Universidad de las Ciencias Informáticas

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

***Título**: Diseño e Implementación del Módulo Buscador Genérico para el proyecto SAGEB.*

***Autores**: Daynelis Cruz Castro*

Eduardo Juárez Hernández

***Tutor**: Ing. Javier Martínez Muñoz*

Ciudad De La Habana. Junio, 2010
"Año 52 de la Revolución"

"Sean capaces siempre de sentir, en lo más hondo, cualquier injusticia realizada contra cualquiera, en cualquier parte del mundo. Es la cualidad más linda del revolucionario."

Declaración de autoría.

Declaramos que somos los únicos autores del trabajo titulado:

Diseño e Implementación del Módulo Buscador Genérico para el proyecto SAGEB.

Y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los___ días del mes de _____del año 2010.

Daynelis Cruz Castro	Eduardo Juárez Hernández	Javier Martínez Muñoz
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

Dedicatoria y Agradecimientos

Agradecimientos

A todas las personas que de una forma u otra nos ayudaron y guiaron en la realización de este trabajo, apoyándonos incondicionalmente, gracias a todos por estar ahí en los momentos más difíciles, a nuestro tutor Javier quien se preocupó y nos guió siempre, a nuestros familiares y personas queridas, a todos nuestros amigos con los que hemos compartido todos estos años, gracias por todos esos momentos lindos.

Dedicatoria

Con un profundo cariño, dedico el fruto de muchos de mis sueños que gracias a estas personas se convirtieron en realidad.

A mis padres, que siempre desde pequeña me apoyaron en todo, por esforzarse tanto como yo para obtener mis metas, por toda la confianza que depositaron en mí.

A mis hermanos que siempre han estado ahí presente en todos los momentos de mi vida buenos y malos.

A mi novio que me ha acompañado todos estos años, apoyándome cada día, brindándome cada mañana una sonrisa, es bueno saber que tenemos una persona al lado que nos quiere.

A toda mi familia en general gracias por todo su apoyo y comprensión.

A todos mis amigos, los que de una forma u otra me enseñaron tantas cosas de la vida, gracias por compartir momentos tan lindos, por ayudarme siempre, los extrañaré mucho a todos.

Dayne

Le dedico estos resultados obtenidos a todas aquellas personas que me han acompañado en el transcurso de mi vida.

A mis padres por ser tan atentos y especiales conmigo en todo momento.

A mis hermanos por ser un ejemplo para mí.

A dos mujeres que ocupan una parte importante en mi vida, Mairin y mi niña preciosa Melany que es mi inspiración.

A todos mis amigos que han compartido todos estos años conmigo, ayudándome y acompañándome incondicionalmente.

A mis compañeros de proyecto por los ratos buenos que pasamos, gracias a todos.

Eduardo

Resumen

Los bancos desempeñan un papel esencial en la asignación de recursos económicos. Su participación en la provisión de fondos, y por ende en el estímulo al desarrollo económico, es fundamental.

La política económica cubana ha ido evolucionando sustancialmente hacia un proceso de ajuste conforme a las nuevas circunstancias en que se desarrollan sus relaciones y en cuyo contexto, la banca juega un nuevo e importante papel. Con el desarrollo de las tecnologías, la informatización de los sistemas bancarios forma parte del avance tecnológico que precisa nuestro país, debido a que estas instituciones son de vital importancia pues deben ser capaces de brindar una mejor respuesta a los clientes que soliciten un determinado servicio.

Debido a limitaciones que enfrenta el sistema del Banco Nacional de Cuba (BNC) se determina la necesidad de realizar un nuevo sistema informático que cubra con la demanda de operaciones que se ejecutan. En el cual se hace necesario desarrollar el módulo Buscador Genérico para facilitarles a los usuarios la búsqueda de cualquier tipo de información.

Para materializar la idea anteriormente expresada fue llevada a cabo la presente investigación, durante la cual se hizo un estudio del estado del arte de los buscadores existentes. Además se realizó un estudio detallado de las tendencias tecnológicas actuales, donde se seleccionaron los lenguajes, la metodología de desarrollo y las herramientas a utilizar para el diseño e implementación de un sistema que satisfaga las necesidades del cliente. Como resultado, se diseñaron, implementaron y probaron las funcionalidades que debería cumplir el producto, el cual posibilitará al país contar con la automatización de este proceso, y por tanto facilitará el trabajo en los sistemas bancarios.

Palabras Claves.

Sistemas bancarios, buscador, tecnologías, sistema informático

Índice

Índice

Índice

Introducción

Al buscar una constante en nuestro tiempo, pareciera que la única que se consigue es el cambio. Y es que el proceso de liberalización financiera, abrió las puertas al incremento de los flujos de capital, que favorecido por los avances tecnológicos en la telemática, permitió el crecimiento y transnacionalización del sector convirtiéndose en una poderosa fuerza de convergencia de alcance internacional que le imprimiría al mundo financiero las características de competitividad, interdependencia e instantaneidad que hoy día posee. Los bancos desempeñan un papel esencial en la asignación de recursos económicos. Su participación en la provisión de fondos y por ende en el estímulo al desarrollo económico, es fundamental.

El crecimiento del comercio internacional durante las pasadas décadas ha evolucionado de la mano del crecimiento de una banca multinacional.

Se puede decir que los antecedentes de la reforma bancaria actual datan de la década de los ’80. A mediados de este período se comenzaba a trabajar en el perfeccionamiento del sistema económico y se introducían cambios en el sistema bancario para hacerlo más eficiente. Ya en estos años, el país se enfrentaba a una situación financiera difícil por la contracción de las facilidades Crediticias otorgadas por países de economía de mercado.

La política económica cubana ha ido evolucionando sustancialmente hacia un proceso de ajuste conforme a las nuevas circunstancias en que se desarrollan sus relaciones y en cuyo contexto, la banca juega un nuevo e importante papel. Con el desarrollo de las tecnologías, la informatización de los sistemas bancarios forma parte del avance tecnológico que precisa el país debido a que estas instituciones son de vital importancia pues deben ser capaces de brindar una mejor respuesta a los clientes que soliciten un determinado servicio.

En busca de este perfeccionamiento el sistema bancario cubano, se trazó toda una estrategia, dentro de la cual ha constituido una medida de importancia capital la separación de las funciones de la banca central y banca comercial, lo cual ha hecho que en la actualidad el sistema bancario esté constituido por dos niveles: el Banco Central y un grupo de bancos e instituciones financieras no bancarias.

Introducción

El Sistema Bancario y Financiero Nacional está compuesto por el Banco Central de Cuba, ocho bancos comerciales, catorce instituciones financieras no bancarias, doce oficinas de representación de bancos extranjeros y cuatro oficinas de representación de instituciones financieras no bancarias.

El BNC es un banco comercial autorizado para ejercer funciones de banca universal y además tiene la función de registro, control, servicio y atención de la deuda externa contraída por el Estado y el propio banco.

Situación problemática:

En el BNC se encuentra implantado el Sistema Automatizado para la Banca Internacional de Comercio (SABIC), dicho sistema posee una tecnología obsoleta ya que es ejecutado en MS-DOS, el cual es un sistema operativo que ha quedado en desuso, además se trabaja sobre varias plataformas y esto resulta extremadamente difícil, convirtiéndose este en un sistema decadente propiciando que no pueda cubrir en su totalidad las exigencias de los procesos que se ejecutan dentro de los bancos cubanos que lo utilizan.

Él sistema no garantiza un mecanismo que le permita a los usuarios buscar cualquier tipo de información en un momento determinado, por lo que este proceso de búsqueda se realiza manualmente y esto hace que sea más lento.

Debido a las limitaciones que enfrenta el sistema del BNC se determina la necesidad de realizar un nuevo sistema informático que cubra con la demanda de operaciones que se ejecutan. Asignándole a la UCI el desarrollo de este sistema conformando a su vez un proyecto productivo. En el cual se hace necesario desarrollar el módulo Buscador Genérico para facilitarles a los usuarios la búsqueda de cualquier tipo de información. Por todo lo planteado anteriormente se define como **problema a resolver:** ¿Cómo automatizar los procesos de búsqueda de información en el proyecto SAGEB? De ahí que el **objeto de estudio** es**:** Buscadores genéricos de información.Dando lugar al **campo de acción:** Búsquedas de información en los subsistemas del proyecto SAGEB. Se redactó el **objetivo general:** Desarrollar el módulo Buscador Genérico que permita realizar búsquedas de información en el proyecto SAGEB, el cual se desglosa en varios **objetivos específicos:**

- ✓ Realizar una búsqueda sobre la existencia de Buscadores genéricos de información en Cuba y el mundo.
- ✓ Determinar el marco teórico de la investigación.

- ✓ Diseñar un Buscador genérico de información.
- ✓ Implementar un Buscador genérico de información.

Para darle cumplimiento a estos objetivos específicos se elaboró un conjunto de **Tareas de la investigación:**

- ✓ Caracterizar los sistemas automatizados nacionales e internacionales que soportan la realización de búsquedas genéricas.
- ✓ Caracterizar las herramientas de desarrollo a utilizar durante el diseño del buscador genérico del proyecto SAGEB.
- ✓ Caracterizar los procesos relacionados con las búsquedas en los subsistemas del proyecto SAGEB.
- ✓ Realizar el modelo de diseño del buscador genérico del proyecto SAGEB.
- ✓ Realizar modelo de componentes del buscador genérico del proyecto SAGEB.

Posibles resultados:
Diseño e implementación de una solución general que permita realizar búsquedas genéricas de la información que gestionan los subsistemas en el proyecto SAGEB.

La presente investigación está estructurada en 3 capítulos, a continuación se explica un resumen de cada uno:

Capítulo #1: Fundamentación Teórica:

En este capítulo se presentan temas que integran la fundamentación teórica de la investigación a realizar, o sea el estado del arte, para lo se hace necesario el análisis y revisión de las fuentes bibliográficas relacionadas con el tema en cuestión, además se hace un análisis sobre las principales tecnologías, metodologías y una comparación entre las herramientas que se utilizan en la actualidad para este tipo de sistemas.

Capítulo #2: Diseño e Implementación del Buscador Genérico de Información.

En el presente capítulo se hace una descripción de la solución propuesta. Se muestran los artefactos generados en el diseño de la propuesta de solución, además se argumentan algunas de las terminologías que se utilizan. Se abarca todo lo concerniente con la implementación del sistema.

Capítulo #3: Prueba del Buscador Genérico de Información.

En este capítulo se aborda todo lo relacionado con el proceso exhaustivo de prueba y refinamiento. Detectando de esta forma los posibles errores existentes en la aplicación con la finalidad de garantizar un buen funcionamiento.

Capítulo 1: Fundamentación Teórica

Introducción

En el presente capítulo se abordará la fundamentación teórica para el desarrollo del módulo Buscador Genérico el cual permitirá realizar búsquedas de información en el proyecto SAGEB. Para ello se realizará un minucioso análisis de los sistemas similares existentes tanto internacionales como nacionales relacionados con el campo de acción, un estudio detallado de las metodologías de desarrollo de software, de los lenguajes de programación, las herramientas y las plataformas de desarrollo que las soportan.

1.1 Buscadores genéricos de información

Las nuevas tecnologías marcan una pauta en el desarrollo de buscadores, haciendo más rápido y sencillo el proceso de búsqueda de información a los usuarios. Los buscadores genéricos pueden ser utilizados en diferentes instituciones donde se maneja gran cantidad de información con el objetivo de agilizar los procesos de búsqueda, los cuales presentan una nueva perspectiva hacia las tecnologías actuales. Se han desarrollado varios buscadores genéricos para dar solución a una serie de dificultades y problemas que se presentan a la hora de acceder a una determinada información. En el estudio realizado se muestran algunos de estos buscadores genéricos.

1.1.1 Buscador genérico similar a nivel internacional

Trac es una web basada en software de gestión de proyectos y de errores. Proporciona una Wiki integrada, una interfaz con los sistemas de control de versiones, y formas convenientes para estar al día de los acontecimientos y los cambios dentro de un proyecto.

El sistema trac permite ser usado por cualquier persona, posee una interfaz sencilla, fácil de utilizar y manejar por parte del usuario, diseñado para mejorar y agilizar todo lo relacionado con las búsquedas de información. Trac permite código wiki en las descripciones de los mensajes de emisión y creación de vínculos y referencias sin fisuras entre las tareas de cambios, los archivos y las páginas wiki.

Permite realizar una búsqueda personalizada así como una búsqueda filtrada, además brinda la posibilidad de escoger en un checkbox las columnas resultantes que deseas y agrupar los resultados por categorías.

1.1.2 Buscadores genéricos similares a nivel nacional

En la universidad de las ciencias informáticas se encuentran desarrollados algunos buscadores genéricos, todos con un único fin, el de mejorar los procesos de búsqueda de información, aunque estos cumplen con sus expectativas dentro de la universidad no contienen en su totalidad todas las funcionalidades que complementan un buscador, entre estos se encuentran.

GUÍATELEFÓNICA: Es la web oficial de teléfonos UCI.

La guía telefónica UCI hace uso de plantillas web para mejorar la apariencia de las páginas. Presenta una interfaz fácil de utilizar y de manejar para los usuarios, la cual evita recargar las páginas con textos, imágenes o gráficos. Además establece mecanismos de barrido visual para el contenido de la página, distribuyendo los elementos de información y navegación según su importancia en zonas de mayor o menor jerarquía visual. Las zonas superiores de la interfaz poseen mayor jerarquía visual que las inferiores.

El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora .Ofrece señalizaciones que agilizan el aprendizaje del usuario a trabajar con el sistema. Presenta tiempos de respuestas rápidos al igual que la velocidad de procesamiento de la información, no mayor de 5 segundos para las actualizaciones.

La guía telefónica UCI implementa funcionalidades que garantizan la obtención del resultado deseado por el usuario final. Entre las principales opciones que ofrece se encuentran: Búsqueda General, Búsqueda Avanzada, Páginas Amarillas y Urgencias. Posee además un conjunto de propiedades que garantizan las búsquedas a los usuarios como son: Paginado, Ordenamiento, Búsqueda a Texto Completo y Búsqueda por alias. La aplicación cuenta con una bibliografía de apoyo al trabajo con la misma, donde se describe cada funcionalidad del sistema.

DIRECTORIO: Es el servicio de búsqueda más completo que presenta la UCI, pues permite el acceso a una amplia gama de información.

Complementa una interfaz sencilla con un diseño práctico destinado a lograr procesos de búsquedas eficientes, con tiempos de respuesta y procesamiento mínimos (7 segundos), que puedan ser utilizados fácilmente por usuarios normales con un conocimiento mínimo del sistema.

Permite realizar una búsqueda general, así como una búsqueda filtrada de personas, entre los criterios por los que se puede filtrar la información se encuentran: nombre, apellidos, usuario, credencial, categoría, área, cargo, esto asegura resultados eficientes y concretos dependiendo del cúmulo inicial de información con que cuente el usuario que va a realizar la búsqueda.

Permite además la posibilidad de realizar búsquedas en la guía telefónica de la UCI, así como la búsqueda de sitios web, que también puede ser filtrada por categoría y temática.

1.2 Modelo de Desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software.

RUP (Proceso Unificado de Desarrollo) es un proceso de desarrollo de software que captura las mejores prácticas del conocimiento de líderes en ingeniería de software y proporciona a los equipos de desarrollo guías, estándares y recomendaciones para la construcción de software de alta calidad. Las mejores prácticas de desarrollo de software están documentadas como principios claves. (1)

Para el desarrollo del Buscador se decidió utilizar RUP por las siguientes razones:

- Está concebido para proyectos y equipos de trabajos grandes (Esta es la principal característica del proyecto en que se desarrolla el presente trabajo).
- Exige una documentación completa de los artefactos a realizar.
- Existe una formalidad prefijada.
- Es un proceso mucho más controlado, con numerosas políticas y normas.
- El proyecto está formado por grandes grupos y distribuidos por módulos.

RUP se caracteriza por ser un:

- **Proceso dirigido por Casos de Uso.**

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema, también guían su diseño, implementación y prueba, constituyen un elemento integrador y una guía del trabajo.

- **Proceso centrado en la arquitectura.**

La arquitectura de un sistema es la estructura u organización de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios). La arquitectura se ve influenciada por la plataforma de software, sistema operativo, gestor de bases de datos y protocolos. RUP presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

- **Proceso iterativo e incremental.**

RUP se propone como estrategia, tener un proceso iterativo e incremental donde el trabajo se divida en partes más pequeñas o mini proyectos. Permitiendo un equilibrio entre Casos de Uso y arquitectura durante cada mini proyecto, y así para todo el proceso de desarrollo.RUP ayuda a mejorar la productividad del equipo de trabajo, definiendo claramente sus actividades, roles y responsabilidades

1.2.1 Flujos de trabajo.

Acorde a la metodología RUP, el presente trabajo se va a realizar enmarcado en los flujos de trabajo de: **Análisis y Diseño** (Flujo del cual solo se realizó el diseño) **Implementación** y **Prueba.**

Diseño: El diseño es un refinamiento del análisis debe ser suficiente para que el sistema pueda ser implementado sin imprecisiones. Contribuirá a obtener una arquitectura estable, sólida y a crear un plano del modelo de implementación capturando los requisitos o subsistemas individuales, interfaces y clases.

Implementación: En este flujo de trabajo se implementan las clases que se obtuvieron en la fase de análisis y diseño en ficheros fuente, binarios y ejecutables. Además se deben hacer los test de unidad donde cada implementador es responsable de probar los componentes desarrollados e integrar estos en un sistema ejecutable. El resultado final de este flujo de trabajo es un sistema ejecutable.

