Para ver una descripción más detallada de los procedimientos almacenados ver **¡Error! No se ncuentra el origen de la referencia.**

3.8 Conclusiones

- La definición de la arquitectura y de los patrones de diseño a utilizar permitió garantizar una mayor flexibilidad en la ejecución de la propuesta de solución, lográndose una independencia entre las capas de la misma.
- Se realizó el análisis y diseño de la solución, definiéndose las clases y servicios, así como la relación entre ellos.
- La identificación de las clases persistentes contribuyó al diseño del modelo de datos de la solución.

CAPÍTULO IV. IMPLEMENTACIÓN Y PRUEBAS

4.1 Introducción

Durante el presente capítulo se describe la implementación del *software* a través de los diagramas de componentes y los estándares de codificación. También se describe la manera en que se realizará el tratamiento de los errores. Se expone además la distribución de los nodos necesarios para el despliegue de aplicación, así como el diagrama de componentes. Finalmente se muestra la realización de pruebas unitarias y de carga con el objetivo de asegurar la calidad de la solución.

4.2 Estándares de codificación

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema. El mantenimiento del código es la facilidad con que el *software* puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores o mejorar el rendimiento. Aunque la legibilidad y el mantenimiento son el resultado de muchos factores, una faceta del desarrollo de *software* en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurar que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutina.

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.

Estilos para la capitalización

Se utilizaron 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. Se pueden utilizar los identificadores de Pascal *case* en caso de tres o más caracteres. Por ejemplo: *BackColor*

Camello

La primera letra en el identificador se pone en minúscula y la primera letra de cada subsiguiente palabra concatenada en mayúscula. Por ejemplo: *backColor*

Mayúscula

Todas las letras en el identificador se capitalizan. Esta convención se utiliza sólo para los identificadores que constan de dos o menos letras. Por ejemplo: *System.IO, System.Web.UI*

Sensibilidad a mayúsculas

Para evitar confusiones y garantizar la interoperabilidad entre lenguajes, se siguieron las siguientes reglas sobre el uso de mayúsculas y minúsculas:

- No utilizar nombres o identificadores que requieran ser *case sensitivity*.
- No utilizar *namespaces* que solo se diferencien con en el uso de las mayúsculas.
- No utilizar funciones con nombres de parámetros que se diferencian solo en el uso de las mayúsculas
- No utilizar *namespaces* con nombres de clases que se diferenciaran solo en el uso de las mayúsculas.
- No crear clases con propiedades que se diferencian solo en el uso de las mayúsculas.
- No crear clases con métodos que se diferencien solo en el uso de las mayúsculas.

Evitando confusión de nombre y tipo

Distintos lenguajes de programación usan diversos términos para declarar los principales tipos de identificadores. Los diseñadores de librerías de clases deben definir una terminología de lenguaje específica. Fueron utilizados nombres que describen a sus identificadores en vez de nombres que describen el tipo de identificador.

Con el objetivo de incrementar la legibilidad del código también se emplearon comentarios en todas las declaraciones de clases y funciones más complejas. También se organizó el código de forma estructurada, en bloques de código, para una mejor lectura del mismo.

4.3 Tratamiento de errores

Para la obtención de buenos resultados con el uso de la herramienta, es preciso informarle al desarrollador los diversos errores que cometa haciendo uso de la misma. En el *framework* se evitan, minimizan y tratan los posibles errores. Los mensajes de error que emite el sistema se muestran en un lenguaje de fácil comprensión para los usuarios.

4.4 Diagramas de componentes

En la figura 21 se muestra el diagrama de componentes del framework donde se relacionan los componentes y paquetes que brindan la arquitectura del mismo. En la solución se implementó la DLL <<Bison.dll>> que se encuentra estructurada en los siguientes paquetes Bison.Services, Bison.Hosting, Bison.Exception, Bison.Activities, Bison.Common y Bison.Entities.

