Universidad de las Ciencias Informáticas
Facultad 2

Título: Directorio de Personas UCI

TRABAJO DE DIPLOMA

PRESENTADO PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Autor(es): Marialis Sanamé Mendoza

Armando R.Machado González

Tutor(es): Ing. Alberto Tamayo Ramos

Co-tutor: Ing. Yunaisi Rente Vazquez

"Año 50 de la Revolución

Ciudad de La Habana, Cuba. Junio del 2008"

DECLARACIÓN DE AUTORÍA

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

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

_____
Marialis Sanamé Mendoza
Autora

_____
Armando R. Machado González
Autor

_____
Ing. Alberto Tamayo Ramos
Tutor

AGRADECIMIENTOS

Marialis

A mis padres **Luis y Marisel** que son mis dos almas gemelas, aunque nos separa un poco de distancia, su corazón, fe y apoyo estuvieron a mi lado toda mi vida.

A mi mejor amigo y hermano, **Luisbel**, por todo su apoyo y confianza.

A mis abuelos maternos, **Raúl y Julia**, por poner toda su fe y amor en mí.

A la memoria de mis abuelos paternos, **Eulogio y María**, pues su amor y su recuerdo fueron fuerzas espirituales que me impulsaron cada día para realizar este sueño.

A toda mi familia, en especial mis tíos **Mercedes, Papito, Manolo y Paulita** por todo el apoyo y confianza y amor que me han que me han brindado.

A mi pareja y amigo por apoyarme y darme fuerzas durante todo este tiempo, gracias **Delin.**

A todas mis amistades que estuvieron junto a mí en todo momento, **Made**, **Iralys** , **Islema, Gladys**, **Rita, Alberto, Yami, Halena** y demás que son muchísimas, y a todas aquellas que aunque la distancia nos separa también estuvieron junto a mi, **Orly, Annalie, Lizandra, Franklin, Lisi, Elvis, Eriannis e Iliannis.**

A mi compañero de tesis **Armando**, a **Yanio y Pedro** por ayudarme en la realización de este trabajo.

A mi tutor **Alberto** por todo su apoyo.

A mi cotutora **Yunaisi Rente** por todo su tiempo y dedicación.

A mis compañeros de aula **2503** por todo su cariño, es un honor haber compartido con todos ustedes tanto tiempo.

A mi profesora **Yadilka** por ser guía en todo momento.

A Fidel y a la Revolución por ser inspiradores de este proyecto al que le estoy profundamente agradecida.

AGRADECIMIENTOS

Armando

A mis padres, a mis hermanos y a mi abuela por darme su apoyo y amor durante toda mi carrera, en especial a mi mamá**, Esther**, que ha sido mi guía en toda mi vida.

A mi compañera de tesis **Marialis, a Yanio, Pedro, Orley y Yoandris** por ayudarme en la realización de este trabajo.

A mis compañeros de grupo **2503** por todo su apoyo durante estos 5 años de carrera.

A mi tutor **Alberto** que con su exigencia y apoyo hizo posible la realización de este trabajo.

A Fidel y a mi Revolución por haberme permitido estudiar en un proyecto como este.

DEDICATORIA

A nuestros padres y hermanos.

"Sólo es posible avanzar cuando se mira lejos. Solo cabe progresar cuando se piensa en grande."

José Ortega y Gasset

RESUMEN

En la Universidad de las Ciencias informáticas (UCI) fue desarrollado entre otros sistemas el Directorio de Búsquedas, que posibilita la automatización del proceso de búsquedas para todo el personal del centro. Este es un proceso dinámico y permanente de gran importancia, caracterizado por el alto nivel técnico y altas demandas de accesibilidad en la universidad.

Sin embargo, estas posibilidades se ven dificultadas por la poca flexibilidad, eficiencia y lentitud con que cuenta el sistema, siendo uno de los sitios de mayor intercambio con el personal debido a su creciente aumento de cada año.