Prueba: En este flujo de trabajo se realizan métodos de prueba ya sea por caja negra donde se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, o por caja blanca que se basa en el

minucioso examen de los detalles procedimentales. Donde se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones.

1.3 Lenguaje de Desarrollo y Modelado

1.3.1 Lenguaje de Modelado Unificado.

RUP utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) como leguaje de modelado, lenguaje que se utilizará para el modelado del Buscador.

UML es un lenguaje estándar para especificar, visualizar, construir y documentar todos los artefactos de un sistema de software. (1)

El objetivo del uso de UML es capturar las partes esenciales del sistema a desarrollar. Su uso permite combinar elementos gráficos mediante reglas para finalmente obtener un modelado visual independiente del lenguaje de implementación dando la posibilidad de que los diseños realizados utilizando UML se puedan implementar en cualquier lenguaje de programación que soporte las posibilidades de UML. (Principalmente lenguajes de programación OO). Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Es un método formal de modelado que aporta varias ventajas, tales como:

- Permite expresar el sistema de modo gráfico, de forma tal que otro pueda entender.
- Proporciona un vocabulario y reglas para permitir la comunicación y se centra en la representación gráfica del sistema.

1.3.2 Lenguaje de programación

Un lenguaje de programación está constituido por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones, que se pone a disposición del programador para que éste pueda comunicarse con los dispositivos hardware y software existentes, por lo que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Los lenguajes de programación pueden ser clasificados atendiendo a varios criterios, los cuales pueden ser: su nivel de abstracción o paradigma.

Como lenguaje de programación, se decidió utilizar Java por ser un lenguaje multiplataforma, ya que posibilita al software correr en cualquier tipo de sistema operativo, es una plataforma de software libre; ósea contiene código abierto.

1.3.2.1 Java

Java es un lenguaje de programación desarrollado por Sun Microsystems Inc. a principios de los años 90. Este lenguaje toma mucha de su sintaxis de C y C++, aunque tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación de punteros o memoria. Aunque a diferencia de éste, que combina la sintaxis para programación genérica, estructurada y orientada a objetos, Java fue construido desde el principio para ser completamente orientado a objetos. Además de ser un lenguaje orientado a objetos, otra cualidad muy importante es la independencia de la plataforma, esto significa que programas escritos en el lenguaje Java pueden ser ejecutados de igual forma en cualquier tipo de hardware.

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para todo tipo de aplicaciones. Java ha sido probado, mejorado y ampliado por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo.

1.4 Patrones de diseño

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables. Han revolucionado el diseño orientado a objetos.(1)

1.4.1 Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un patrón de diseño que considera dividir una aplicación en tres módulos claramente identificables y con funcionalidad bien definida: El Modelo, las Vistas y el Controlador. El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar, sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos

que hacen que esos datos estén dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación. El modelo desconoce la existencia de las vistas y del controlador. Ese enfoque suena interesante, pero en la práctica no es aplicable pues deben existir interfaces que permitan a los módulos comunicarse entre sí, por lo que se sugiere que el modelo en realidad esté formado por dos submódulos: El modelo del dominio y el modelo de la aplicación.

Modelo del dominio

Se podría decir que el modelo del dominio (o el modelo propiamente dicho) es el conjunto de clases que un ingeniero de software modela al analizar el problema que desea resolver; así, pertenecerían al modelo del dominio: El cliente, la factura, la temperatura, la hora, etc. El modelo del dominio no debería tener relación con nada externo a la información que contiene.

Modelo de la aplicación

El modelo de la aplicación es un conjunto de clases que se relacionan con el modelo del dominio, que tienen conocimiento de las vistas y que implementan los mecanismos necesarios para notificar a estas últimas sobre los cambios que se pudieran dar en el modelo del dominio. El modelo de la aplicación es llamado también coordinador de la aplicación. (2)

Las vistas

Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo, así por ejemplo, se puede tener una vista mostrando la hora del sistema como un reloj analógico y otra vista mostrando la misma información como un reloj digital.

Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación.(3)

El controlador

El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador

tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. (4)

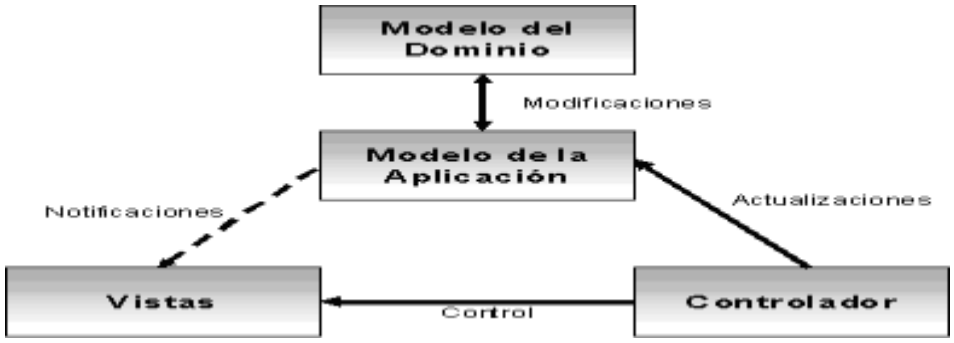

Fig.1: Modelo Vista Controlador

Desarrollar una aplicación siguiendo este patrón de diseño tiene entre sus ventajas:

- ✓ La aplicación está implementada modularmente.
- ✓ Sus vistas muestran información actualizada siempre.
- ✓ El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.

1.4.2 Patrones GRASP.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Constituyen un apoyo para entender el diseño y aplica el razonamiento para el diseño de una forma sistemática, racional y aplicable. Los 5 patrones básicos se refieren a cuestiones y aspectos fundamentales del diseño, algunos de estos patrones utilizados en este trabajo son:

Experto: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Creador: Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:
B agrega los objetos A.

B contiene los objetos A.
B registra las instancias de los objetos A.
B utiliza específicamente los objetos A
B contiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).
B es un creador de los objetos A. Si existe más de una opción, prefiera la clase B que agregue o contenga a la clase A.
Alta Cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande.

Bajo Acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento. Las clases deben comunicarse con un número pequeño de clases tanto como sea posible.

Controlador: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Utilice la misma clase de controlador con todos los eventos del sistema en el mismo caso de uso. (5)

1.4.3 Patrones GoF.

Los patrones GoF se clasifican en 3 grandes categorías basadas en su **propósito**: creacionales, estructurales y de comportamiento.
Creacionales: Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
Estructurales: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
Comportamiento: Los patrones de comportamiento nos ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.
Los patrones GoF se clasifican también en 2 **ámbitos**: Clases y objetos por eso se tienen 6 tipos de patrones:

Creacionales:

Creacional de la Clase**:** Usan la herencia como un mecanismo para lograr la instanciación de la Clase. Por ejemplo el método Factoría Creacional del objeto: Son más escalables y dinámicos comparados de los patrones creacionales de Clases. Por ejemplo la Factoría abstracta y el patrón Singleton.

Estructurales:

Estructural de la Clase: Usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples Clases. Por ejemplo el patrón Adaptador (Clase).

Estructural de Objetos: Crean objetos complejos agregando objetos individuales para construir grandes estructuras. La composición del patrón estructural del objeto puede ser cambiado en tiempo de ejecución, el cual nos da flexibilidad adicional sobre los patrones estructurales de Clases. Por ejemplo el Adaptador (Objeto), Facade, Bridge, Composite.

Comportamiento:

Comportamiento de Clase**:** Usan la herencia para distribuir el comportamiento entre Clases. Por ejemplo Interpreter.

Comportamiento de Objeto: Permite analizar los patrones de comunicación entre objetos interconectados, como objetos incluidos en un objeto complejo. Ejemplo Iterator, Observer, Visitor. (6)

1.5 Tecnologías y Herramientas de Desarrollo

Las herramientas de desarrollo son aquellos programas o aplicaciones que tengan cierta importancia en el desarrollo de un programa (programación). Pueden ser de importancia vital (como un ensamblador, un compilador o un editor) o de importancia secundaria, como una Interfaz de Desarrollo Estructurada (IDE). A continuación una breve descripción de las herramientas utilizadas.

1.5.1 Herramienta CASE

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Están destinadas a:

- ✓ Mejorar la productividad en el mantenimiento y desarrollo del software.
- ✓ Automatizar, desarrollo del software, generación de código, pruebas de errores y gestión del proyecto.
- ✓ Mejorar el tiempo y costo de desarrollo y mantenimiento de los sistemas informáticos.

- ✓ Aumentar la calidad del software.
- ✓ Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.5.1.1 Visual Paradigm

Es una herramienta CASE de modelado que utiliza UML como lenguaje de modelado profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar ingeniería tanto directa como inversa. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o .Pdf, y permite control de versiones. Cabe destacar igualmente su robustez, usabilidad y portabilidad. En definitiva, VISUAL PARADIGM es una herramienta muy a tener en cuenta a la hora de ponerse manos a la obra con un proyecto importante. Para el desarrollo de este trabajo se escogió esta herramienta CASE de modelado pues además de todas estas características tiene un coste favorable y realiza de una forma íntegra los planes de construcción del software.

1.5.2 Eclipse

Eclipse IDE comenzó como un proyecto de International Business Machines (IBM) Canadá. Fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge.

Como característica principal tiene el empleo de módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente, al permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++.La arquitectura *plug-in* permite escribir cualquier extensión deseada en el ambiente.

Se utilizó la última versión estable que es la 3.3 la cual fue liberada el 25 de junio de 2007. Dentro de la rama 3.3, su versión actual más avanzada es la 3.3.1.1, liberada el 23 de octubre de 2007.

Características:

- Editor de Texto.
- Resaltado de Sintaxis.

- Compilación en tiempo real.
- Pruebas unitarias con Junit.
- Control de Versiones.
- Asistentes (Wizards), para la creación de proyectos, clases, tests, etc.
- Refactorización.

1.5.3 Control de versiones

Una versión, revisión o edición de un producto, es el estado en el que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Un sistema de control de versiones debe proporcionar un mecanismo de almacenaje de los elementos que deba gestionar y un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto) entre otros aspectos.

1.5.3.1 Subversión 1.6.2.1

Subversión es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS el cual posee varias deficiencias. Este sistema proporciona un mecanismo de almacenaje de los elementos que se deben gestionar y un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto) entre otros aspectos.

Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversión es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial.

1.5.4 Tomcat 6.0

Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JSP de Sun Microsystems. Dado que fue escrito en Java, funciona en cualquier sistema operativo que disponga de una máquina virtual Java. Es cada vez más utilizado por las empresas en los entornos de producción debido a su contrastada estabilidad. Tomcat 6.0 gestiona todos los host virtuales que se ejecutan en el mismo proceso o procesos diferentes, eso significa poder tener problemas de recursos y sobre todo de memoria. Tomcat 6.0 tiene la posibilidad de configurar mediante parámetros la configuración de la pila dinámica de memoria.

Las opciones que puede establecer son:

- Tamaño inicial de la pila de Java :Parámetros -Xms
- Tamaño máximo de la pila de Java :Parámetro -Xmx
- Tamaño de la pila de proceso de Java :Parámetro -Xss
- Tamaño Máximo Memoria Permanente :Parámetro : -XX:MaxPermSize(6)

1.5.5 Máquina Virtual JDK 6.0

La Máquina Virtual Java (MVJ) es el núcleo del lenguaje de programación Java. De hecho, es imposible ejecutar un programa Java sin ejecutar alguna implantación de la MVJ. En la MVJ se encuentra el motor que en realidad ejecuta el programa Java y es la clave de muchas de las características principales de Java, como la portabilidad, la eficiencia y la seguridad.

Siempre que se corre un programa Java, las instrucciones que lo componen no son ejecutadas directamente por el hardware sobre el que subyace, sino que son pasadas a un elemento de software intermedio, que es el encargado de que las instrucciones sean ejecutadas por el hardware. Es decir, el código Java no se ejecuta directamente sobre un procesador físico, sino sobre un procesador virtual Java. También se usa esta idea de máquina virtual en lenguajes de programación compilados. En estos casos lo que se persigue es compilar los fuentes del programa para una máquina determinada. Esta máquina ni siquiera tiene por que existir físicamente. Posteriormente, la máquina virtual adecuada podría ejecutar estos programas independientemente del sistema operativo que esté ejecutándose por debajo.

1.5.6 Gestor de base de datos

Un sistema gestor de base datos es el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a llevar a cabo la definición de los datos, así como el mantenimiento de su integridad, el control de su seguridad, su privacidad y su manipulación.Un sistema gestor de base de datos está compuesto del gestor de la base de datos, que se trata de un conjunto de programas no visibles al usuario final que se encarga de la privacidad, integridad, la seguridad de los datos y la interacción con el sistema operativo. Proporcionando una interfaz entre los datos, los programas que los manejan y los usuarios finales. Entre los Sistemas Gestores de Base de Datos (SGBD) más utilizados en el mundo se encuentran Oracle, MySQL, Microsoft SQL Server, PostgreSQL, InterBase, entre otros. Todos estos presentan un enfoque relacional con un buen basamento matemático centrado en el Álgebra Relacional. Estos sistemas presentan disímiles ventajas, entre las que se encuentran:

- ✓ Facilidad de manejo de grandes volumen de información.
- ✓ Gran velocidad.
- ✓ Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.

1.5.6.1 SQL Server

Uno de los principales gestores de base de datos es el SQL Server, se ha convertido en una de las más utilizadas a nivel mundial. Una de las principales ventajas que presenta es la flexibilidad de su código. Tiene incorporado operaciones avanzadas como los procedimientos almacenados, vistas y disparadores (triggers). Trabaja con múltiples plataformas GNU/Linux, Mac OS X, Windows XP, Solaris, entre otros. SQL Server es conocida como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones Web dirigidas a bases de datos. SQL Server es una base de datos rápida y fiable que se integra a la perfección con PHP y que resulta muy adecuada para aplicaciones dinámicas basadas en Internet. SQL Server incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, administrar el sistema, proteger y hacer volcados de datos. Puede desarrollar sus propias aplicaciones de base de datos y ejecutarlos en casi todos los sistemas operativos, incluyendo algunos de los que probablemente no has oído nunca hablar. SQL Server utiliza el lenguaje de consulta estructurado (SQL).

1.5.7 Navegador

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden. Permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web.

1.5.7.1 Mozilla Firefox

Mozilla Firefox es el nuevo e innovador navegador open source. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. Es Software libre. Se trata de un práctico y ágil navegador, que está en renovación constante. Tiene la capacidad de modificarlo totalmente a gusto del usuario y según las necesidades del mismo. Esto se consigue gracias a la multitud de "extensiones" que existen, y que cada día aparecen más, que permiten añadirle nuevas funciones de todo tipo. Se utilizó el Mozilla Firefox 3.0 ó superior.

Algunas características de Mozilla Firefox son:

- ✓ Navegación por pestañas, esta es una de las principales características que tiene Firefox.
- ✓ Es Software libre.
- ✓ Trabaja de forma excelente en computadoras sin hardware muy potente, el programa está diseñado para realizar un bajo consumo de recursos.
- ✓ Soporta Windows (en cualquiera de sus versiones), Linux (Desde la versión 2.2 del Kernel) y Mac (Desde Mac OS X 10.1.x a la 10.2.x y nuevas versiones).

1.6 Frameworks utilizados en la solución

Un framework, constituye una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Un framework (la traducción aproximada es marco de trabajo) es una estructura de soporte compuesta por componentes genéricos, personalizables, intercambiables y configurables para la organización de un proyecto de software. Son muy utilizados y aceptados, producto de su capacidad de promover la disminución del tiempo de desarrollo del software, la

reutilización del código del diseño, el código fuente y el uso de patrones como buenas prácticas de programación.

1.6.1 Capa de presentación

Esta capa estará dividida en dos partes. Una subcapa del lado del servidor, encargada de recibir todos los pedidos de la interfaz de usuario, controlar el flujo de presentación del sistema y enviar las respuestas correspondientes a la interfaz de usuario. La otra subcapa estará en el cliente, utilizándose los componentes visuales de Java Script para manejar los eventos y validaciones del lado de cliente. La subcapa colocada en el lado del servidor estará relacionada con la capa de Negocios.

DojoToolkit:

Dojo es un Framework que contiene APIs y widgets (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX. Dojo resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URLs para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Los complementos de Dojo son componentes pre empaquetados de código JavaScript, HTML y CSS que pueden ser usados para enriquecer aplicaciones web.

- Menús, pestañas y Tortis.
- Tablas ordenables, gráficos dinámicos y dibujado de vectores 2D.
- Efectos de animación y la posibilidad de crear animaciones personalizables.
- Calendario, selector de tiempo y reloj.

Una característica importante de las aplicaciones AJAX es la comunicación asíncrona entre el navegador y el servidor. Tradicionalmente, se realizaba con el comando JavaScript XMLHttpRequest. *Dojo* suministra una capa de abstracción (*dojo.io.bind*) para varios navegadores web con la que se pueden usar otros transportes (como *IFrames* ocultos) y diferentes formatos de datos. De esta forma se podrá obtener los campos que se van a enviar como parámetros del formulario de una manera sencilla. *Dojo* provee de un sistema de paquetes para facilitar el desarrollo modular. El script de inicio inicializa una serie de jerarquías de paquetes de espacios de nombre (*io*, *event*, etc.) bajo el paquete raíz *dojo*.