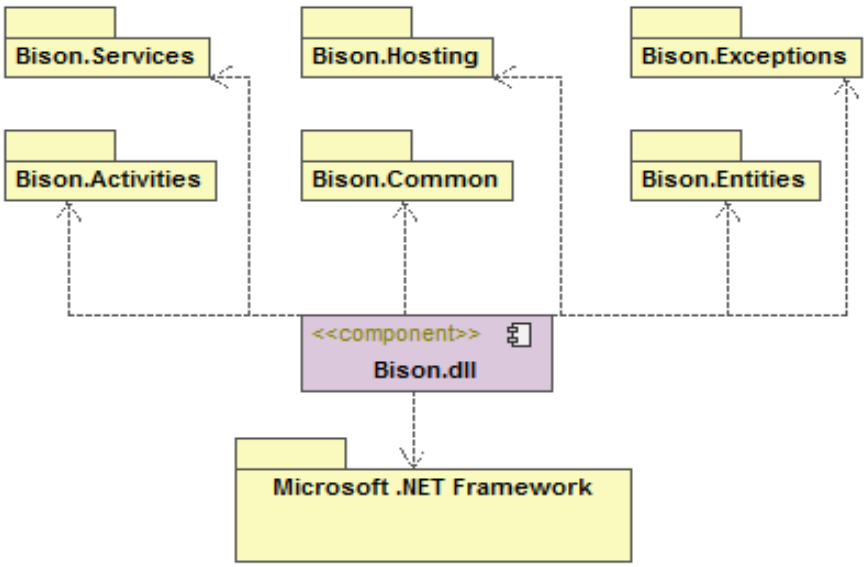


Figura 21. Diagrama de componentes del framework

4.4.1 Descripción del diagrama de componentes

El diagrama de componentes muestra cómo fue desarrollado el *framework*. El mismo está compuesto por la DLL <<Bison.dll>> que contiene todas las funcionalidades del *framework*. Este componente depende del *framework* .NET de *Microsoft* que contiene todas las librerías base que provee la tecnología .NET. A su vez el componente Bison.dll se encuentra estructurado en los siguientes 6 paquetes: Bison.*Common*, Bison.*Entities*, Bison.*Services*, Bison.*Activities*, Bison.*Hosting*, y Bison.*Exception*.

4.5 Diagramas de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema, graficando los nodos y conexiones necesarias para un funcionamiento exitoso de la aplicación. Para el despliegue de la aplicación se necesita un servidor de aplicaciones *Internet Information Services* 7. Es

necesario también un servidor de base de datos *Oracle 11g Enterprise Edition*. En la figura 22 se puede ver el diagrama de despliegue.

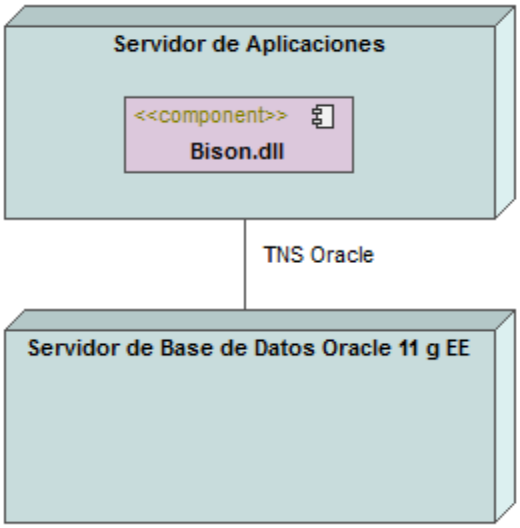


Figura 22. Diagrama de despliegue

4.6 Pruebas

Durante todas las etapas del desarrollo de *software* las pruebas son fundamentales, a partir de ellas es posible controlar que los productos cumplan requisitos mínimos de calidad, además de garantizar la calidad de estos productos. Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente. El objetivo de esta actividad es verificar que los requerimientos de *software* han sido satisfechos a través de pruebas unitarias y de carga.

4.6.1 Prueba de caja blanca

Este tipo de prueba 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. (22)

Según Pressman, mediante la prueba de la caja blanca el ingeniero del *software* puede obtener casos de prueba que cumplan una serie de requisitos:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.

- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello, que se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican al *software*, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende, una mayor calidad y confiabilidad.

Para la aplicación de este método es necesario conocer la estructura interna del *software* y son derivadas a partir de las especificaciones del diseño o el código. Se comprueban los caminos lógicos del *software* dado un código específico. Para la realización de las pruebas de caja blanca se utilizan las pruebas unitarias**.**

Pruebas unitarias: Son una forma de probar pequeñas e individuales porciones de código. A través de ellas se verifica que cierto módulo o funcionalidad se ejecuta dentro de los parámetros y especificaciones concretadas en los requisitos. Las mismas permiten detectar efectivamente la inyección de defectos durante fases sucesivas de desarrollo o mantenimiento. Las pruebas unitarias típicamente son automatizadas, pero pueden llevarse a cabo de forma manual. Cuando son automatizadas es buena práctica que forme parte del repositorio que contiene al código probado. (23)

La herramienta para pruebas *Team System* genera pruebas unitarias específicas para métodos, incluidos los métodos privados. Con el objetivo de realizar las pruebas de caja blanca a las funcionalidades del *framework* se implementaron las Pruebas Unitarias a todos los servicios para comprobar la validez del código y el acceso a datos. A continuación se muestran las pruebas realizadas al servicio *SearchWorkflowService* (ver **Figura 23. Código de la prueba unitaria aplicada a la funcionalidad *PropertyWorkflowById***) en la cual se comprobó la efectividad de este servicio (ver **Figura 24. Resultado de ejecutar el caso de prueba al servicio *SearchWorkflowService***).

```
/// <summary>
///A test for PropertyWorkflowByState
///</summary>
[TestMethod()]
public void PropertyWorkflowTest()
{
    var target = new SearchWorkflowService();
    string cadena = "CreacionCuenta";
    var property = new PropertyWorkflowLocation
    {instanceId = new Guid("2727f274-d5dc-439c-bd9b-b20051a0c9e2")};
    var propertyworkflowlist = new List<PropertyWorkflow>()
       {
           new PropertyWorkflow()
               {
                   Name = "Solicitud.Nombre",Path = "Solicitud.Nombre",
                   ValueInfo = "SolicitudTest"
               },
           new PropertyWorkflow()
               {
                   Name = "Solicitud.Usuario",Path = "Solicitud.Usuario",
                   ValueInfo = "rmachado"
               }
       };
    property.listProperty = propertyworkflowlist;
    var expected = new List<PropertyWorkflowLocation>() { property };
    List<PropertyWorkflowLocation> actual =
                     target.PropertyWorkflowByState(cadena);
    if (actual == null)
        Assert.Inconclusive("Inconclusive.");
    else
        Assert.AreEqual(expected[0], actual[0]);
}
```

Figura 23. Código de la prueba unitaria aplicada a la funcionalidad *PropertyWorkflowById*

Figura 24. Resultado de ejecutar el caso de prueba al servicio *SearchWorkflowService*

4.6.2 Pruebas de sistema

Se realizaron pruebas de carga a la aplicación, el objetivo fundamental de la realización de este tipo de pruebas es verificar el rendimiento las funcionalidades. Comparar valores respecto a diferentes

entradas, buscando diferencias que no deben existir y en caso de que existan documentarlas e informarlas para su posterior corrección.

Las pruebas de carga serán generadas con la herramienta para pruebas *Team System*. La variable de entrada que se utilizará toma un valor de 20 usuarios concurrentes para la realización de las pruebas de carga. Las variables de salida serán analizadas verificando los valores obtenidos como respuesta del sistema, comprobándolos con una variación de cantidad en las conexiones de usuarios. Esto se refiere a que la variable que tiene como valor 20, tomará primero valor 10 para comprobar el rendimiento con un número de peticiones media, logrando de esta forma valores para comparar a la hora de dar resultados.