Ello impone la necesidad de la creación de un nuevo directorio que brinde al personal del centro una mayor satisfacción al interactuar con la información que se busca en el sistema, basándose en la arquitectura que aplica la universidad, Arquitectura Orientada a Servicios (SOA), que permitirá un mejor manejo de la información, rapidez y un óptimo funcionamiento del mismo.

En el presente trabajo se realiza el análisis, diseño e implementación del Directorio de Búsqueda de la UCI.

Para ello se estructura en 5 capítulos donde se recoge la fundamentación teórica del trabajo, el modelo del dominio, los requisitos del sistema, la descripción de la solución propuesta y el estudio de factibilidad.

Se propone como solución una aplicación Web, basada en la Arquitectura SOA, que consuma los diferentes servicios web contribuyendo la adaptación de este nuevo sistema a la arquitectura de la universidad y a la mejora del proceso de búsqueda de información de personas.

PALABRAS CLAVE

SOA, servicios web, directorio.

ÍNDICE

INTRODUCCIÓN

En los últimos años, las aplicaciones informáticas están encaminadas cada vez más hacia una infraestructura de soporte integrado entre las operaciones de las empresas y sus clientes, donde el resultado de este proceso, ha sido la creación y mantenimiento de un número considerable de aplicaciones en el interior de las entidades, cada una responsable de sus propias tareas.

Crear estas aplicaciones, requiere en muchos casos de funcionalidades ya implementadas como parte de otros sistemas.

La Universidad de las Ciencias Informáticas (UCI) desde su surgimiento en el año 2002, tiene como principal objetivo el desarrollo de la industria cubana del software. Para su desarrollo y evolución fomentó la creación de diferentes estrategias y servicios adicionales, necesarios e indispensables para la satisfacción y buen funcionamiento de la misma.

Una de las estrategias es facilitar su integración y funcionamiento con la construcción de servicios. Estos servicios se encargan de brindar un conjunto de funcionalidades bien definidas a las aplicaciones que las requieran. De esta manera, una aplicación final simplemente orquestaría la ejecución de estos servicios, añadiendo su lógica particular y la presentación de una interfaz al usuario final.

En la actualidad el proceso de búsqueda de información referente a las personas en la UCI se realiza a través de un Directorio de Búsqueda de Personas, desarrollado sobre la Plataforma .NET, convirtiéndose en una de las aplicaciones más explotadas en la universidad.

Sin embargo este sistema ha venido presentando algunos problemas imposibilitando su eficiente funcionamiento y rapidez. Entre ellos:

- El proceso de búsqueda de la información de las personas es lento e impreciso.
- No permite filtrar las búsquedas de información por muchos campos, lo que resta flexibilidad al proceso de búsqueda de personas.
- No se ajusta a la Arquitectura Orientada a Servicios (SOA), que actualmente se encuentra en la etapa inicial de su desarrollo en la universidad.

- No está desarrollado en software libre, lo que forma parte de la estrategia de migración de todas las aplicaciones a este tipo de software de la universidad.
- La interfaz del usuario es muy poco amigable, debido a que el entorno de trabajo es completamente de texto.

La estrategia que persigue la universidad es adaptar este sistema a la arquitectura SOA.

SOA pretende ser un modelo sobre el cual las organizaciones puedan construir una arquitectura corporativa de referencia, que contemple y favorezca la interacción de aplicaciones. Para esto, se pretende que el sistema utilice los servicios componentes de la arquitectura que garanticen interfaces independientes de la implementación y transparentes a la ubicación de los servicios web.

A raíz de esto surge la necesidad de desarrollar un nuevo directorio que permita utilizar servicios web para el desarrollo de sus funcionalidades enfocándolo hacía las bases de la arquitectura SOA, estableciéndose como **problema científico** lo siguiente:

¿Cómo mejorar el proceso de búsqueda de información de personas?

Teniendo como **objetivo general** el desarrollo de un nuevo directorio de personas en software libre.