Los paquetes de Dojo pueden contener múltiples archivos. Adicionalmente, ofrece funciones para leer y escribir cookies, proporcionando en el lado cliente una abstracción llamada Dojo Storage. Dojo Storage permite a la aplicación web almacenar datos en el lado cliente, persistencia y seguridad.

1.6.2 Capa de lógica de negocio

Esta capa está divida en dos subcapas principales sin dejar de incluir otras que se necesiten y que estén relacionadas con el negocio. En la Fachada se expondrán todas las funcionalidades que la capa de presentación necesitará. Esta capa invocará métodos de la subcapa de Desarrollo del negocio. En la capa de desarrollo del negocio se implementará el negocio de los módulos en cuestión, y de aquí se accederá de ser necesario a la Capa de Acceso a Datos y/o a otras Capas de Negocios.

Spring MVC: Spring Framework (también conocido simplemente como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores. Por su diseño ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar. Está diseñado como una serie de módulos que pueden trabajar independientemente uno de otro. Además intenta mantener un mínimo acoplamiento entre la aplicación y el propio framework de forma que podría ser desvinculada de él sin demasiada dificultad.

Entre las principales características de este framework se encuentran:"Se centra en la capa intermedia y proporciona enganches para manejar la solución elegida para la capa de presentación y de integración. Permite Internacionalización y localización. Desde el punto de vista de la programación orientada a aspectos Spring permite el uso de aspectos en tiempo de ejecución mediante proxys dinámicos y permite la integración de AspectJ que ofrece funcionalidad completa. La Inyección de dependencias (DI) es una de las bases de Spring sobre la que se cimienta el resto de la arquitectura. La DI se encuentra en el corazón de Spring."

La configuración de Spring está basada en XML. Al no tener anotaciones no le hace dependiente de Java EE 5. Aunque en la versión 2.5 se introduce el uso de las anotaciones. El diseño de Spring está pensado para ofrecer un modelo de cómo debe trabajar la aplicación y cómo se comunican sus partes. Está expresamente concebido para que deba ser extensible y acoplable con otros frameworks, ya que no ofrece una solución completa que abarque desde la presentación al modelo.

1.6.3 Capa de Acceso a Datos

En esta capa se implementarán los métodos encargados en interactuar con el gestor de Base de Datos.

Hibérnate: solución ORM para Java, constituye el framework utilizado en el desarrollo de la capa de acceso a datos se selecciona a partir de sus características y ventajas. Hibérnate es una capa de persistencia objeto/relacional y un generador de sentencias sql. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada puede generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySql, etc.

Hibérnate tiene como objetivo fundamental solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el orientado a objetos y el relacional. Para lograr esto le permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibérnate hace posible a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos.

Hibérnate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente.

Conclusiones

Después de un profundo estudio investigativo realizado, se han expuesto las condiciones y problemas que rodean el objeto de estudio a través de una reseña del estado del arte del tema a tratar, lo cual ha ayudado a la toma de decisiones en cuanto a la estrategia y tecnologías a utilizar, y ha enriquecido la investigación hecha, evidenciándose la necesidad de desarrollar un buscador genérico que le permita a los usuarios buscar cualquier tipo de información.

En el desarrollo de este capítulo se realiza un estudio detallado de los buscadores genéricos existentes en Cuba y en el mundo, el sistema gestor de base de datos a utilizar, las metodologías de desarrollo de software, lenguaje de programación seleccionado, entre otras tecnologías. Este análisis se toma como punto de partida en la elección de las herramientas a utilizar en el desarrollo del módulo que dará solución a la problemática planteada.

Capítulo 2: Diseño e Implementación del Sistema

Introducción

En el presente capítulo se exponen elementos importantes para una solución satisfactoria tales como los requisitos funcionales así como una descripción resumida del caso de uso del sistema. Además se realiza el diseño del sistema obteniéndose los diagramas de clases del diseño web, interacción (secuencia), despliegue, clases persistentes y el modelo de datos. También se presentan los diagramas de componentes elaborados a partir de los diagramas de clases de diseño. Se muestra el código fuente de uno de los principales métodos con la descripción del mismo.

2.1 Características del sistema.

2.1.1 Los requisitos funcionales:

Los requisitos funcionales especifican el comportamiento previsto del sistema. Este comportamiento puede ser expresado como los servicios, tareas o funciones del sistema que se requiere para llevar a cabo. Además de las siguientes características:

- ✓ Son declaraciones de los servicios que debe proporcionar el sistema.
- ✓ Especifica la manera en que éste debe reaccionar a determinadas entradas.
- ✓ Especifica cómo debe comportarse el sistema en situaciones particulares.
- ✓ Pueden declarar explícitamente lo que el sistema no debe hacer. (6)

Para dar cumplimiento a las funcionalidades del sistema se detectaron **29** requerimientos funcionales, a continuación se listan los mismos:

RF1 Buscar Acuerdo.

RF2 Buscar Capacidad Financiera.

RF3 Buscar Negociación de CC.

RF4 Buscar Negociación de CC.

RF5 Buscar Documento de Embarque.

RF6 Buscar Discrepancia de DE.

RF7 Buscar Cuenta.

RF8 Buscar Banco.
RF9 Buscar Comisión.
RF10 Buscar Transacción.
RF11 Buscar Cuenta Banco.
RF12 Buscar Cuenta de Concepto.
RF13 Buscar Tasa de Cambio.
RF14 Buscar Depósito a Plazo.
RF15 Buscar Cliente Jurídico.
RF16 Buscar Persona Natural.
RF17 Buscar Cuenta de Cliente.
RF18 Buscar Disponibilidad de Fondos.
RF19 Buscar Reservación de Fondos.
RF20 Buscar Sobregiro Autorizado.
RF21 Buscar Préstamos.
RF22 Buscar mensaje Conformado SLBTR.
RF23 Buscar mensaje Conformado SWIFT.
RF24 Buscar Mensajes Recibidos (Buzón de Mensajes).
RF25 Buscar mensajes.
RF26 Buscar Cheque.
RF27 Buscar Carta de Remesa.
RF28 Buscar Chequera.
RF29 Buscar Carta de Crédito.

2.1.2 Diagrama de Casos de Uso del sistema.

El Diagrama de Casos de Uso es un diagrama diseñado para el desarrollo de software dentro del contexto de la Metodología Iterativa Incremental. Brinda valiosa información para el desarrollo de un proyecto, así mismo es un puente de comunicación entre el cliente, el analista y el programador. (7)Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto. (8)

A partir de los requisitos funcionales identificados se realizó el diagrama de caso de usos del sistema.

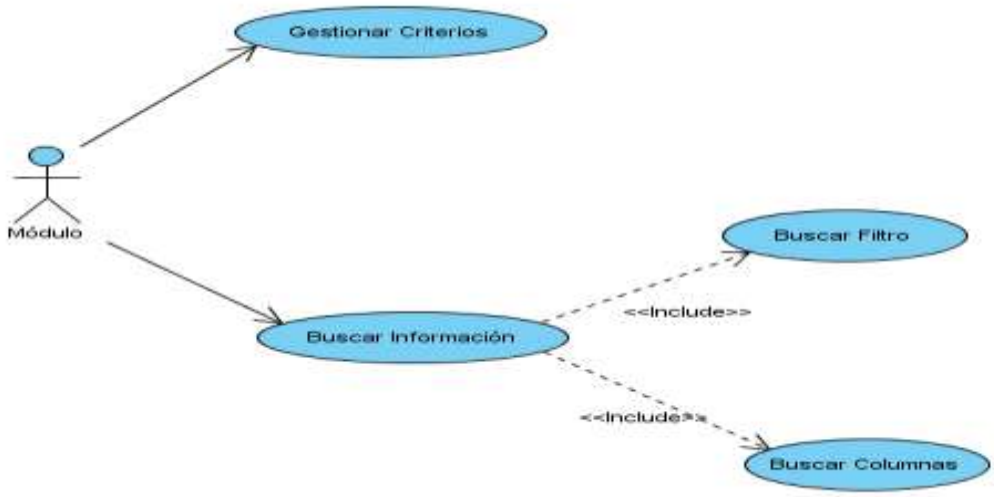

Fig. 2: Diagrama de Casos de Uso del sistema.

2.1.3 Descripción del Caso de Uso del sistema.

A continuación se muestra la descripción del caso de uso determinado para satisfacer los requerimientos funcionales de sistema. Para ver la descripción de los casos de uso restantes remitirse a los **anexos 1**"Descripción de los casos de usos".

Nombre del Caso de Uso	Buscar información
Actor	Módulo.
Propósito	Buscar cualquier tipo de información en un momento determinado.
Resumen	El caso de uso se inicia cuando el Módulo desea buscar una determinada información, el sistema responde esta petición mostrando los resultados en una tabla resultante.
Referencias	Todos los requisitos funcionales.

Tabla #1: Descripción resumida del Caso de Uso: Buscar información.

2.2 Arquitectura y diseño.

La arquitectura propuesta está compuesta por una arquitectura de tres capas y una capa transversal a las otras capas con los objetos del dominio.A continuación se explicará cada una de las capas lógicas del sistema:

Capa de presentación

En esta capa se desarrollará la lógica de presentación. En el servidor se utilizará Spring MVC para recibir, controlar y enviar una respuesta a las peticiones realizadas desde el cliente. Se utilizará también Spring WebFlow para representar y controlar los flujos de presentación que sean complejos y reutilizables por varios módulos de la aplicación. En el cliente se utilizará la librería Dojo para generar las interfaces que interactuarán con el usuario. La capa de presentación estará relacionada con la Capa de Negocios y Capa de Dominio.

Capa de negocio

Esta capa estará divida en dos subcapas. Estas subcapas son Fachada y Manager. La Fachada será el punto de intercambio entre la capa de presentación y la capa de negocio. Esta capa no tendrá lógica de negocio sino que agrupará funcionalidades según su naturaleza para que pueda ser invocada desde la capa de presentación. La subcapa Fachada capa delegará a la subcapa Manager la realización de la lógica del negocio. Por otro lado la subcapa Manager tendrá la jerarquía de clases suficiente para implementar el negocio de la aplicación. Esta subcapa utilizará la capa de acceso a datos para obtener datos que están persistidos y la capa de dominio para generar los objetos de dominios.

Desde la capa de negocio se envolverá todas las funcionalidades de la aplicación en diferentes niveles de transacción para evitar inconsistencia en los datos persistentes.

Se realizará la auditoría de la aplicación desde los objetos del negocio que se utilizarán en esta capa para monitorear cada funcionalidad que se ejecute en el sistema.

Se utilizará el contenedor de Spring Framework para declarar y representar las relaciones de dependencia de cada una de las clases que existan. Spring Security para aplicar seguridad a nivel de métodos. Se utilizará las políticas de transacciones que propone Spring framework. Spring AOP para ejecutar dichas transacciones y la auditoría de cada uno de los métodos del negocio.

Capa de acceso a datos

En esta capa se realizarán todas las operaciones relacionadas con el gestor de base de datos y cualquier recurso que contenga información persistida. La capa de negocio interactuará con esta capa a través de interfaces. Para desarrollar esta capa se utilizará el patrón DAO (Data Acces Object en inglés), en español objetos de acceso a datos. Se utilizará la filosofía de un objeto DAO genérico utilizado. La implementación utilizada en el sistema SIGEP para esto será reutilizada.

El diseño general de esta capa será una interface que responderá a un grupo de operaciones de una entidad de dominio persistente y la correspondiente implementación de esa interface.Se utilizará el framework de persistencia Hibérnate y los módulos Spring ORM y Spring DAO. Hibérnate como tecnología ORM para el manejo objetual de acciones de persistencia con la base de datos. Spring ORM y Spring DAO para la soportar la integración con Hibérnate y la utilización del patrón DAO.

2.2.1 Patrón arquitectónico MVC.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes. Para el desarrollo del sistema se escogió el framework Spring MVC, framework que está basado en un patrón clásico del diseño como arquitectura MVC, formado por 3 niveles:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. termina sus operaciones devuelve el flujo al controlador y este envía el resultado a la capa de presentación.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (2)

2.2.2 Patrones Estructurales

Facade (Fachada): El objetivo de la utilización de este patrón es proveer un intermediario reduciendo el número de objetos con los que interactúa un cliente y una interfaz, proporcionando un bajo acoplamiento,

además de emitir cambios en la lógica implementada sin afectar al cliente que la utiliza ya que la misma está oculta tras la Fachada.

2.2.3 Patrones de Comportamiento

Patrón de acceso a datos

Data Access Object (DAO): Centraliza todo el acceso a datos en una capa independiente, aislándolo del resto de la aplicación. Desacoplando la lógica de negocio de la lógica de acceso a datos, de manera que se pueda cambiar la fuente de datos fácilmente. Su principal beneficio es que reduce la complejidad de los objetos del negocio al abstraerlos de la implementación real de la comunicación con la fuente de datos. Cada DAO tendrá que implementar los métodos declarados en la interfaz DAO correspondiente.

2.2.4 Tipos de patrones GoF que implementa Spring

En la categoría de **creacionales** Spring utiliza el patrón:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Command: Encapsula peticiones en forma de objetos permitiendo así parametrizar los clientes utilizando distintas peticiones, encolar las peticiones y ofrecer la posibilidad de deshacer las operaciones. Permite solicitar operaciones sin tener que saber cómo o quien lleva a cabo esas operaciones.

En la categoría de **comportamiento**:

Observar: Para no recurrir a soluciones fuertemente acopladas (que reducen la posibilidad de reutilización), este patrón define una dependencia uno-a-muchos entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente.

Controlar (Controlador Frontal): Permite que sólo exista un único punto de entrada en la aplicación, esto ayuda a centralizar las restricciones de seguridad y a delegar las peticiones en otros componentes.

2.3 Pautas del diseño.

Para lograr una agradable apariencia y facilitar el uso de la aplicación se definieron algunas pautas de diseño como son:

- Las opciones para mostrar las columnas del grid siempre van a ser checkbox.

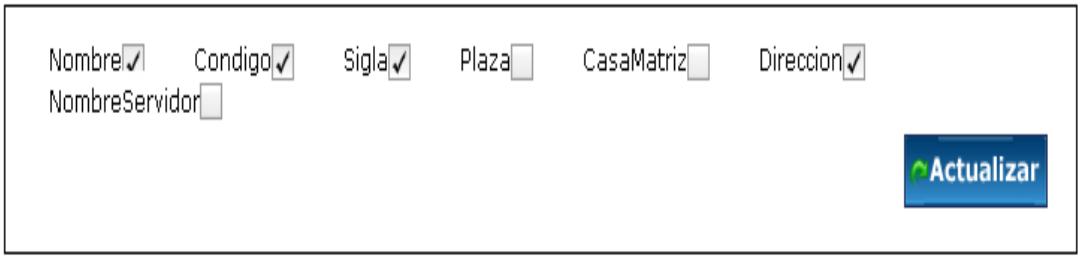

Fig. 3: Muestra las columnas del grid.

- Los botones para efectuar cualquier tipo de operación serán ubicados en el extremo inferior derecho de su contenedor.

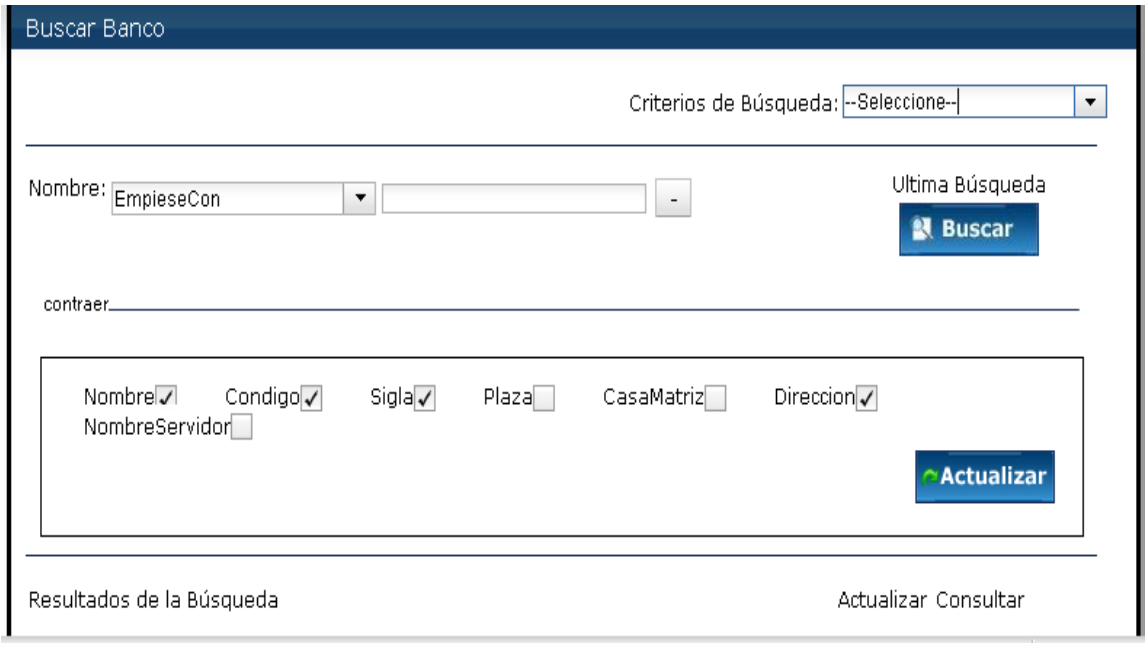

Fig. 4: Muestra los botones de la aplicación.