Para la aplicación del procedimiento de pruebas de rendimiento diseñado al *software* se identifica un entorno con las siguientes características:

Hardware	Datos
Microprocesador	Intel(R) Xeon(R) CPU 3.00 GHz (Servidores) Intel(R) Core(TM) 2 Duo CPU E7300 a 2.66 GHz(PC Clientes)
Memoria RAM	2 GB (PC Clientes) 3 GB (Servidores)
Ancho de Banda	100 Mbps
Disco Duro	320 GB(PC Clientes) 80 GB (Servidores)

Tabla 17. Especificaciones de *hardware*

Software	Datos
Sistema Operativo	*Windows* XP *Services Pack* 3 (PC Cliente) *Windows* Server 2008 (Servidor de aplicaciones) Suse Linux *Enterprise Server* 10 (Servidor de base de datos)
Gestor de Base de Datos	Oracle 11g *Enterprise Edition*

Servidor de Aplicaciones	*Internet Information Services* 7
Antivirus	*Kaspersky Workstation* 6.0
Otras herramientas	*Microsoft Office* 2007
Software de prueba	*Visual Studio Team System* 2008 *Services Pack* 1

Tabla 18. Especificaciones de *software*

Tipo de prueba: Prueba de Carga

Funcionalidad a probar: Permitir buscar instancias de *workflow* según criterios.

Variables	Valores
Número de usuarios concurrentes [Número de hilos]	10
Tiempo entre conexión y conexión [Periodo de subida (en segundos)]	0
Número de iteraciones [Contador del bucle]	1

Tabla 19. Variables para el caso de prueba de carga 1

Variables	Valores
Número de usuarios concurrentes [Número de hilos]	20
Tiempo entre conexión y conexión [Periodo de subida (en segundos)]	0
Número de iteraciones [Contador del bucle]	1

Tabla 20. Variables para el caso de prueba de carga 1

Análisis e interpretación de los resultados

Es de gran importancia el análisis de los resultados de las pruebas aplicadas durante el desarrollo de *software* porque brinda la posibilidad de medir el porciento de cumplimiento de los requisitos especificados.

La aplicación pasó todas las pruebas definidas, no surgieron problemas con la conexión a la base de datos, los servidores fueron accesibles en todo momento, garantizando la calidad de los resultados. Los resultados para los casos de prueba expuestos anteriormente se muestran a continuación:

Tipo de prueba: Prueba de Carga

Resultados Generales:

Número de usuarios: 10

Variables	Valores
Cantidad de usuarios	10
Pruebas/Segundo	227
Pruebas fallidas	0
Ave. Tiempo por prueba (Segundo)	0.039

Tabla 21. Resultados del Caso de prueba de carga 1.

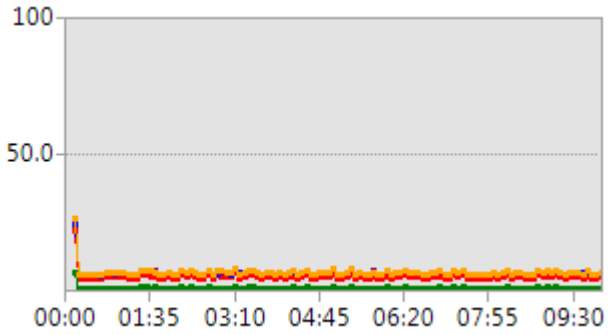
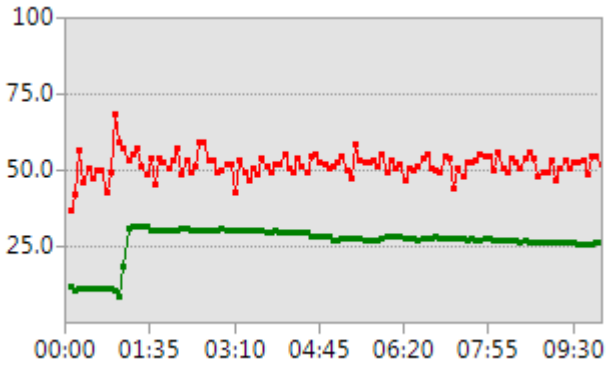