La utilización de esta estrategia en el directorio de personas traerá consigo numerosos beneficios. Entre ellos el aumento de la flexibilidad y el dinamismo en el momento de buscar cualquier información de un usuario determinado, debido a que la reutilización de los diferentes servicios web existentes en la universidad reduce los costes de mantenimiento y tiempo de actualización de la aplicación.

En correspondencia con lo formulado, el **objeto de estudio** estará enmarcado en los procesos de búsquedas de información.
Enfocando el **campo de acción** en el desarrollo del proceso de búsqueda de información de personas.

La **idea a defender** es la implementación de un nuevo Directorio de Búsqueda de personas en software libre, basado en SOA que le permita darle flexibilidad, rapidez y eficiencia al proceso de búsqueda en la universidad.

Para darle cumplimiento al objetivo de este trabajo nos planteamos un grupo de tareas de investigación:

1. Realizar un análisis sobre la arquitectura (SOA) a desarrollar en la UCI.
2. Realizar un análisis de los sistemas similares existentes en el mundo y en Cuba con el objetivo de obtener las buenas prácticas y principales deficiencias.
3. Realizar el modelamiento de los flujos de trabajo análisis, diseño e implementación de la aplicación.
   - Modelo de Análisis
   - Modelo de Diseño.
   - Modelo de Implementación.
4. Realizar un análisis de los diferentes servicios web implementados en la UCI.
5. Consumir los servicios web necesarios para las funcionalidades del sistema.
6. Realizar la implementación del sistema con las características definidas sobre la base de la arquitectura SOA.
7. Realizar el estudio de factibilidad para el sistema final.

La propuesta de este trabajo contribuye a mejorar el proceso de búsqueda de información del directorio para el logro de un eficiente servicio, garantizando con el empleo de esta nueva estrategia a desarrollar, una mejor alternativa para el intercambio de información entre un sistema altamente escalable y los usuarios del mismo.

El contenido del presente trabajo se estructura en 5 capítulos:

Capítulo 1. Fundamentación Teórica: Se exponen los fundamentos generales que sirven de soporte teórico en la solución del problema. Se analizan las herramientas y lenguajes de programación idóneas para el desarrollo del Directorio de Personas. Se plantea la metodología a emplear en el desarrollo del mismo. Se realiza un estudio de la arquitectura a emplear en la propuesta de sistema y un análisis de algunos sistemas similares a este a nivel internacional como nacional.

Capítulo 2. Características del Sistema. Se refleja la descripción de los principales procesos involucrados en el objeto de estudio. Se definen los conceptos en un Modelo del Dominio para capturar correctamente los requisitos y poder construir un sistema consistente así como las características y

funcionalidades del sistema. Se especifican y se detallan los requisitos funcionales y no funcionales del sistema. Además incluye la descripción de los actores y los casos de uso del sistema.

Capítulo 3. Análisis y Diseño: Se describe los diagramas de clases de Aplicaciones Web del diseño y los diagramas interacción del mismo para cada realización de los casos de uso. Muestra la descripción de cada una de las clases.

Capítulo 4. Implementación: Se plantea la construcción de la solución propuesta reflejando la transformación de los elementos del modelo del diseño en términos de componentes a través de los diagramas de componentes y el diagrama de despliegue.

Capítulo 5. Estudio de factibilidad: Se realiza el estudio de factibilidad económica para este sistema, en el que se determina si es factible o no el desarrollo del software propuesto, analizando los diferentes criterios que influyen en el cálculo del esfuerzo, tiempo de desarrollo y costo del proyecto.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se realiza una investigación acerca de la Arquitectura Orientada a Servicios (SOA) así como una profunda investigación de diferentes sistemas similares tanto en el ámbito internacional como nacional .Se describen además las tecnologías actuales, herramientas y lenguajes de desarrollo utilizados para el análisis, diseño e implementación del sistema sobre las cuales se apoya la propuesta.