- Los resultados obtenidos serán mostrados en un grid.

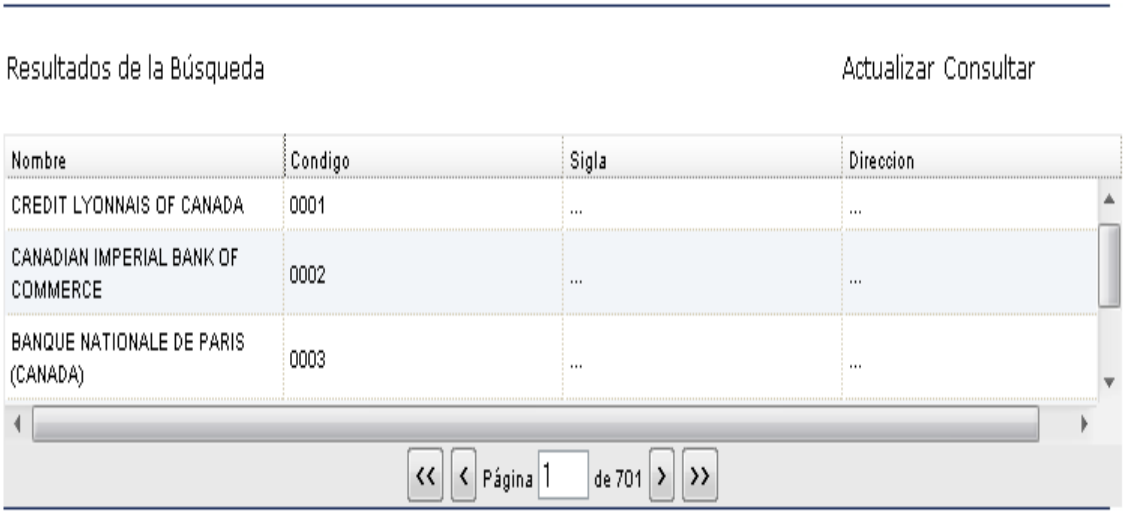

Fig. 5: Se muestra la tabla de resultados.

- Los criterios de búsqueda a seleccionar se mostraran en un combobox.

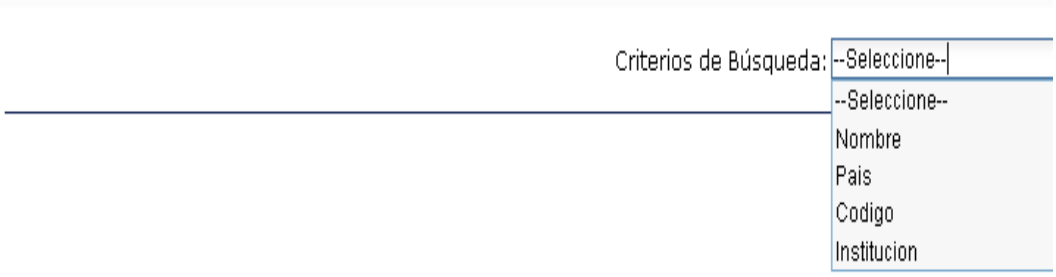

Fig. 6: Se muestra los criterios de búsqueda.

- La información a buscar será recogida en un textbox.

Fig. 7: Se muestran los textbox donde se recogerá la información.

- La información para machear la información recogida será mostrada en un combobox.

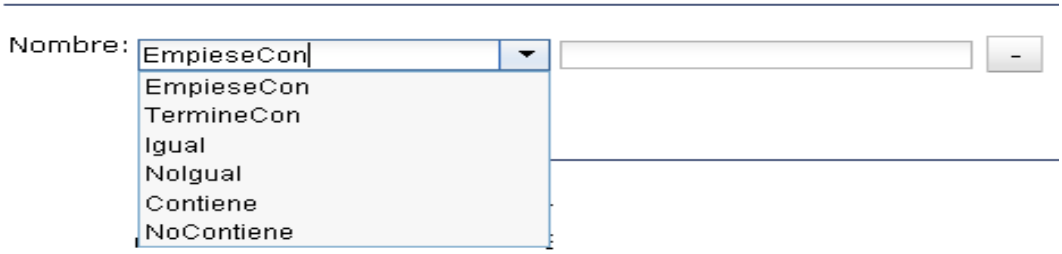

Fig. 8: Se muestran los combobox donde se machea la información.

- La estructura para recoger la información de los criterios deben de ir en una misma línea.

Fig.9: Muestra la alineación de los combobox.

- La opción Buscar en la interfaz estará ubicada en el extremo superior derecho.

Fig. 10: Muestra la ubicación del botón buscar.

- El paginado del grid siempre será de 5x5.
- Se dejaran opciones en blanco para que las otras páginas que incluyan al buscador la puedan configurar conforme a sus necesidades.
- Los checkbox serán mostrados en línea.

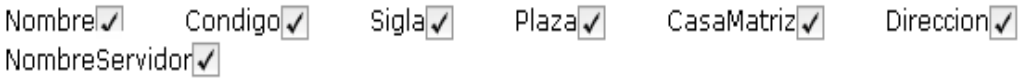

Fig. 11: Muestra la alineación de los checkbox.

2.4 Vistas arquitectónicas.

En 1995 Philippe Kruchten propuso su célebre modelo de 4+1 vistas arquitectónicas las cuales fueron vinculadas a (RUP). Las vistas muestran los diferentes aspectos del sistema que son modelados. Una vista no es un gráfico, pero es una abstracción consistente de un número de diagramas.

Vistas arquitectónicas:

- ✓ **Vista de Casos de Uso**: Muestra la funcionalidad del sistema percibido por actores externos.
- ✓ **Vista Lógica:** Muestra como la funcionalidad es diseñada dentro del sistema, define la estructura y el comportamiento del sistema.
- ✓ **Vista Concurrente o de Procesos**: Muestra la concurrencia en el sistema dividido en procesos y procesadores. Da cuenta de los aspectos de comunicación e integración.
- ✓ **Vista de Despliegue**: Muestra la arquitectura física del sistema.
- ✓ **Vista de Componentes o Implementación**: Muestra la organización de componentes del código y su implementación.

No todas las arquitecturas de software requieren todas las vistas del modelo 4+1, algunas vistas pueden ser omitidas, pero todos los escenarios se pueden encontrar en ellas. La finalidad de estas vistas es, unificar los criterios de modelado, la necesidad de facilitar la comunicación entre los diseñadores y establecer estándares variados que permitan una mayor comprensión del sistema que se esté modelando.

2.5 Modelo de Diseño

El modelo de diseño, es un modelo de objetos que describe la realización física de los casos de uso, centrándose en como los requisitos funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además el modelo de diseño sirve de abstracción de la implementación del sistema y es de ese modo utilizado como una entrada fundamental de las actividades de implementación. Los casos de uso son realizados por las clases de diseño y sus objetos. Esto se representa por colaboraciones en el modelo de diseño y denota realización de caso de uso-diseño. (12)

El modelo de diseño está compuesto por los artefactos clases del diseño, realización caso de uso-diseño, diagramas de interacción, diagramas de clases, subsistemas del diseño, interfaz, descripción de la

arquitectura (vista del modelo de diseño), modelo de despliegue, descripción de la arquitectura (vista del modelo de despliegue).

2.5.1 Diagrama de clases del diseño.

Un diagrama de clases del diseño representa las clases del diseño y sus objetos, así como los subsistemas del diseño. Para ver los diagramas de clases por capas remitirse a los **anexos 2** “Diagramas de clase por capas”.

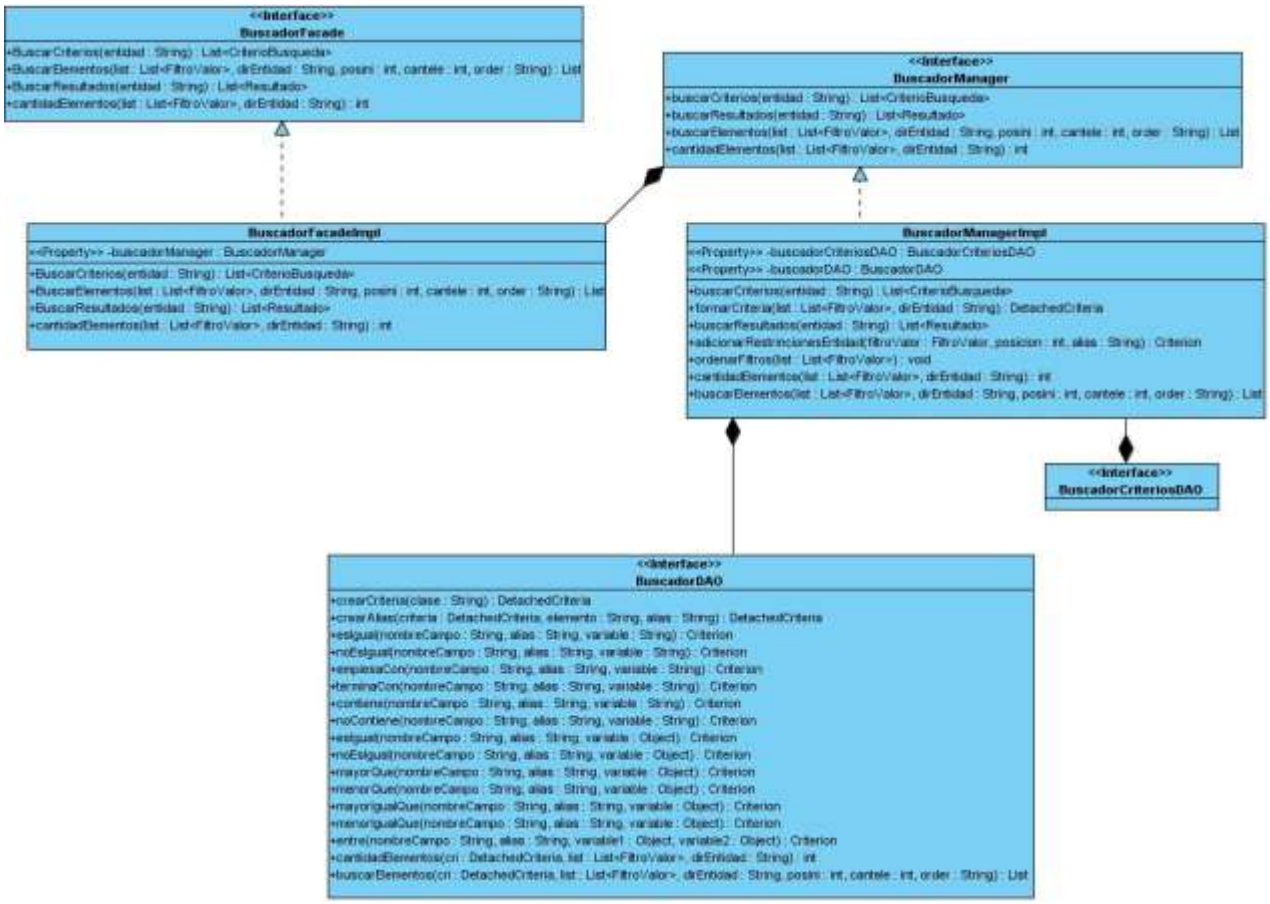

Fig. 12: Diagrama de clase de diseño General.

2.5.2 Descripción de las clases de diseño.

A continuación se muestra la descripción de cada una de las funcionalidades de la clase **BuscadorManagerImpl**.

Nombre de la clase: BuscadorManagerImpl
Nombre del método: buscarCriterios ()
Descripción: Éste método dado una clase por parámetros permite buscar todos los criterios relacionados a la clase dada.
Nombre del método: formarCriteria ()
Descripción: Este método se encarga de clasificar los filtros y seguido a esto forma el objeto criteria.
Nombre del método: buscarResultados ()
Descripción: Este método dado una clase va a la base de datos y busca todos los posibles resultados.
Nombre del método: adicionarRestriccionesEntidad ()
Descripción: Este método coge el objeto criteria y se encarga de adicionar las restricciones elegidas por el usuario.
Nombre del método: ordenarFiltros ()
Descripción: Éste método es el encargado de ordenar para darle una prioridad a los filtros escogidos por el usuario.
Nombre del método: cantidadElementos ()
Descripción: Éste le pide a la base de datos la cantidad de elementos que existe con las restricciones escogidas, es un método de mucha importancia para el

paginado.
Nombre del método: buscarElementos ()
Descripción: Este método es el encargado de buscar los elementos con esas restricciones.

Tabla #4: Descripción de la clase “BuscadorManagerImpl”

2.5.3 Diagramas de interacción.

Los diagramas de interacción muestran en detalle a los objetos y mensajes entre objetos de un determinado escenario para un caso de uso con el objetivo de modelar el comportamiento dinámico del sistema y verificar la coherencia del sistema validándolo con el modelo de clases.
Los diagramas de interacción se expresan de dos formas:

Diagrama de colaboración: Diagrama de interacción que destaca la organización estructural delos objetos que envían y reciben mensajes.

Diagrama de secuencia: Diagrama de interacción que destaca el orden temporal de los mensajes.

2.5.3.1 Diagramas de secuencia.

Los diagramas de secuencia se encuentran entre los diagramas más efectivos para modelar la interacción entre objetos en un sistema. Contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar estos escenarios. Muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes, dibujados como flechas desde la línea de vida origen hasta la línea de vida destino.

A continuación se mostrará el diagrama de secuencias correspondiente al caso de uso Buscar. Para ver los demás diagramas de secuencia remitirse a los **anexos 3**” Diagramas de Secuencia”.

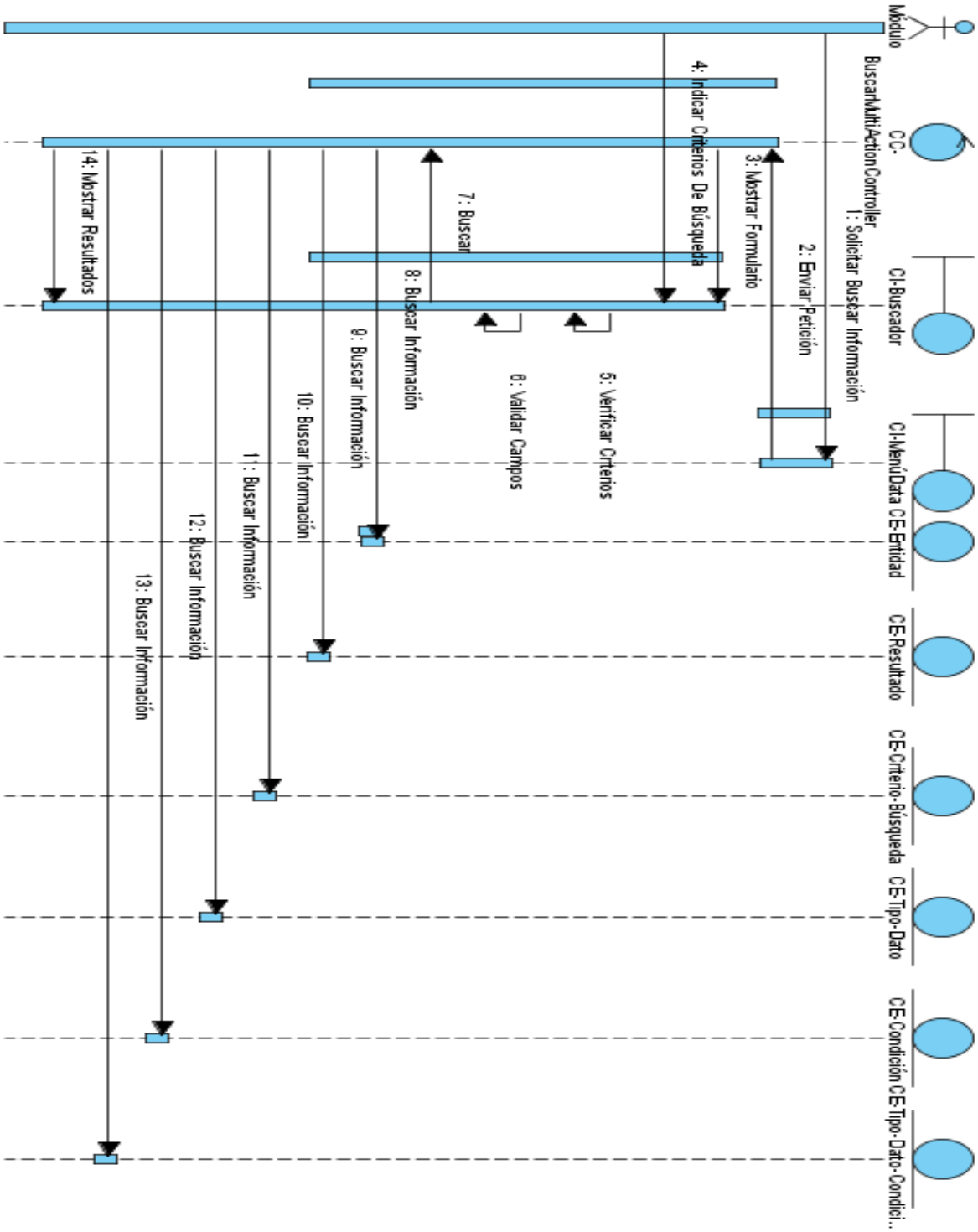
Módulo
BuscarMultiActionController
CC-
CI-Buscador
CI-MenúData
CE-Entidad
CE-Resultado
CE-Criterio-Búsqueda
CE-Tipo-Dato
CE-Condición
CE-Tipo-Dato-Condici...
1: Solicitar Buscar Información
2: Enviar Petición
3: Mostrar Formulario
4: Indicar Criterios De Búsqueda
5: Verificar Criterios
6: Validar Campos
7: Buscar
8: Buscar Información
9: Buscar Información
10: Buscar Información
11: Buscar Información
12: Buscar Información
13: Buscar Información
14: Mostrar Resultados

Fig. 16: Diagrama de Secuencia del Caso de Uso Buscar Información.