Figura 25. Tiempo de respuesta de la prueba de carga a las funcionalidades del servicio SearchWorkflowService con 10 usuarios

Contador	Instancia	Categoría	Computadora	Color	Mín.	Máx.	Ave
Ave. tiempo	PropertyWorkflowByIdTest	Prueba de	REISBEL		0.046	0.23	0.054

por prueba		carga					
Ave. tiempo por prueba	StateWorkflowByIdTest	Prueba de carga	REISBEL		0.011	0.069	0.014
Ave. tiempo por prueba	PropertyWorkflowTest	Prueba de carga	REISBEL		0.059	0.25	0.069
Ave. tiempo por prueba	PropertyWorkflowByStateTest1	Prueba de carga	REISBEL		0.059	0.27	0.069
Ave. tiempo por prueba	PropertyWorkflowByStateTest	Prueba de carga	REISBEL		0.059	0.27	0.069

Tabla 22. Tiempo de respuesta de la prueba de carga a las funcionalidades del servicio SearchWorkflowService con 10 usuarios



.

Figura 26. Uso de los recursos al ejecutar la prueba de carga a las funcionalidades del servicio SearchWorkflowService con 10 usuarios

Contador	Categoría	Computadora	Color	Rango	Mín.	Máx.	Ave
% Tiempo Procesador	Procesador	REISBEL		100	37.7	69.3	52.4
MBytes disponibles	Memoria	REISBEL		1,000	92	320	271

Tabla 23. Uso de los recursos al ejecutar la prueba de carga a las funcionalidades del servicio SearchWorkflowService con 10 usuarios

Al ejecutar la prueba de carga con 20 usuarios se tuvieron los siguientes resultados:

Número de usuarios: 20

Variables	Valores
Cantidad de usuarios	10
Pruebas/Segundo	227
Pruebas fallidas	0
Ave. Tiempo por prueba (Segundo)	0.075

Tabla 24. Resultados del caso de prueba de carga 2

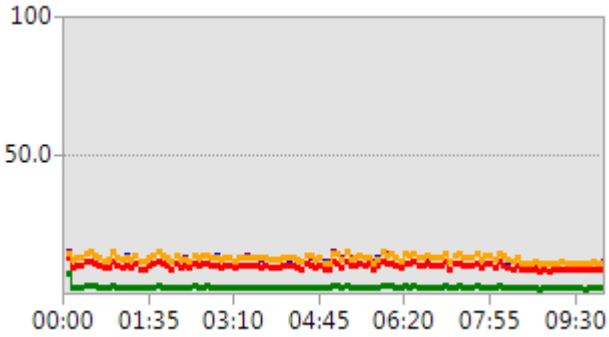


Figura 27. Tiempo de respuesta de la prueba de carga a las funcionalidades del servicio *SearchWorkflowService* con 20 usuarios

Contador	Instancia	Categoría	Computadora	Color	Mín.	Máx.	Ave
Ave. tiempo por prueba	PropertyWorkflowByIdTest	Prueba de carga	REISBEL		0.087	0.13	0.10
Ave. tiempo por prueba	StateWorkflowByIdTest	Prueba de carga	REISBEL		0.021	0.075	0.026
Ave. tiempo por prueba	PropertyWorkflowTest	Prueba de carga	REISBEL		0.11	0.16	0.13
Ave. tiempo por prueba	PropertyWorkflowByStateTest1	Prueba de carga	REISBEL		0.11	0.16	0.13
Ave. tiempo por prueba	PropertyWorkflowByStateTest	Prueba de carga	REISBEL		0.11	0.16	0.13

Tabla 25. Tiempo de respuesta de la prueba de carga a las funcionalidades del servicio *SearchWorkflowService* con 20 usuarios