1.2 La Arquitectura Orientada a Servicio (SOA).

SOA es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario. (1)

Proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de servicios web (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar una arquitectura SOA utilizando cualquier tecnología basada en servicios, por ejemplo, REST[1] o XML-RPC[2].

La complejidad de una arquitectura SOA podría variar mucho, dependiendo de la cantidad de servicios y las funcionalidades de estos, así, se pueden obtener aplicaciones que utilicen servicios de negocio, para su ejecución.

1.2.1 Elementos básicos que conforman SOA.

- **Proveedores de Servicios.**

[1] Son las siglas en inglés de *Representational State Transfer*. Es un estilo de arquitectura y no un estándar definido. REST se basa en las URIs para identificar recursos, los cuales se retornan en XML puro. Debe aclarase que REST es una manera simplificada de ejecutar métodos remotos.

[2] Es un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. Tiene otras aplicaciones entre las que destaca su uso como estándar para el intercambio de datos entre diversos software con lenguajes privados como SOAP.

- **Consumidores de Servicios.**
- **Bus Empresarial de Servicios.**

En realidad todo componente dentro de una organización puede ser tanto proveedor como consumidor de servicios. Todos los servicios interactúan entre si a través de un Bus Empresarial de Servicios(o ESB por sus siglas en Inglés). (2)

Existen otros elementos como:

- **Cliente-Proveedor.**

Es el componente que invoca un servicio provisto por un proveedor.

- **Concepto de Servicio.**

Es una unidad de trabajo realizada por un componente de software a fin de conseguir un resultado específico. El servicio debe ser alcanzable por parte de los consumidores a través de una interfaz programática.

- **Utilización de Servicios Web.**

Los servicios web son un conjunto de estándares que definen un protocolo de invocación remota de servicios, basados en HTML y XML.

- **ESB.**

Se basa en la mejor práctica de patrones de diseño para integración de aplicaciones**.**

1.2.2 ¿Qué es un servicio en SOA?

Es una función de aplicación empaquetada como un componente reutilizable para ser usado en un proceso de negocio. Proporciona información o facilita el cambio de datos de negocio de un estado válido y consistente a otro. A través de protocolos bien definidos, pueden ser invocados diferentes servicios web de manera que se hace hincapié en la interoperabilidad y en la trasparencia de localización. (3)