2.6 Modelo de datos.

El modelo de datos describe la representación lógica y física de la persistencia de los datos utilizados por la aplicación. En los casos en que la aplicación utilizara un sistema de gestión de base de datos relacional (sus siglas en inglés RDBMS), el modelo de datos también puede incluir procedimientos almacenados, disparadores, restricciones etc. que definen la interacción de los componentes de aplicación.

El modelo de datos estará compuesto por las entidades que pasarán a ser las tablas de la base de datos que será utilizada por el sistema.

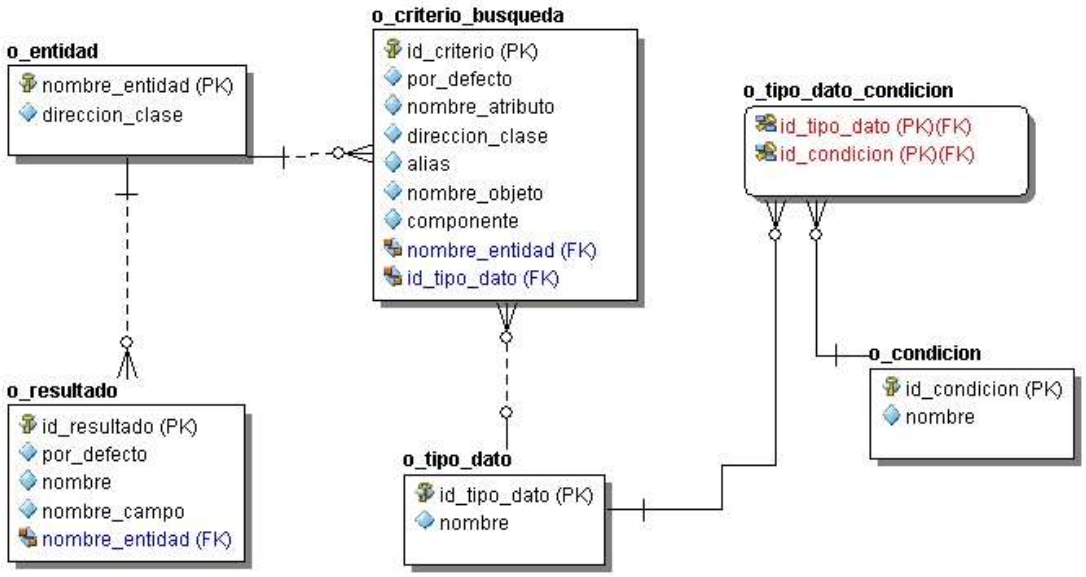

Fig. 19: Modelo de Datos.

2.6.1 Descripción de las tablas.

A continuación se muestra la descripción de las tablas:

Criterio-búsqueda: Tabla que almacena todo lo relacionado con los datos de los criterios de búsqueda.

Atributo	Tipo	Descripción
id_criterio	int	Identificador de la tabla **"Criterio-búsqueda"**.

nombre_atributo	String	Nombre del atributo.
direccion_clase	String	Dirección de la clase.
alias	String	Nombre que voy a mostrar.
nombre_objeto	String	Nombre del objeto.
componente	String	Componente a utilizar.
nombre_entidad	String	Identificador de la tabla "entidad"(llave Foránea).
id_tipo_dato	int	Identificador de la tabla "tipo-dato"(llave Foránea).
por_defecto	bit	Criterios que van por defecto.

Tabla #5: Descripción de la tabla "**Criterio-búsqueda**"

Entidad: Almacena todos los datos relacionados con las entidades que se va a buscar.

Atributo	Tipo	Descripción
nombre_entidad	String	Identificador de la tabla **"entidad"**.
direccion_clase	String	Dirección de la clase.

Tabla #6: Descripción de la tabla "**Entidad**"

Resultado: Almacena todos los datos relacionados con el resultado obtenido.

Atributo	Tipo	Descripción
Id-resultado	int	Identificador de la tabla **"Resultado"**.
por_defecto	bit	Criterios que van por defecto.
nombre	String	Nombre del resultado
Nombre-campo	String	Nombre del campo.
nombre_entidad	String	Identificador de la tabla

		“entidad” (llave Foránea).

Tabla #7: Descripción de la tabla “**Resultado**”

2.7 Implementación

2.7.1 Modelo de implementación.

La Vista de implementación contiene la organización de los módulos en términos de paquetes y capas, pueden incluirse también la trazabilidad de la vista lógica. Esta vista toma en cuenta los requerimientos que facilitan la programación, los niveles de reutilización y las limitaciones impuestas por el entorno de desarrollo. Es representada por un diagrama de componentes o especificaciones de paquetes que son básicamente un subconjunto del modelo de despliegue. Para modelarla se dispone de dos elementos, los paquetes que representan una partición física del sistema y los componentes que representan la organización de los módulos de código fuente. A continuación se muestra la Vista de Implementación del sistema.

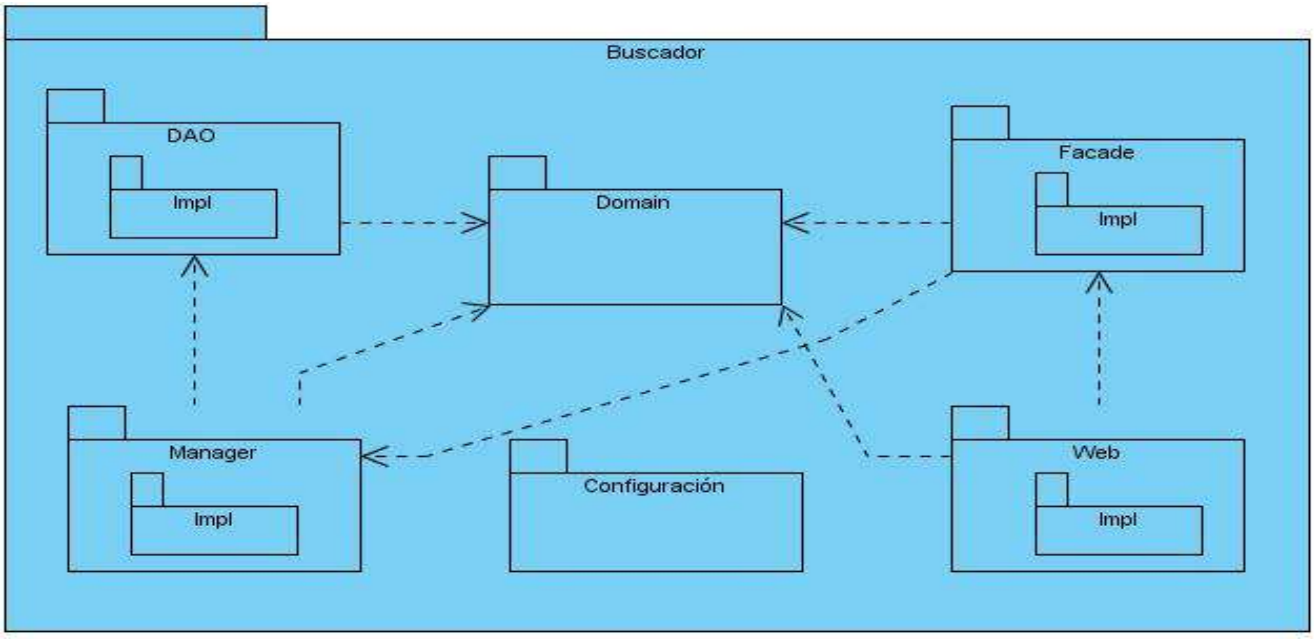

Fig. 20: Modelo de Implementación.

La vista de implementación del sistema queda constituida por un grupo de componentes, y subsistemas de implementación que modelan el empaquetado físico real del sistema.

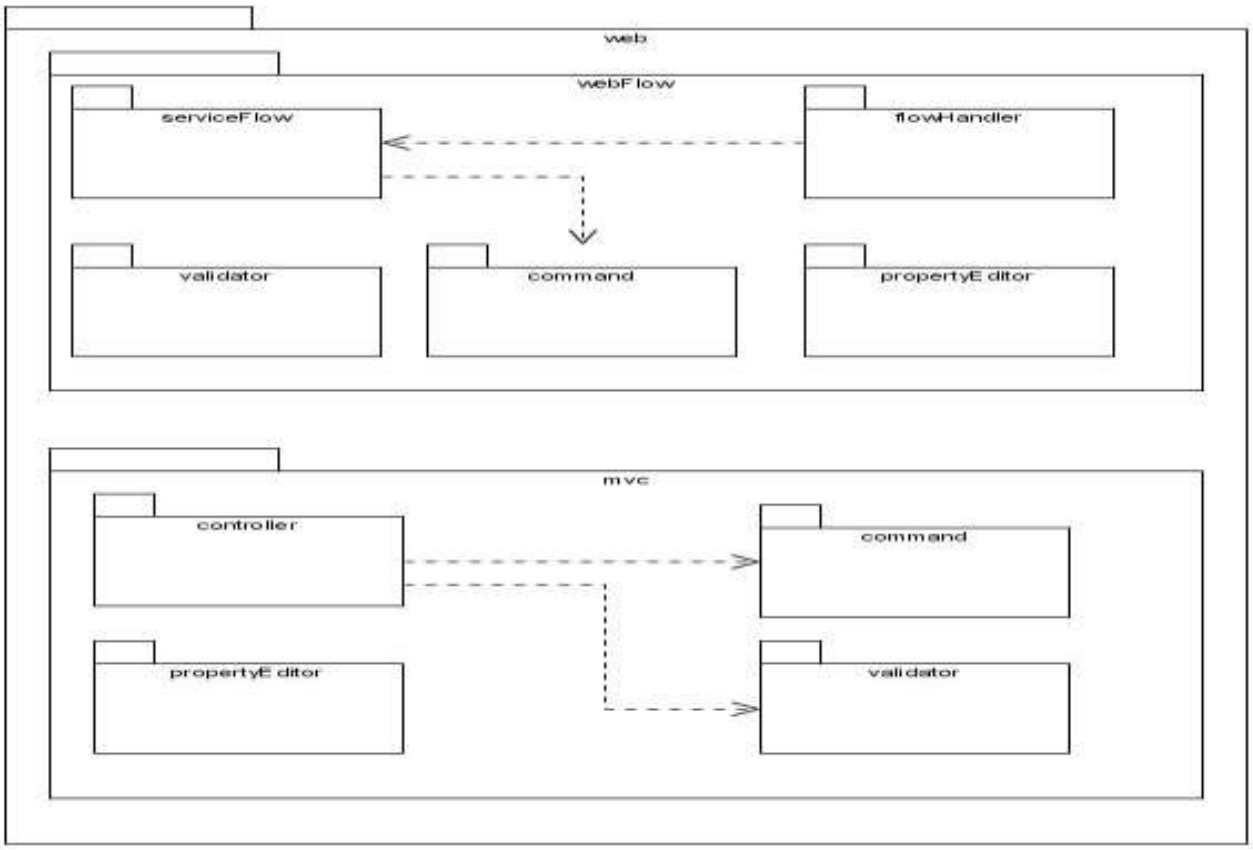

Fig. 21: Paquetes de clases en la capa de Presentación.

Todo el desarrollo que pertenezca a la Capa de Presentación se colocará debajo del paquete web. Este paquete se divide en dos sub-paquetes (webFow) y (mvc).

En el paquete webFlow estarán las clases que se desarrollen sobre Spring WebFlow. Este sub-paquete tendrá en principio los sub-paquete:

- propertyEditor: Clases que convierten ciertos datos en objetos determinados
- validator: Clases para desarrollar las validaciones en el servidor

- command: Clases para mapear datos que se introducen desde el cliente a objetos del negocio.
- serviceFlow: Clases que hereden de la clase Action o MultiAction y/o clases que interactúan con el paquete Facade de la Capa de Negocio
- flowHanlder: Clases FlowHandler para personalizar el trabajo con el WebFlow.

En el paquete mvc estarán las clases que se desarrollen sobre Spring MVC. Este sub-paquete tendrá en principio los sub-paquete:

- propertyEditor: Clases que convierten ciertos datos en objetos determinados
- validator: Clases para desarrollar las validaciones en el servidor
- command: Clases para mapear datos que se introducen desde el cliente a objetos del negocio.
- controller: Clases que heredan de las clases “Controller” que brinda Spring MVC.

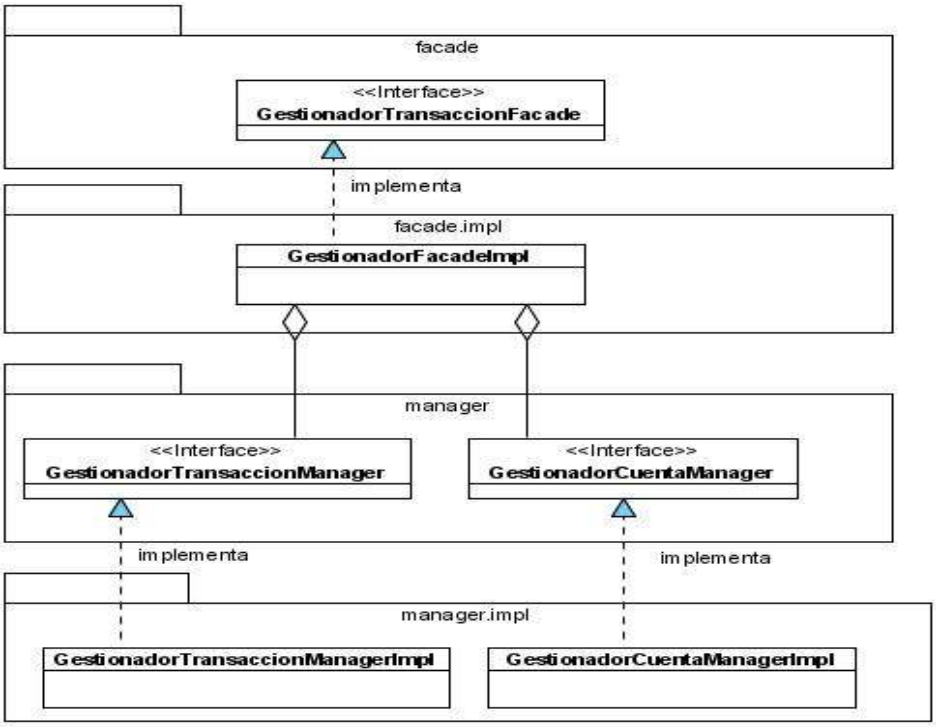

Fig. 22: Paquetes de clases en la capa de Negocio.

- facade*:* Se declaran las interfaces para brindar los servicios que se consumirán por otros módulos o por la capa de presentación.
- facade.impl*:* Se crean las clases que implementan los métodos de las interfaces creadas en el paquete Facade.
- manager: Se declararán las interfaces que brindarán las funcionalidades del negocio.
- manager.impl: Se crean las clases que implementan los métodos de las interfaces creadas en el paquete manager.

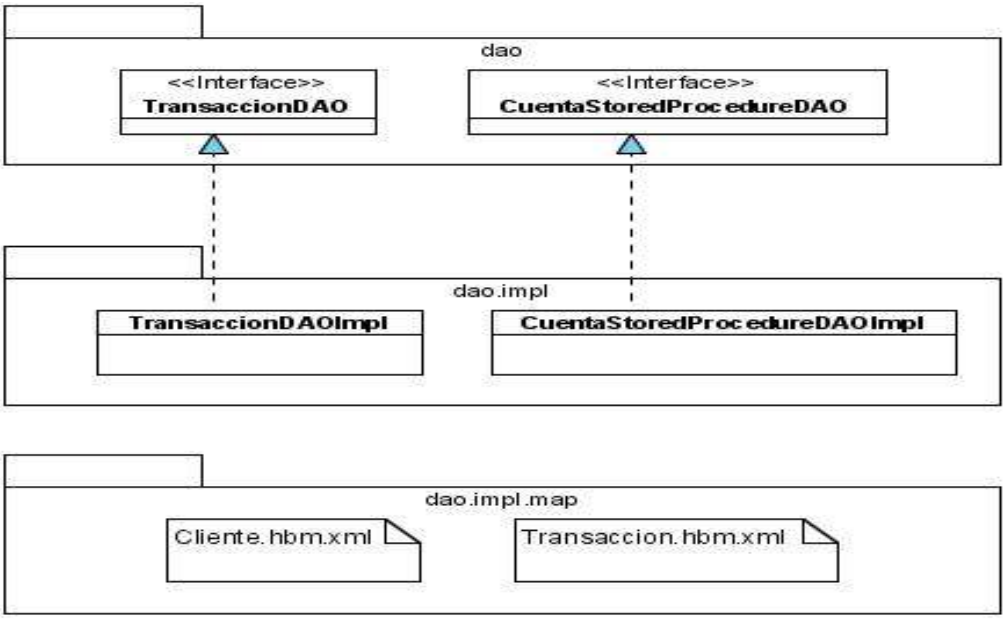

Fig. 23: Paquetes de clases de la Capa de Acceso a Datos.