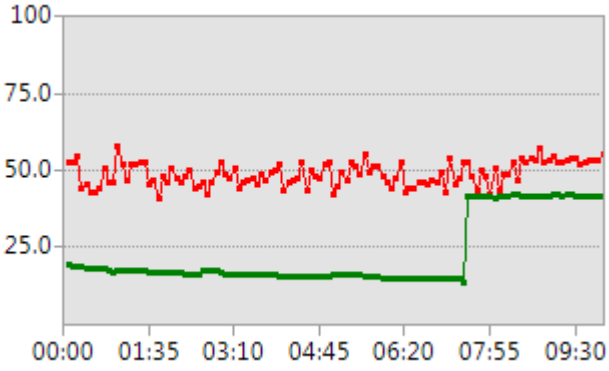


Figura 28. Uso de los recursos al ejecutar la prueba de carga a las funcionalidades del servicio SearchWorkflowService con 20 usuarios

Contador	Instancia	Categoría	Computadora	Color	Rango	Mín.	Máx.	Ave
% Tiempo Procesador	_Total	Procesador	REISBEL		100	41.7	58.6	49.5
MBytes disponibles	-	Memoria	REISBEL		1,000	140	427	233

Tabla 26. Uso de los recursos al ejecutar la prueba de carga a las funcionalidades del servicio *SearchWorkflowService* con 20 usuarios

Durante la ejecución de la prueba el *CPU* se mantuvo entre el 40 y el 60 % de utilización, la memoria física estuvo por debajo del 50 % y la taza de errores se mantuvo en 0. Analizando los valores de cantidad de peticiones y rendimiento se puede concluir que el sistema responde satisfactoriamente, para 20 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 10 usuarios concurrentes. Luego del análisis de los valores, se puede llegar a la conclusión de que las funcionalidades del servicio *SearchWorkflowService* cumplen con los requerimientos establecidos.

4.7 Conclusiones

- El uso de los estándares de codificación garantizó que la implementación realizada fuera de manera organizada, permitiendo un mayor entendimiento de la misma.
- Se desarrolló una primera versión del sistema, definiéndose los componentes que intervienen en la solución final de la aplicación así como la relación de los nodos físicos que la componen.
- Las pruebas realizadas validaron la calidad de las funcionalidades de la solución.

CONCLUSIONES GENERALES

- El modelado del dominio posibilitó el diseño de la propuesta de solución.
- El estudio de las tecnologías permitió escoger las herramientas, la metodología y los lenguajes más adecuados para el desarrollo de la solución.
- Se definieron los requerimientos funcionales que el sistema debe cumplir.
- La definición de la arquitectura y de los patrones de diseño a utilizar permitió garantizar una mayor flexibilidad en la ejecución de la propuesta de solución.
- Se realizó el análisis y diseño de la solución, definiéndose las clases y servicios, así como la relación entre ellos.
- La identificación de las clases persistentes contribuyó al diseño del modelo de datos de la solución.
- Se usaron estándares de codificación que garantizaron que la implementación realizada fuera de manera organizada, permitiendo un mayor entendimiento de la misma.
- Se desarrolló una primera versión del sistema, definiéndose los componentes que intervienen en la solución final de la aplicación así como la relación de los nodos físicos que la componen.
- Se realizaron pruebas que permitieron validar la calidad de las funcionalidades de la solución.

RECOMENDACIONES

Como parte del proceso de mejora continua del desarrollo de *software* se proponen las siguientes recomendaciones que tributarían a un producto de mayor calidad. Para el desarrollo de futuras versiones del *framework* deben tenerse en cuenta los siguientes aspectos:

- Implementar nuevas funcionalidades al servicio de trazas.
- Desarrollar un servicio de consulta a las trazas.
- Desarrollar un componente con interfaz gráfica que facilite la configuración del *framework* debido a que actualmente este procedimiento se realiza manualmente a través del fichero de configuración de la aplicación en desarrollo, esta vía provoca inconvenientes para los desarrolladores, pues los errores derivados de una mala configuración retrasan el desarrollo.
- Desarrollo de un modulo de gestión de instancias de *workflow*.
- Someter la solución a un número mayor de pruebas simulando ambientes reales con el fin de lograr niveles superiores de optimización.