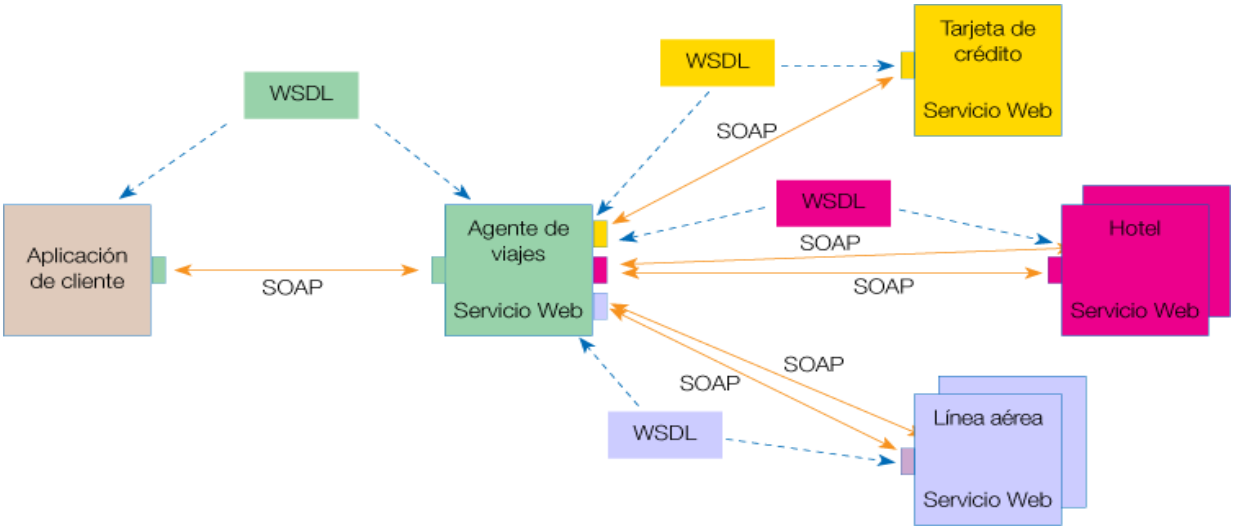

Figura 1.1 Los servicios web en funcionamiento.

1.2.3 Mensajes en un ambiente SOA.

El mensaje es el contenido de la invocación del servicio que lleva la información propia del negocio necesaria para la realización de las operaciones atadas al mismo.

- Concepto y uso de interfaces.

Un aspecto importante dentro de la arquitectura SOA es mantener una adecuada separación entre la implementación de un servicio y su interfaz. La interfaz de un servicio define la forma en que este puede ser invocado a través preferiblemente de un protocolo estándar como un servicio web.

- Relación Interfaz-Componente-Mensaje.

El componente brinda una funcionalidad (servicio) a través de una interfaz en la cual se intercambia un mensaje que contiene la información de negocio necesaria para la realización de una tarea específica.

1.2.4 SOA, publicación en directorios.

El objetivo de una arquitectura SOA en un directorio, es proveer de la transparencia en la localización del servicio Web, es decir, de la posibilidad de utilizar un determinado servicio que se encuentre en cualquier lugar, sin la necesidad de tener que modificar el código existente.

En el nivel más alto de la Arquitectura Orientada a Servicio se pueden encontrar tres componentes:
− El servicio: Cada servicio tiene una funcionalidad a la que pueden acceder el resto de servicios y clientes.
− El directorio: El directorio tiene información sobre los servicios y la funcionalidad de estos. Y también tiene la información de cómo se puede acceder a cada servicio.
− El cliente: Usa el directorio para localizar servicios y poder usar su funcionalidad. Un cliente puede ser otro servicio que quiere acceder o utilizar la funcionalidad que aportan otros.

Existen además tres colaboraciones entre los componentes, que pueden llegar a ser una de las partes más importantes de la arquitectura:
− Localización de servicios: Los clientes potenciales de los servicios localizan estos por medio del directorio. El directorio aporta a los clientes la información sobre como encontrar un servicio.
− Publicación de servicios: Un componente publica un servicio, haciéndolo disponible a los clientes a través del directorio.
− La comunicación entre los servicios y el cliente: El cliente hace peticiones al servicio a través del protocolo de red especificado en la información del servicio que tiene el directorio. El servicio recoge la petición del cliente y le retorna la información pedida. (4)

En la universidad, se cuenta con un directorio en el cual se encuentra la publicación de todos los servicios implementados, el Directorio de Servicios Web UCI, desde el cual las aplicaciones pueden acceder al mismo y utilizar aquellos que sean necesarios para sus funcionalidades.

Beneficios para el negocio en una aplicación

− Eficiencia. Transforma los procesos de negocio en servicios compartidos con un menor coste de mantenimiento.
− Capacidad de respuesta. Rápida adaptación y despliegue de servicios, clave para responder a las demandas de clientes y usuarios.
− Adaptabilidad. Facilita la adopción de cambios añadiendo flexibilidad y reduciendo el esfuerzo.

Beneficios Tecnológicos

− Reduce la complejidad gracias a la compatibilidad basada en estándares frente a la integración punto a punto.
− Reutiliza los servicios compartidos que han sido desplegados previamente.
− Integra aplicaciones heredadas limitando así el coste de mantenimiento e integración.
− Beneficios en el desarrollo, ya que las aplicaciones son reutilizables, más fácil de mantener y tienen la capacidad de ampliación de las funcionalidades del sistema, exponiéndolas de una forma segura.