Dao: Se encuentran las interfaces DAO e interfaces StoredProceduredDAO.

dao.impl: Se encuentran las clases que implementan las interfaces del paquete dao.
dao.impl.map*:* Se encuentran los ficheros mapeos de Hibérnate.

2.8.2 Diagrama de Componentes

El modelo de implementación se estructura a partir de los diagramas de componentes, por tal razón debe entenderse el funcionamiento interno de los módulos para luego asociarlos como un todo dentro de los subsistemas que componen. Al utilizar la metodología RUP debe conocerse que la misma hace énfasis en

el uso de diagramas de componentes para modelar los subsistemas de implementación. “Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Estos diagramas muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, bibliotecas cargadas dinámicamente, entre otros. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.” (15)

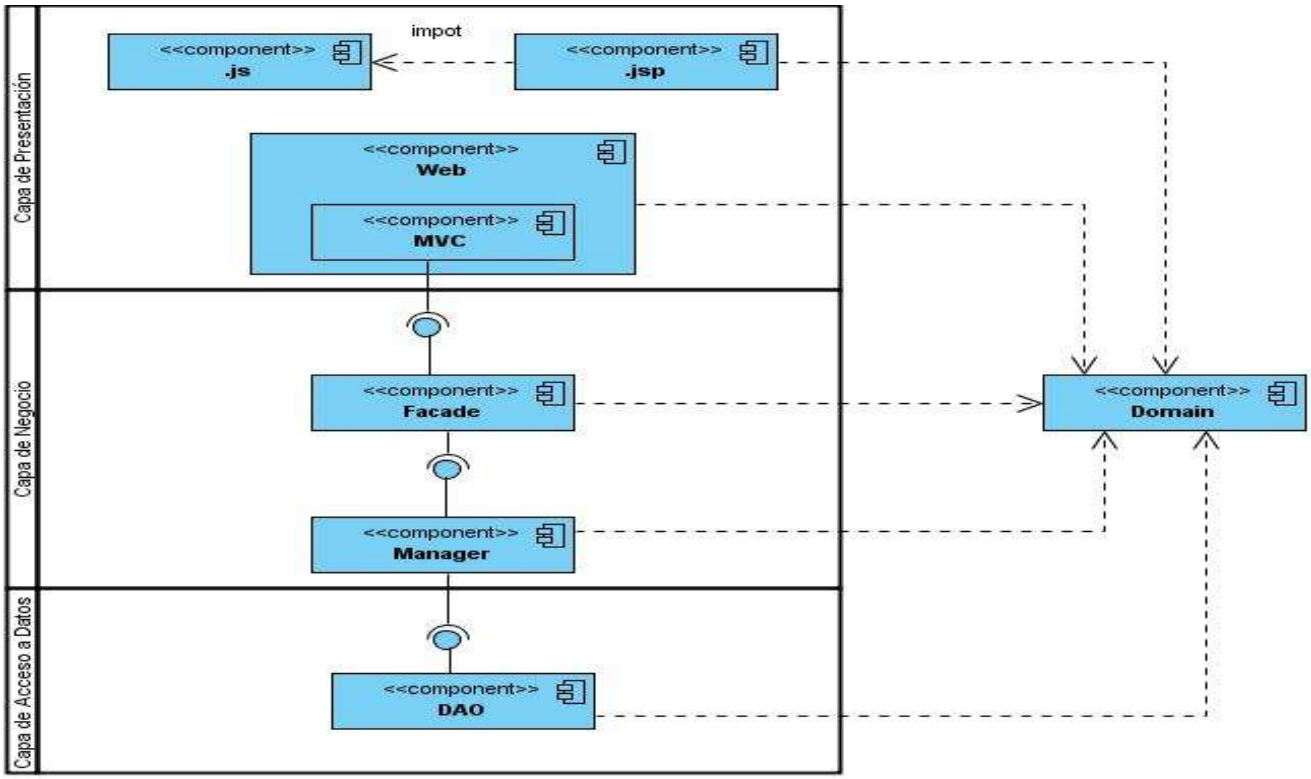

Fig. 24: Diagrama de Componentes.

2.8.3 Estándares de codificación

Cuando se trabaja dentro de un equipo de desarrollo debe generalizarse un estándar de codificación de manera que exista similitud de formato entre el código de los implementadores. Es una realidad que cada programador tiene un estilo personal para escribir su código fuente pero al definirse un estándar para utilizar se debe acoplar dicho estilo personal con el de todo el equipo. Se especifica como estándar de

codificación al estilo de la sintaxis que se usará en la programación, lo anterior puede incluir la forma de estructurar el código fuente, la manera de colocar las llaves, la identación del código, el empleo de los paréntesis, entre otros aspectos.

A continuación se especifican los criterios de estandarización de código utilizados para el desarrollo del Buscador Genérico:

- ✓ Nombres de los archivos

Los nombres de archivos deben ser sustantivos y comenzar con letra mayúscula, si está compuesto por más de una palabra se forma el nombre con todas las palabras juntas y la primera letra de cada palabra con letra mayúscula. Solamente se permite el uso de palabras completas, no se pueden usar abreviaturas y el nombre debe describir correctamente al archivo.

- ✓ Organización del archivo

Cada archivo contiene varias secciones (funciones) y deben separarse con un espacio en blanco entre ellas.

Los archivos con más de 1000 líneas de códigos deben ser evitados debido a que dificultarían un posterior mantenimiento.

- ✓ Comentarios al inicio

Todos los archivos deben empezar con un comentario donde se especifique el autor del fichero y de manera opcional se puede poner una breve descripción de su propósito. Por ejemplo:

/*

* @Autor: Universidad de las Ciencias Informáticas.

*/

- ✓ Identación

Se usará 1 tabla como unidad de identación.

Longitud de línea

Evitar líneas mayores de 80 caracteres.

- ✓ Estructura de las líneas

Cuando una expresión no quepa en una sola línea, se debe dividir siguiendo los siguientes principios:

Dividir después de una coma.

Dividir después de un operador (lógico o aritmético).

Alinear la nueva línea al principio al mismo nivel que la expresión de la línea anterior.

- ✓ Cantidad de declaraciones por línea

Es recomendado que solamente haya una declaración por línea, ello facilita la posibilidad de añadirle comentarios a cada línea en caso de ser necesario.

- ✓ Lugar de las declaraciones

Poner todas las declaraciones al principio de cada bloque de código (se denota bloque de código al área que está entre dos llaves “{” y “}”).

- ✓ Inicialización de variables

Siempre tratar de inicializar las variables en el mismo lugar donde son declaradas. La única razón para no hacerlo es que su valor inicial dependa de alguna operación computacional previa.

- ✓ Declaración de funciones y de clases

Se deben seguir las siguientes reglas a la hora de declarar clases y funciones en Java script:

No debe haber espacios entre el nombre de las funciones y el paréntesis “(” que da inicio a la lista de parámetros.

La llave de apertura “{” al final de la misma línea de la declaración.

La llave de cierre "}" empieza una línea por ella misma al mismo nivel de identación que la línea donde se abrió, excepto cuando no haya nada entre ambas llaves, estaría la del cierre a continuación de la de apertura.

Las funciones se separan por una línea en blanco.

- ✓ Expresiones simples

Cada línea debe contener como máximo una sola expresión. No se puede hacer lo siguiente:

cantidadUsada++; cantidadDisponible--; //ERROR!!!

- ✓ Líneas en blanco

Las líneas en blanco mejoran la legibilidad del código al separar secciones lógicas dentro del mismo.

Se debe usar una línea en blanco siempre que pase alguna de las siguientes situaciones:

Entre funciones.

Entre secciones lógicas dentro de una misma función para mejorar la legibilidad.

Después de las declaraciones de las variables.

- ✓ Espacios en blanco

Se deben usar los espacios en blanco en las siguientes circunstancias:

Una palabra reservada seguida por un paréntesis debe separarse con un espacio en blanco, asimismo, la llave de apertura "{" siempre debe estar precedida por espacio en blanco. Ejemplo:

```
while (true) {
    ...
}
```

Después de las comas en una lista de argumentos debe usarse un espacio en blanco.

- ✓ Todos los operadores binarios excepto él "**.**" deben separarse de sus operandos con un espacio en blanco. Por el contrario, con los operadores unarios ("++", "--") nunca se debe usar los espacios en blanco para separar. Ejemplos:

```
a += c + d;
a = (a + b) / (c * d);
while (d++ == s++) {
    n++;
}
alert("la longitud es " + longitud + "\n");
```

- ✓ Las expresiones de la instrucción *for* se deben separar con un espacio en blanco. Ejemplo:

```
for (expresion1; expresion2; expresion3)
```

- ✓ Clases

Las clases deben tener el mismo nombre del archivo.

- ✓ Funciones

Las funciones deben ser verbos, comenzar siempre con letra minúscula y en caso de ser varias palabras, se pondrían todas seguidas siendo la primera letra de la primera palabra minúscula y la primera letra de cada una de las restantes palabras sería mayúscula. No se puede usar abreviaturas y el nombre debe describir completamente la acción principal de la función.

- ✓ Variables

Se cumple lo mismo que en las funciones, aunque para el caso que se usen variables temporales se pueden usar abreviaturas (i, j, k, etc.).

- ✓ Constantes

Las constantes se escriben con mayúscula la palabra completa y si son más de una palabra se separan con un guión bajo "_".

Conclusiones:

A lo largo de este capítulo se revelaron los resultados de las disciplinas de Diseño e Implementación en el proceso de desarrollo del Buscador Genérico. Se desarrollaron los Diagramas de Clases y de Secuencia de los Casos de Uso fundamentales en el sistema, así como los tipos de clases que existen en el diseño de la solución y el Modelo de Datos. Se expuso también un grupo de componentes desarrollados para ser utilizados en el sistema, además de una serie de reglas que se asumieron para la codificación del software. Como complemento se brinda también el Diagrama de Componentes.

Capítulo 3: Validación de la Solución Propuesta

Introducción

3.1 Pruebas de Software

La prueba es el proceso de ejecutar un sistema bajo condiciones o requerimientos especificados con la intención de descubrir errores. Se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas (configuración de la prueba), registrándose los resultados obtenidos. Las pruebas mejoran la integridad de un sistema, al detectar las desviaciones del diseño y los errores en el sistema. Tienen como objetivo detectar las áreas propensas a errores. Esto ayuda a la prevención de errores en un sistema e incrementan el valor del producto al adaptarlo a las necesidades del usuario.

3.1.1 Características de una buena prueba

Entre las principales características de una buena prueba se encuentran las siguientes:

- Una buena prueba ha de tener una alta probabilidad de encontrar un fallo. Para alcanzar este objetivo el responsable de la prueba debe entender el software e intentar desarrollar una imagen mental de cómo podría fallar.
- Una buena prueba debe centrarse en dos objetivos:
  - ✓ Probar si el software no hace lo que debe hacer.
  - ✓ Probar si el software hace lo que no debe hacer.
- Una buena prueba no debe ser redundante. El tiempo y los recursos son limitados, así que todas las pruebas deberían tener un propósito diferente.
- Una buena prueba debería ser la “mejor de la cosecha”. Esto es, se debería emplear la prueba que tenga la más alta probabilidad de descubrir una clase entera de errores.
- Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja, pero si se quieren combinar varias pruebas a la vez se pueden enmascarar errores, por lo que en general, cada prueba debería realizarse separadamente.

Para comprobar la calidad del producto realizado, se propone realizar pruebas de caja negra a cada caso de uso, para demostrar que cumplen con las precondiciones y poscondiciones especificadas en cada uno.

En la realización de esta prueba, se tendrá en cuenta la técnica de la Partición de Equivalencia, la cual permite examinar los valores válidos e inválidos de las entradas existentes en el software, además, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

3.1.2 Niveles de Pruebas

- ✓ **Pruebas unitarias:** Es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. La idea es escribir casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto. Estas pruebas aisladas proporcionan cinco ventajas básicas, fomentan el cambio, simplifica la integración, documenta el código, separa la interfaz del código y hace que los errores estén más acotados y sean fáciles de localizar.
- ✓ **Pruebas de Integración:** A partir del esquema del diseño, los módulos probados se vuelven a probar combinados para probar sus interfaces. Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos.

  Pruebas de Sistemas: El software ensamblado totalmente con cualquier componente hardware que requiera, se prueba para comprobar que se cumplen los requisitos funcionales. Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. A este tipo de pruebas donde se estudia el producto completo se les llama Pruebas de Sistema.
- ✓ **Pruebas de Aceptación:** Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado.

3.2 Métodos de Prueba

3.2.1 Pruebas de Caja Negra o Funcionales:

Pruebas que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene, sin tener en cuenta su funcionamiento interno. Se encargan de saber qué es lo que hace el software, pero sin dar importancia a cómo lo hace.

Algunas técnicas utilizadas en la prueba de caja negra son:

- ✓ **Partición de Equivalencia:** Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

- ✓ **Análisis de Valores Límite:** La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica nos lleva a elegir los casos de prueba que ejerciten los valores límite.

Para realizar este tipo de prueba se usara el requisito Adicionar Criterios el objetivo de este requisito es adicionarle los criterios a las entidades los cuales van a ser utilizados para las búsquedas de las mismas.

Nombre del Requisito	Descripción General	Escenario de Pruebas	Flujo del Escenario
Adicionar Criterios.	El sistema permite adicionar un nuevo criterio.	EP 1.1 Adicionar un criterio introduciendo los datos correctamente al presionar el botón **Aceptar.**	- Se presiona el botón **Adicionar**. - Se introducen los datos correctamente. - Se presiona el botón **Aceptar**. - Se adicionan los datos correctamente. - Se presiona el botón **Adicionar.**

		EP 1.2 Adicionar un criterio introduciendo los datos Inválidos al presionar el botón **Aceptar.**	- Se introducen los datos inválidos. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s)."
		EP 1.3 al presionar el botón **Cancelar.**	- Se presiona el botón **Adicionar.** - Se introducen los datos o no en los campos. - Se presiona el botón **Cancelar.**

3.2.2 Pruebas de Caja Blanca o Estructurales:

Se denomina cajas blancas a un tipo de prueba de software que se realiza sobre las funciones internas de un módulo. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las de caja blanca están dirigidas a las funciones internas. Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas.

Algunas técnicas de prueba de Caja Blanca son:

- ✓ **Prueba de Condición:** Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
- ✓ **Prueba de Flujo de Datos:** Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- ✓ **Prueba de Bucles:** Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.
- ✓ **Prueba del Camino Básico:** Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circula el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

Para realizar la prueba es necesario realizar primeramente el análisis de complejidad del algoritmo sobre el que se va a realizar la prueba, con el propósito de calcular los valores de la complejidad ciclomática. El algoritmo a analizar es el que se presenta a continuación:

```
public DetachedCriteria ponerAlias(String valor, DetachedCriteria criteria,
   Map<String, String> map) {1
  String aliasAnterior = "";1
  String alias = "";1
  StringTokenizer tk = new StringTokenizer(valor, ".");1
  int largo = tk.countTokens();1
  for (int j = 0; j < largo; j++) {2
   alias = tk.nextToken();3
        if(!map.containsValue(alias + ".")){4
   criteria = getBuscadorDAO().crearAlias(criteria,aliasAnterior + alias, alias);5
        map.put(alias, alias + ".");5
        }6
        aliasAnterior = alias + ".";6
  }7
  return criteria;8
 }9
```

Fig.25: Representación del algoritmo Poner Alias.

Después de este paso, es necesario representar el grafo de flujo asociado, en el cual se representan distintos componentes como es el caso de:

Nodo: Son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: Son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

Regiones: Son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

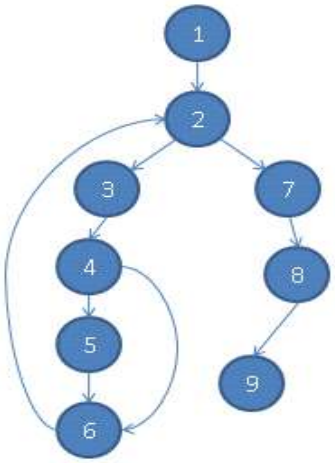

Fig.26: Grafo de Flujo asociado al algoritmo.

La complejidad ciclomática está basada en la teoría de grafos y nos da una métrica del software extremadamente útil. La complejidad se puede calcular de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como.

   V (G)=A-N+2

   Donde A es el n6mero de aristas del grafo de flujo y N es el número de nodos del mismo.
3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como.

   V (G) =P+ 1

   Donde P es el número de nodos predicado contenido en el grafo de flujo G.

Determinamos la complejidad ciclomática del grafo de flujo asociado al algoritmo.

V (G) = 3 regiones

V (G) = 10 aristas - 9 nodos +2 = 3

V (G) = 2 nodos predicado + 1 = 3

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 4, lo que significa que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

Camino básico #1:
1-2-7-8-9
Camino básico #2:
1-2-3-4-6-2-7-8-9
Camino básico #3:
1-2-3-4-5-6-2-7-8-9

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.
Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:
Descripción*:* Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
Condición de ejecución*:* Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
Entrada: Se muestran los parámetros que entran al procedimiento.
Resultados Esperados: Se expone el resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico # 1:
Descripción: Los datos de entrada cumplirán con los siguientes requisitos: el valor entrado es un string, el parámetro DatachedCriteria es de tipo criteria y Mapa es de tipo mapa.

Condición de ejecución: El valor es igual a" Nombre", Map es igual Null y Criteria es igual a object.