BIBLIOGRAFÍA

1. Coalición para el Manejo de Workflows. [En línea] http://www.wfmc.org.

2. Aalast W., Hee K. *Workflow Management Models, Methods, and Systems MI TPress.* 2002.

3. zur Muehlen, M. *Workflow-based Process Controlling: Foundation, Design and Application of Workflow-driven Process Information Systems. Logos.* Berlin : s.n., 2004.

4. Web de OMG. [En línea] http://www.omg.org.

5. Web de OASIS. [En línea] http://www.oasis-open.org.

6. [En línea] http://wwww.Workflowpattrens.com.

7. Fischer, L. *Workflow Handbook, Workflow Management Coalition.Future Strategies, Lighthouse Point.* 2005.

8. Proyecto Yawl. [En línea] http://www.sourceforge.net/projects/yawl.

9. Sistema manejador de workflow jBPM. [En línea] http://www.jbpm.org.

10. *MSDN Library.* [En línea] http://www.msdn.microsoft.com/en-us.

11. G.Kappel, S. Rausch-Schott, and W. Resschitzegger. *A Framework for Workflow Management Systems Based on Objects.* 1998.

12. G. Vossien, M. Weske. *The Wasa2 Object-Oriented Workflow Management System. University of Münster, Germany.* 1999.

13. Mylopoulos., A. Gal and J. *Supporting Distributed Autonomous Information Services Using Coordination.International Journal of Cooperative Information.*

14. Bob Bryla, Kevin Loney. *Oracle Database 11g DBA Handbook (Osborne ORACLE Press Series).*

15. oracle. [En línea] http://www.oracle.com/technology/products/database/oracle11g.

16. Altova. [En línea] http://www.altova.com/list/xmlschema-dev/201001.

17. Oracle Data Provider. [En línea] http://www.oracle.com/technology/obe/hol08/dotnet.

18. PL / SQL. [En línea] http://www.oracle.com/technology/tech/pl_sql.

19. *Flujo de trabajo Captura de requisitos.* Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.

20. Jacobson, Ivar. *El proceso unificado de desarrollo de Software.* España : Addison-Wesley Iberoamericana España, S.A., 2000.

21. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley Professional; illustrated edition edition (November 10, 1994).

22. LEON, F. *Ingeniería del Software.* 2006.

23. Beck, Kent. XProgramming. [En línea] [Citado el: 3 de 5 de 2010.] http://www.xprogramming.com/testfram.htm.

24. Support Microsoft. [En línea] [Citado el: 19 de febrero de 2010.] http://support.microsoft.com/.

25. SUN. [En línea] [Citado el: 19 de febrero de 2010.] http://java.sun.com.

26. MSDN. [En línea] [Citado el: 19 de febrero de 2010.] http://msdn.microsoft.com/.

27. Microsoft. [En línea] [Citado el: 19 de febrero de 2010.] http://www.microsoft.com/downloads/details.aspx?familyid=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=es.

28. UCI. *Documentos Proyecto Identidad Cuba.* 2009.

29. Mozilla developer center. [En línea] [Citado el: 2 de febrero de 2010.] https://developer.mozilla.org/.

30. msdn. [En línea] http://msdn.microsoft.com/en-us/library/aa753587%28VS.85%29.aspx.

31. MSDN. [En línea] http://msdn.microsoft.com/es-es/library/4w3ex9c2.aspx.

32. Microsoft TechNet. Microsoft TechNet. [En línea] [Citado el: 24 de febrero de 2010.] http://technet.microsoft.com/en-us/windowsserver/sharepoint/bb684453.aspx..

33. *Framework para integrar el acceso a dispositivos.* Campo, Adonis C. Legón. Ciudad de la Habana : s.n., 2007.

34. bundesdruckerei. *bundesdruckerei.* [En línea] 1 de marzo de 2010. http://www.bundesdruckerei.de/.

35. SUN. *SUN.* [En línea] 3 de Marzo de 2010. http://java.sun.com/javacard/.

36. Digital Persona. *Digital Persona.* [En línea] 20 de marzo de 2010. http://www.digitalpersona.com/.

37. Mozilla Addons. *Mozilla Addons.* [En línea] 16 de marzo de 2010. https://addons.mozilla.org/.

38. axiompass. *axiompass.* [En línea] 5 de marzo de 2010. http://www.axiompass.com/AxiomDMS.aspx.