1.2.5 ¿Por qué usar SOA?

Existen varias razones para adoptar un enfoque SOA, y más concretamente un enfoque SOA basado en servicios web.

- Reutilización.

El factor fundamental en el cambio a SOA es la reutilización de los servicios de negocio. Las funciones de negocio pueden ser expuestos como servicios web y ser reutilizadas para cubrir nuevas necesidades del negocio.

- Interoperabilidad.

El objetivo de una arquitectura débilmente acoplada es que los clientes y servicios se comuniquen independientemente de la plataforma en que residan. Los protocolos de comunicación con servicios web son independientes de la plataforma, lenguaje de codificación y sistema operativo por lo que facilitan la comunicación con los socios del negocio.

- Escalabilidad

Como los servicios de SOA están débilmente acoplados, las aplicaciones que usan esos servicios escalan fácilmente .Esto es debido a que existe muy poca dependencia entre las aplicaciones clientes y los servicios que usan.

- Flexibilidad

Es otra de las características que proporciona el acoplamiento débil entre los servicios. Cualquier cambio en la implementación de uno de ellos no afectaría al resto.

- Eficiencia de coste.

Las arquitecturas SOA se basan en la exposición de servicios ya existentes para ser reutilizados. Al usar servicios web se reutiliza la infraestructura web existente virtualmente, por lo que se limita considerablemente el coste.

1.2.6 Los Directorios de Búsquedas en el mundo.

La necesidad de mejorar la interacción con el usuario en el contexto de las búsquedas en Internet deviene imperativa en los últimos tiempos ante la relevancia de la Web como medio de difusión del conocimiento, la multiplicación de la información existente en la misma, y la utilización que en la sociedad actual, se hace de la información como bien competitivo. La investigación en este campo tiende a la estructuración de la información recuperada, a la presentación facetada de los resultados mediante interfaces dinámicas que integren consulta y presentación y que soporten el hipertexto y los mapas visuales de representación de la información , permitiendo al usuario obtener una imagen del sistema con la que interactuar para localizar su tema de interés, con el menor esfuerzo mental tanto a la hora de plantear su necesidad de información como al interpretar los resultados obtenidos.

A continuación se caracterizan algunos sistemas de búsquedas empleados en diferentes lugares en el mundo especificando sus éxitos y problemas de cada uno de ellos.

KartOO

Uno de los ejemplos más populares de la investigación aplicada en este campo, se encuentra precisamente en la propuesta del meta buscador KartOO, cuya interfaz integrada permite al usuario expresar su consulta mediante una búsqueda gráfica en los mapas de resultados ofrecidos por la herramienta.

KartOO dispone de una tecnología de búsqueda de información y de representación visual y el meta buscador es solamente el escaparate de una serie de productos que se adaptan a todos los tipos de fuentes (portales, bases de datos, servidores de archivos, etc.) y de gestión de conocimientos.

Al realizar una búsqueda los resultados se presentan en forma de mapa en el centro de la página, en la parte superior de la misma aparece un icono para acceder a la página principal de KartOO, la ayuda,

borrar la búsqueda y el cuadro de búsqueda. En este margen superior junto al cajetín aparecen unos iconos poco intuitivos que sirven para activar las siguientes funciones:

- Búsqueda únicamente en español.
- Búsqueda en todas las lenguas.
- Búsqueda simple.
- Búsqueda avanzada
- Historial: Muestra las búsquedas que el usuario ha realizado en KartOO.

La detección de navegadores y los formularios están programados en PHP porque es simple y eficaz. (5)

Oracle Internet Directory

Los servicios de directorio son básicos en una estrategia de gestión de identidades. Oracle Internet Directory es un directorio escalable, altamente disponible y seguro. Oracle Internet Directory da servicio como un repositorio central de usuarios para Oracle Identity Management[3], simplificando la administración de los usuarios en el entorno Oracle y proporcionando un directorio de aplicación basado en estándares para las organizaciones.