Entrada: Valor="Nombre", Map=Null, Criteria=object

Resultados Esperados: Se espera relacionar los objetos para conectarse a la base de datos.
No se pudo relacionar los objetos porque el atributo que buscábamos es de la entidad natal.

Caso de prueba para el camino básico # 2:
Descripción: Los datos de entrada cumplirán con los siguientes requisitos: el valor entrado es un string, el parámetro DatachedCriteria es de tipo criteria y Mapa es de tipo mapa.

Condición de ejecución: El valor es igual a "Nombre.id", Map es igual a id y Criteria es igual a object.

Entrada: Valor=”Nombre.id”, Map=id, Criteria=object

Resultados Esperados: Se espera relacionar los objetos para conectarse a la base de datos.
No se pudo relacionar los objetos porque ya la relación exista.

Caso de prueba para el camino básico # 3:
Descripción: Los datos de entrada cumplirán con los siguientes requisitos: el valor entrado es un string, el parámetro DatachedCriteria es de tipo criteria y Mapa es de tipo mapa.

Condición de ejecución: El valor es igual a “Nombre.id”, Map es igual a Null y Criteria es igual a object.

Entrada: Valor=”Nombre.id”, Map=Null, Criteria=object

Resultados Esperados: Se relacionan los objetos para conectarse a la base de datos.

3.3 Evaluación del modelo de diseño propuesto

Son varios los puntos de vista relacionados con la calidad del software. Desde metodologías hasta las distintas normas de calidad, que pueden estar orientados tanto a los procesos de desarrollo como a los productos de software. No es objetivo abundar sobre los temas de calidad, pero si desarrollar una evaluación del diseño propuesto del módulo Buscador Genérico.
“Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad del módulo, medidas que pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes.” (16)
Atributos de calidad que se abarcan:

- ✓ **Responsabilidad (Cohesión*)*:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

- ✓ ***Complejidad del mantenimiento:*** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

- ✓ ***Complejidad de implementación****:* Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

- ✓ ***Reutilización****:* Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.

- ✓ ***Acoplamiento****:* Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

- ✓ ***Cantidad de pruebas****:* Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas empleadas para evaluar la calidad del diseño de los componentes y su relación son las siguientes:

- ***Tamaño operacional de clase (TOC):*** Está dado por el número de métodos asignados una clase. Los atributos que afecta son la Responsabilidad, Complejidad de implementación y la Reutilización. De manera que mientras mayor sea el tamaño operacional de clase mayor será la Responsabilidad y Complejidad de implementación, mientras que su Reutilización disminuye.

Ver instrumentos y tabla de resultados en **Anexo 5** "Instrumento de medición de la métrica Tamaño operacional de clase (TOC)".

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

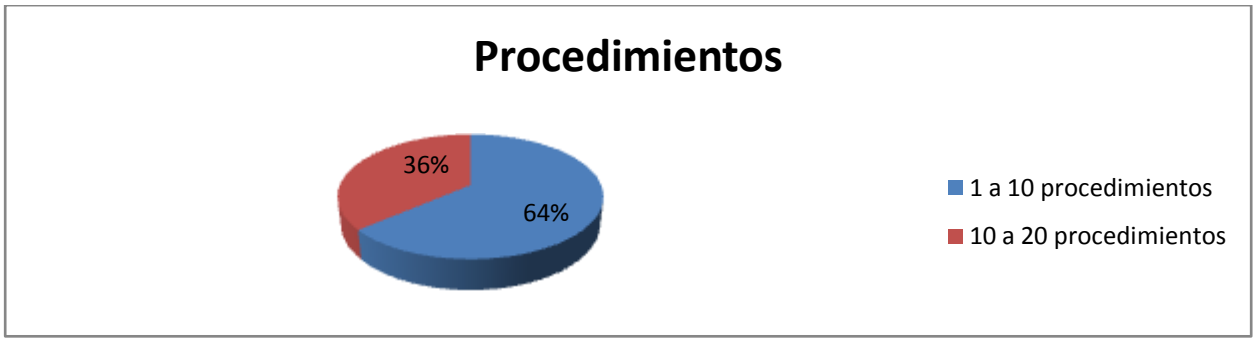

Fig.27: Representación de los resultados obtenidos en el instrumento, agrupados intervalos.

Para la observación de las gráficas donde se muestran los resultados obtenidos en el instrumento referirse al **Anexo 6** "Representación de la incidencia de los resultados de la evaluación de la métrica TOC."

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del componente Buscador Genérico tiene una calidad aceptable teniendo en cuenta que el 64% de las clases incluidas en este componente posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones.

Relaciones entre clases (RC): Esta dado por el número de relaciones de uso de una clase. Los atributos que afecta son el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas. De manera que mientras mayor sean las relaciones entre las clases mayor será el Acoplamiento, la Complejidad de mantenimiento y la Cantidad de pruebas, mientras que su Reutilización disminuye.

Ver instrumentos y tabla de resultados en **Anexo 7 "**Instrumento de medición de la métrica Relación entre Clases (RC)**"**.

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

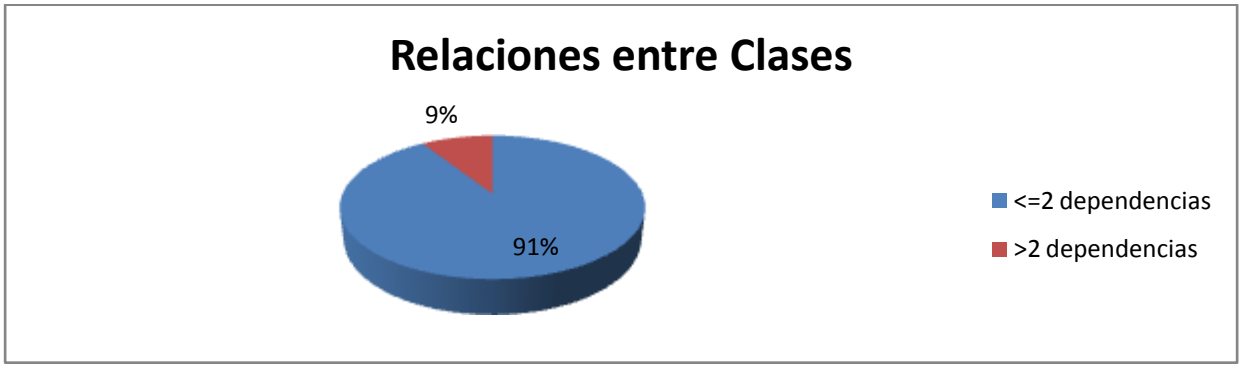

Fig.28: Representación resultados obtenidos en el instrumento agrupados en intervalos.

Para la observación de las gráficas donde se muestran los resultados obtenidos en el instrumento referirse al **Anexo 8 "**Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC)".

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del componente Buscador Genérico tienen una calidad

aceptable teniendo en cuenta que el 91% de las clases incluidas en este componente poseen menos de 2 dependencias de otras clases.

Conclusiones:

En el desarrollo de este capítulo se comienza con el análisis de diferentes aspectos como el significado de las pruebas de software, sus objetivos y alcance. Se realiza una descripción de los test de unidad, dentro de los cuales se analizaron los tipos de prueba: Caja Blanca y Caja Negra, la ejecución de las mismas y los resultados obtenidos, además se evaluó mediante métricas el diseño propuesto. Resulta importante explicar que los casos de pruebas presentados no son los únicos diseñados para las aplicaciones sino que el objetivo es plasmar la forma de la realización de los mismos.

Conclusiones

Durante el desarrollo del trabajo de diploma y el tiempo dedicado a la investigación, Diseño, implementación y prueba de la solución propuesta, se adquirieron conocimientos necesarios para el desarrollo del sistema y fundamentales para la formación profesional del autor. Luego de la realización de las actividades anteriores se logró:

- ✓ Obtener una valoración de las tendencias y tecnologías actuales que permitieran la correcta selección de entornos de desarrollo integrado, frameworks y lenguajes de programación.
- ✓ Seleccionar las herramientas necesarias para el desarrollo del trabajo correctamente.
- ✓ Estructurar, a partir de los Diagramas de componentes, el modelo de implementación del Buscador.
- ✓ Desarrollar y documentar los casos de prueba necesarios para verificar los algoritmos empleados.

Se confirma haber dado cumplimiento al objetivo general trazado desde el inicio del trabajo. La afirmación anterior se sustenta en el logro del Desarrollo del módulo Buscador Genérico en el proyecto SAGEB, lo cual garantiza mayor rapidez y mejor funcionamiento a la hora de realizar cualquier tipo de búsqueda de información.

Recomendaciones

Con la realización de este trabajo se recomienda:

- ✓ El componente Buscador genérico sea reutilizado en otros proyectos para generar búsquedas y reportes de una forma eficiente.
- ✓ Trabajar con más profundidad el tema de las conjunciones y disyunciones para mayor maniobrabilidad y poder realizar búsquedas más complejas.
- ✓ Agregarle Spring WebFlow a parte del ya existente Spring MVC para darle mayor capacidad de integración con el sistema.

Bibliografías

1. **Alfredo Sánchez Rodríguez, Frank Popa Sourd.** para el desarrollo de aplicaciones para la salud en Cuba. Aprobado. [En línea] 28 de 11 de 2007. [Citado el: 12 de 20 de 2009.]

2. **Salazar, Luis Antonio.** *Prolegómenos Sobre el Lenguaje de Modelado Unificado (UML).* [En línea] 2003. [Citado el: 17 de 12 de 2009.] [Disponible en] http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/prolego.pdf.

3Patrones de diseñoJoaquin Gracia[Disponible en] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

4. John Deacon. Model-View-Controller (MVC) Architecture. [Online] [Disponible en] http://www.jdl.co.uk/briefings/mvc.pdf.

5. Joseph Bergin. Building Graphical User Interfaces with the MVC pattern. [Online] [Disponible en] http://csis.pace.edu/ bergin/mvc/mvcgui.html..

6. Steve Burbeck. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). [Online] [Disponible en] http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

7. **Larman., Craig.** UML y Patrones. [Online] [Cited: 01 5, 2010.] [Disponible en] http://bibliodoc.uci.cu/pdf/reg00061.pdf.

8. Patrones de Diseño en aplicaciones Web con Java J2EE. [En línea] 2005. [Citado el: 6 de 01 de 2010.] [Disponible en] http://java.ciberaula.com/articulo/diseno_patrones_j2ee/.

9. Solución de OutOfMemory en Tomcat 6.0 para Windows. [Online] Febrero 3, 2010. [Disponible en] http://www.telepieza.com/wordpress/2009/06/12/solucion-de-outofmemory-en-tomcat-60-para-windows/.

10. Grupo de investigación Kybele & Universidad Rey Juan Carlos. [En línea] Departamento de Lenguajes y Sistemas Informáticos II. [Citado el: 12 de 03 de 2010.] [Disponible en] http://kybele.escet.urjc.es/documentos/ISG/Estructurado/%5BISG-2006-07%5DRequisitosSoftware.pdf.

11. *Universidad de San Carlos de Guatemala. Facultad de Ingeniería.* [En línea] Escuela de Ciencias y Sistemas. Diagrama de Casos de Uso., 2005. [Citado el: 20 de 03 de 2010.] [Disponible en] http://www.cif.acuareladelsur.org/tutoriales/casosdeuso.pdf.

12. **Ivar Jacobson, Grady Booch, James Rumbaugh.** El proceso unificado de desarrollo desoftware (volumen 1). [En línea] [Citado el: 23 de 03 de 2010.] [Dispopnible en] http://bibliodoc.uci.cu/pdf/reg00060.pdf.

13. **Booch Grady, Jacobson Ivar, Rumbaugh James.** El Proceso Unificado de Desarrollo de Software. [En línea] Adison Wesley, 2000. [Citado el: 28 de 03 de 2010.]

14. **Torres, Patricio Letelier.** Desarrollo de Software Orientado a Objeto usando UML. Departamento Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. [En línea] [Citado el: 8 de 04 de 2010.]

15. **Pressman, Roger.** Ingeniería de Software. Un enfoque práctico. [Online] 2008. [Cited: 04 19, 2010.]

16. **Bauer, Gaving King y Christian.** *Hibernate in Action[Disponible en] http://www.manning.com/bauer. .*

17. *Documentación oficial de Hibérnate[Disponible en] http://www.hibernate.org/5.html. .*

18. *Hibérnate y Joins con la clase Criteria[Disponible en] http://www.hibernate.org/. .*

19. *Desarrollando una aplicación Spring Framework MVC paso a paso[Disponible en]http://www.davidmarco.es/tutoriales/SpringFrameworkMVC.html. .*

20. **González, Héctor Suárez.** *Manual de Hibérnate,[Disponible en] http://www.javahispano.org/articles.article.action?id=82. .*

21. **Smith, Aitor García y Glen.** *Guía del autoestopista a Hibérnate[Disponible en] http://www.javahispano.org/articles.article.action?id=80. .*

22. *Libros en pantalla de SQL Server 2008. Integridad de los datos, [Disponible en]http://msdn.microsoft.com/eses/. .*

23. *El proceso unificado de desarrollo desoftware (volumen 1),[Disponible en] http://bibliodoc.uci.cu/pdf/reg00060.pdf. .*

24. *kynetia.Tipos de Pruebas [Disponible en]http://www.kynetia.es/calidad/tipos-de-pruebas.html. .*

25. *Pruebas (n.d.) [Disponible en] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php. .*

26. *Free Download Manager. Paradigma visual para UML. [Disponible en] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.*

27. *Etapa de pruebas . [Disponible en] http://fabian-quintosemestre.blogspot.com/2007/09/etapa-de-pruebas.html.*

28. *mozilla.org. (1998–2009). [Disponible en] from http://www.mozilla.org/about/.*

29. *TortoiseSVN. (n.d.). [Disponible en] from Un cliente de Subversion para Windows: https://forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf.*

Anexos

Anexo 1 “Descripción de los casos de usos”.

Caso de Uso:	Buscar información
Actores:	Módulo.
Resumen:	El caso de uso se inicia cuando el Módulo desea buscar una determinada información, el sistema responde esta petición mostrando los resultados en una tabla resultante.
Precondiciones:	
Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El módulo solicita la búsqueda de información.	2. El sistema muestra un formulario con los campos a completar.
3-El módulo indica los criterios de búsqueda para buscar la información.	4-El sistema verifica que el módulo haya indicado algún criterio de búsqueda.
	5-El sistema valida que no queden campos vacíos.
	6-El sistema busca la información que coincida con los criterios de búsqueda indicados.
	7-El sistema muestra una tabla ordenada según los criterios con los resultados de la búsqueda realizada.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si el módulo no indicó ningún criterio de Búsqueda, el sistema la busca toda.
	6.1 Si el sistema no encuentra la información muestra la tabla vacía.

Tabla #2: Descripción del Caso de Uso: Buscar información

Caso de Uso:	Gestionar Criterios
Actores:	Módulo
Resumen:	El caso de uso se inicia cuando el Módulo desea introducir un nuevo criterio al sistema modificar o eliminar un criterio ya existente.
Precondiciones:	
Flujo Normal de Eventos	
	El sistema muestra un formulario con las opciones de Insertar y modificar o eliminar criterios.
1) El módulo selecciona la opción: a) Insertar criterios. Sección Insertar criterios. b) Modificar criterios. Sección Modificar criterios. c) Eliminar criterios. Sección Eliminar criterios.	
Sección Insertar Criterios	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra un formulario con los campos a completar.
2. El Módulo introduce los criterios y selecciona la opción aceptar.	3. El sistema verifica que el criterio no esté registrado.
	4. El sistema guarda en memoria los datos insertados.
Flujos Alternos	
Sección Modificar Criterio	
Acción del Actor	**Respuesta del Sistema**

	1. El sistema muestra un formulario con el listado de los criterios.
2. El módulo selecciona un criterio del listado y presiona el botón aceptar.	3-El sistema modifica el criterio seleccionado.
	4-El sistema actualiza los datos y guarda en memoria los datos.