39. sagem. *sagem.* [En línea] 15 de marzo de 2010. http://www.sagem.com/index.php?id=774&L=8.

40. ECMA. *ECMA.* [En línea] 3 de abril de 2010. http://www.ecma-international.org/.

41. IEE. [En línea] 1990 . http://ieeexplore.ieee.org/.

42. World Wide Web Consortium. *World Wide Web Consortium.* [En línea] 9 de mayo de 2010. http://www.w3.org.

43. Ingeniería de software orientada a objetos con Java e Internet. [aut. libro] A WEITZENFELD. 2006.

44. Vegas, Sira, Juristo, Natalia y Moreno, Ana M. Técnicas de Evaluación de Software. [En línea] http://is.ls.fi.upm.es/docencia/erdsi/Documentacion_Evaluacion_7.pdf.

45. MSDN. [En línea] noviembre de 2007. http://msdn.microsoft.com/es-es/library/bb397926(VS.90).aspx.

46. W3e. [En línea] 11 de Mayo de 2010. http://www.w3.org/XML/.

47. W3e. [En línea] 11 de mayo de 2010. http://www.w3c.es/divulgacion/guiasbreves/XHTML.

48. W3e. [En línea] 5 de Mayo de 2010. http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo.

GLOSARIO

A

ADA
Lenguaje de programación orientado a objetos, fuertemente tipado y concurrente., 26

API
Conjunto de funciones residentes en sistema operativo en que permiten que una aplicación se ejecute., 19

C

Capability Maturity Model Integration
modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software., 26

D

Debugging
es el proceso de identificar y corregir errores de programación., 21

E

Escalabilidad
propiedad deseable de un sistema que indica su habilidad para extender el margen de operaciones sin perder calidad., 16

F

Framework
Conjunto de clases modeladas de forma general para resolver problemas relacionados en un contexto específico., V

I

Interoperabilidad
Cooperación de sistemas diferentes que colaboran para llevar a cabo un objetivo., 11

M

Microsoft
empresa multinacional estadounidense, fundada en 1975 por Bill Gates y Paul Allen. Dedicada al sector de la informática, con sede en Redmond, Washington, Estados Unidos., V

Microsoft Solution Framework
es un conjunto de principios, modelos, disciplinas, conceptos y directrices para el desarrollo de soluciones informáticas de Microsoft., 26

Middleware
Es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas., 19

Motor de *workflow*
Software que proporciona el entorno de ejecución para una instancia de proceso., 13

P

Proceso de negocio
Conjunto de uno o más procedimientos o actividades, que en conjunto alcanzan un objetivo de negocio, siempre dentro del contexto de la organización en que estén definidos., 8

R

Redes de Petri
Permiten expresar eventos concurrentes en sistema de evolución en paralelo., 18

Reingeniería
La re-concepción y el rediseño de los procesos de negocios., 12

S

Stakeholders
Persona que es afectada por las actividades de una organización., 9

T

Triggers
Procedimientos que se ejecutan cuando se cumple una condición al realizar una operación en base de datos., 22

W

Windows Workflow Foundation
Tecnología extensible para el desarrollo de soluciones de workflow, permite la automatización de tareas, actividades y procesos permitiendo utilizar transacciones integradas utilizando workflow., 2

Workflow Management Coalition
Es la coalición o grupo de trabajo que se encarga de estandarizar los procedimientos para el diseño, desarrollo e implantación de una herramienta de workflow en una organización. Se dedica un apartado completo a ella y el significado de las siglas se puede consultar en el apartado., 8

Workflow
Automatización de un proceso de negocio, en su totalidad o en parte, en el que intervienen diferentes participantes entre los que circulan tareas, información, documentos y datos, de acuerdo con reglas preestablecidas para ello., V

X

XML
Metalenguaje extensible de etiquetas., 13