Gestión de Servicios Web

Oracle Internet Directory es uno de los directorios que presenta una solución completa para añadir mejores prácticas reguladas por políticas a servicios web existentes o de nueva creación, y proporciona las capacidades clave de seguridad y gestión necesarias para desplegar arquitecturas SOA a lo largo de las aplicaciones de negocio. (6)

Directorio X.500 de la UCA (Universidad de Cádiz, España)

El directorio X500 podría decirse que es la guía telefónica de las redes internacionales (Internet) ya que permite buscar direcciones postales, números de teléfono y direcciones de correo electrónico de las personas que forman las organizaciones que están conectadas a Internet.

Permitiéndose tanto búsquedas de personas dentro de una organización como un listado de las personas que forman una determinada organización. La Universidad de Cádiz dispone de un servidor X.500 donde se almacenan los datos de todo el personal de la Universidad, en este las personas se

[3] Oracle Identity Management es una infraestructura de gestión de identidades integrada, escalable y robusta.

encuentran organizadas por departamentos o por servicios. Por tanto es posible realizar la búsqueda de una determinada persona o ver las personas que forman un departamento. (7)

1.2.7 Los Directorios de Búsquedas en Cuba.

Cuba no ha quedado al margen de este universo informático, al quedar oficialmente conectada a Internet, desde Octubre de 1996, surge la necesidad de elaborar diferentes proyectos principalmente con información introducida desde el país , aunque no cuenta con ningún servicio de directorio implementado en la arquitectura SOA ,se puede ver como el desarrollo de estos servicios no solo ha abarcado instituciones de enseñanza sino que se han ido creando en todas las esferas del país, según las necesidades y demandas de cada una.

El Directorio Electrónico de Cuba (D.E.C)

El Directorio Electrónico de Cuba (D.E.C) es un proyecto abarcador que pretende registrar, en una primera etapa, a todos los usuarios de la Red CENIA Internet que lo deseen y de una manera progresiva ir incorporando a usuarios de las demás redes nacionales que así lo soliciten El proyecto en la actualidad se encuentra en la etapa de análisis y evaluación y la información que contendrá se almacena en una base de datos del sistema.

Una vez planteado el proyecto, lo primero que se hizo fue realizar una convocatoria a la libre inscripción de todos los usuarios de la Red CENIA Internet mediante un mensaje electrónico a los mismos en el cual se le explicaba el objetivo del proyecto y un formulario a llenar para la captación de los datos necesarios para conformar la base de datos.

Se comenzó a recibir información de los distintos usuarios y de las diferentes instituciones usuarias de la Red y en la actualidad, se cuenta con aproximadamente 300 registros personales. Se debe destacar que también se recibieron datos de usuarios que pertenecen a otras redes, interesados en formar parte de este incipiente directorio. (8)

Directorio Universitario del ISPJAE (Instituto Politécnico José Antonio Echeverría)

A través de este directorio podrá ser consultada la información de los estudiantes y trabajadores del ISPJAE desde Internet, y permitir de esta manera que aumenten las relaciones de trabajo.

La inscripción al directorio es totalmente libre para los estudiantes y trabajadores los cuales pueden incorporar sus datos y modificarlos cada vez que lo consideren necesario. Para la entrada de los datos se sigue como política la autenticación basada en el usuario y contraseña para el correo electrónico, de esta manera se puede identificar el acceso para modificar o entrar datos en este directorio, quedando la información totalmente validada. (9)

1.2.8 El Directorio de Búsqueda en la UCI.

En la actualidad la Universidad de las Ciencias Informáticas se desarrollan un conjunto de proyectos que son importantes para la vida interna universitaria y atraen a un número cada vez mayor de participantes. Tal es el caso de un motor de búsqueda interno, una guía telefónica, un directorio de personas y una distribución cubana de Linux, por sólo citar algunos.