Tabla #3: Descripción del Caso de Uso: Gestionar Criterios

Anexo 2 "Diagramas de clase por capas"

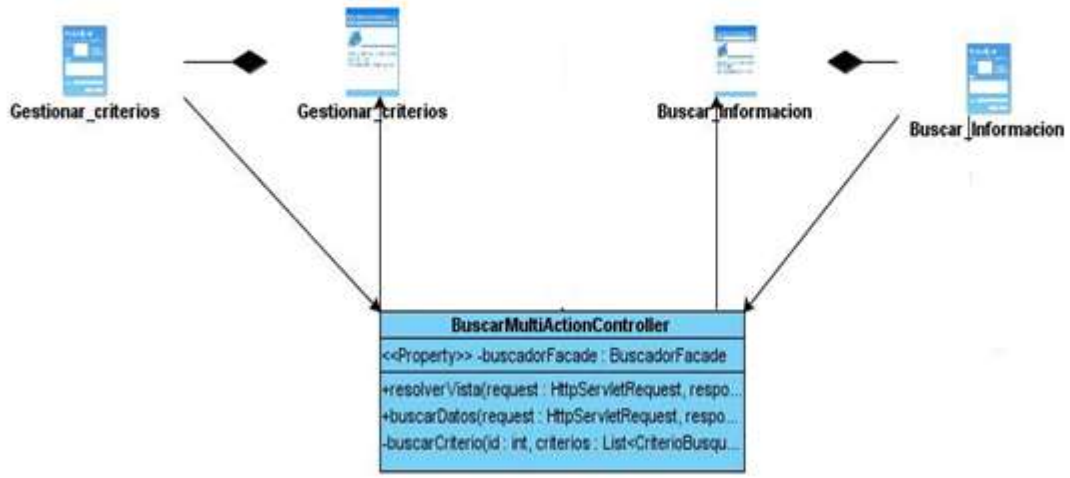

Fig.13: Capa de Presentación

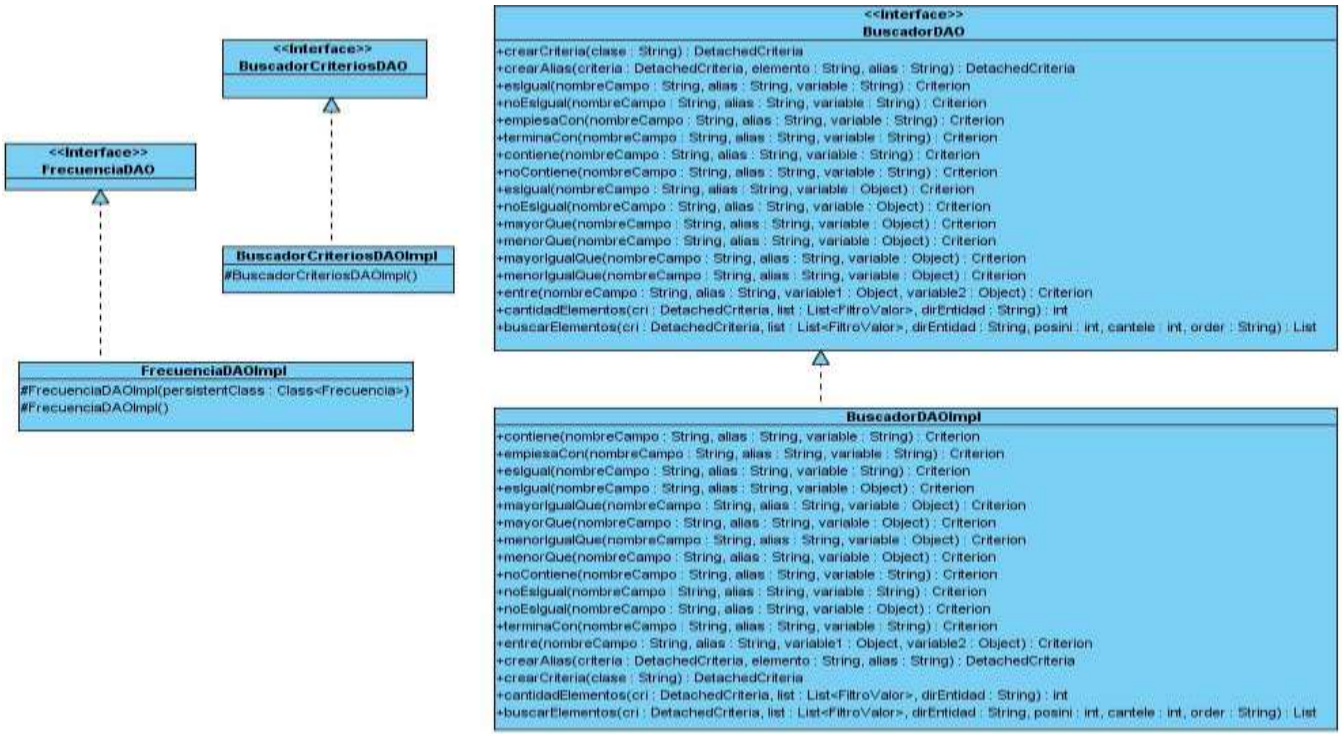

Fig.14: Capa de acceso a Dato

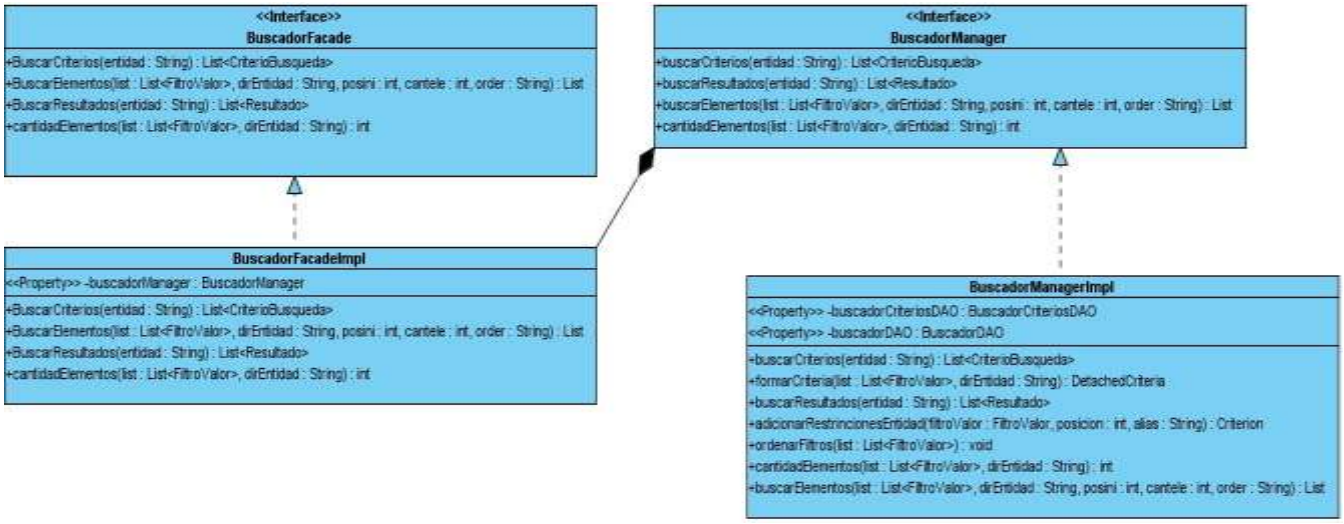

Fig.15: Capa del Negocio

Anexo 3"Diagramas de Secuencia"

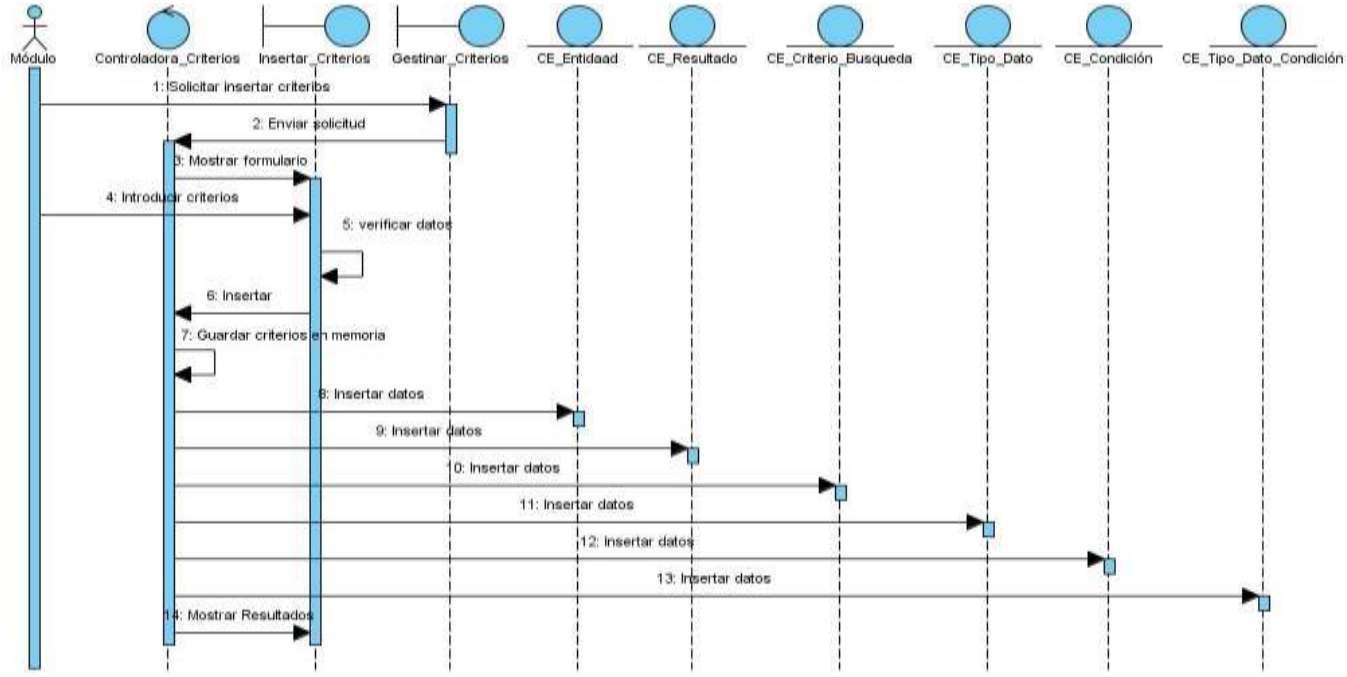

Fig.17: Caso de uso Gestionar Criterios flujo de evento "**Insertar Criterios"**

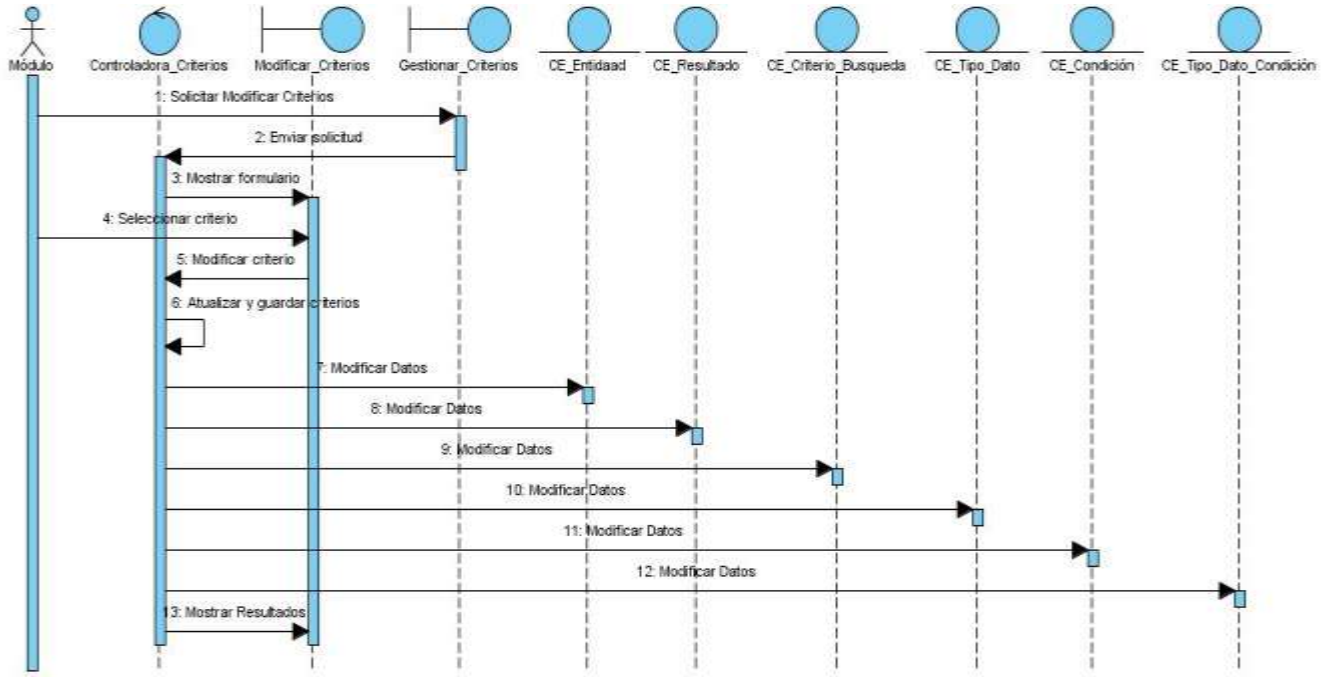

Fig.18: Caso de uso Gestionar Criterios flujo de evento "**Modificar Criterios"**

Anexo 4*:* Resultados de la evaluación de la métrica TOC.

No	Clases	Procedimientos	Responsabilidad	Complejidad	Reutilización
1	BuscadorFacadeImpl	7	Baja	Baja	Alta
2	BuscadorManagerImpl	13	Media	Media	Media
3	BuscadorDAOImpl	18	Media	Media	Media
4	BuscadorCritDAOImpl	3	Baja	Baja	Alta
5	Entidad	8	Baja	Baja	Alta
6	Resultado	8	Baja	Baja	Alta
7	CriterioBusqueda	14	Media	Media	Media
8	TipoDato	9	Baja	Baja	Alta
9	Condición	6	Baja	Baja	Alta
10	FiltroValor	12	Media	Media	Media
11	AyudaFiltValor	5	Baja	Baja	Alta

Tabla #8: Resultados de la evaluación de la métrica TOC

Anexo 5: Representación de la incidencia de los resultados de la evaluación de la métrica TOC.

Métrica TOC	Alta	Media	Bajo
Responsabilidad	0%	36%	64%
Complejidad	0%	36%	64%
Reutilización	64%	36%	0%

Tabla#9: Representación de la métrica TOC en los atributos Responsabilidad, Complejidad y Reutilización.

Anexo 6: Resultados de medición de la métrica Relación entre Clases (RC).

No	Clases	Cantidad de Relaciones	Acoplamiento	Complejidad de Mantenimiento	Cantidad de Pruebas	Reutilización
1	BuscadorFacadeImpl	1	Baja	Baja	Baja	Alta

2	BuscadorManagerImpl	3	Alta	Alta	Alta	Baja
3	BuscadorDAOImpl	1	Baja	Baja	Baja	Alta
4	BuscadorCritDAOImpl	1	Baja	Baja	Baja	Alta
5	Entidad	2	Media	Media	Media	Media
6	Resultado	1	Baja	Baja	Baja	Alta
7	CriterioBusqueda	2	Media	Media	Media	Media
8	TipoDato	2	Media	Media	Media	Media
9	Condición	1	Baja	Baja	Baja	Alta
10	FiltroValor	1	Baja	Baja	Baja	Alta
11	AyudaFiltValor	1	Baja	Baja	Baja	Alta

Tabla #10: *Resultados de la evaluación de la métrica RC.*

Anexo 7: Representación de la incidencia de los resultados de la evaluación de la métrica RC.

Métrica RC	Alta	Media	Bajo
Acoplamiento	9%	27%	64%
Complejidad de Mantenimiento	9%	27%	64%
Cantidad de Pruebas	9%	27%	64%
Reutilización	64%	27%	9%

Tabla #11: Representación de la métrica TOC en los atributos Acoplamiento, Complejidad de Mantenimiento Cantidad de Pruebas y Reutilización.

Glosario de Términos

API: Una interfaz de programación de aplicaciones (Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

BNC: Banco Nacional de Cuba

CSS: *Cascading Style Sheets* (Hojas de estilo en cascada), ofrece la posibilidad de separar la estructura de un documento estructurado escrito en HTML o XML de su presentación.

Framework: Denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

GoF: Acrónimo de de los vocablos en inglés: Gang of Four, que en español equivalen: a Grupo de los Cuatro o Banda de los Cuatro. Este es el nombre con el que se conoce comúnmente a los autores del libro Design Patterns, referencia en el campo del diseño orientado a objetos, estos son: Erich Gamma, Richard Helm , Ralph Johnson y John Vlissides.

Herramientas CASE: CASE es el acrónimo de los vocablos en inglés: Computer Aided Software Engineering, que en español equivalen a: Ingeniería de Software Asistida por Computadoras. Estas herramientas son útiles en todos los aspectos del ciclo de vida de desarrollo del software: realización de modelos y diagramas, diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

HTML: Acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. HTML es una aplicación de SGML conforme al estándar internacional ISO 8879

Java Server Page (JSP): Es una tecnología que nos permite mezclar código HTML estático con código HTML generado dinámicamente.

Glosario de Términos

Multiplataforma: Se utiliza el término para denominar a los programas, lenguajes de programación u otra clase de software que pueden brindar sus prestaciones funcionando sobre diversas combinaciones de hardware y software.

MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario.
Navegador: Un navegador web o cliente HTTP, es un programa que permite interpretar la información y el código que contiene una página web (esté alojada en un servidor dentro de la World Wide Web o en uno local) y presentarla de manera legible a los usuarios.

Open-source: Práctica de desarrollo de software que promueve el acceso al código fuente de los sistemas computacionales.

SABIC: Sistema Automatizado para la banca internacional de comercio.

Servlets: Son objetos que corren dentro del contexto de un contenedor de Servlets (ejemplo Tomcat) y extienden su funcionalidad. También podrían correr dentro de un servidor de aplicaciones. Un Servlet es un programa que se ejecuta en un servidor.

SQL: *Structured Query Language* (lenguaje de consulta estructurado), es un lenguaje computacional de Bases de Datos diseñado para la recuperación y gestión en sistemas de bases de datos relacionales.

UML: Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

XML: *Extensible Markup Language* (lenguaje extensible de marcado), es un metalenguaje extensible de etiquetas creado por el World Wide Web Consortium (W3C).

XMLHttpRequest: *Extensible Markup Language / Hypertext Transfer Protocol*, es empleado para realizar peticiones HTTP y HTTPS a servidores WEB. Para los datos transferidos se usa cualquier codificación de las siguientes: texto plano, XML, JSON (Java Script Object Notation) y HTML.