Con el surgimiento se creó el Directorio de Búsqueda con el objetivo de satisfacer una de las mayores necesidades del personal del centro. Desde finales del curso 2005-2006 el directorio de la UCI pasó a formar parte del grupo de desarrollo del sistema automatizado para la gestión académica (Akademos), por lo que desde hace algunos meses se ha venido trabajando en aras de hacer más rápida y sencilla las búsquedas en el centro.
El actual Directorio de Búsqueda esta desarrollado en la plataforma .NET y a pesar de satisfacer gran parte de las necesidades de búsqueda que exige el usuario que interactúa con el, presenta varios problemas como los que se desglosan a continuación:

1. Lento y poco flexible.
2. Imposibilita la exportación de datos.
3. Difícil mantenimiento.
4. No se ajusta a la arquitectura desarrollada en la universidad, arquitectura SOA.
5. No esta desarrollado sobre software libre.

Entre otras.

La universidad actualmente se encuentra en un comienzo de la adopción de una arquitectura SOA, aún existen pocos servicios web y pocos procesos definidos de los cuales documentados hay muy pocos.
Entre los sistemas de legados que cuentan con algunos servicios se encuentran:

Akademos
Sistema de Acreditación
Pase.

Entre las proyecciones inmediatas que tiene el centro se encuentran documentar totalmente la infraestructura de la universidad e integrar otros sistemas a esta nueva arquitectura, entre los que se encuentra el Directorio de Personas.

El nuevo Directorio pretende dar solución a los problemas existentes en el sistema anterior adaptándose a la nueva arquitectura desarrollada en el centro (SOA), permitiendo el soporte y la integración de información procedente de múltiples aplicaciones, bases de datos y ficheros, así como la migración del mismo a software libre.

1.3 Tendencia.

La tendencia en este trabajo es mejorar la interacción del usuario con el nuevo directorio, de forma que el servicio de búsqueda sea más rápido, flexible y fiable además de brindar nuevas funcionalidades al mismo.

1.4 Metodología y tecnología actuales.

Para el desarrollo del sistema, se escoge como metodología RUP y además se realiza una comparación con las otras metodologías existentes definiendo sus ventajas y desventajas.

1.4.1 Metodologías existentes

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. La metodología indica cómo hay que obtener los distintos productos parciales y finales. (10)

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que se obtiene son clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. Muchas veces no se toma en cuenta el utilizar una metodología adecuada lo que trae

consigo que los proyectos, en problemas son los que salen del presupuesto, tienen importantes retrasos, o simplemente no cumplen con las expectativas del cliente.

Para dar una idea de que metodología se puede utilizar y cual se adapta a este medio se describen tres de ellas de las que se considera las más importantes tal como: XP, OMT y FDD y por último la metodología a seguir: RUP.

Extreme Programming (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo, pequeño equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. (11)

Características de XP, la metodología se basa en:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueden hacer pruebas de las fallas que pudieran ocurrir.
- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo.

¿Qué es lo que propone XP?

- El manejo del cambio se convierte en parte sustantiva del proceso
- El cliente o el usuario se convierte en miembro del equipo

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores
- La simplicidad, al desarrollar y codificar los módulos del sistema
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Se recomienda para proyectos de corto plazo con equipos pequeños y rotables en cuanto a roles. Se Basa en los Use Stories (historias de uso), que al igual que el anterior definen los detalles técnicos sin meterse con los detalles de implementación. En cuanto a carga de trabajo, XP es proceso ligero pero

no se les asignan roles organizativo al equipo, roles como el modelado o generación de la documentación, esto es reemplazado por la presencia de un representante especializado del cliente.

Este proceso está muy orientado a la implementación, esto lo hace bueno para el equipo de desarrollo ya que no debe preocuparse de la documentación y ya que los equipos rotan todos aprenden de todos, por otro lado el cliente también se siente satisfecho pues recibe un software que se adapta exactamente a sus deseas, pero esto implica que para esto debió designar a una persona totalmente involucrada en el negocio, lo que podría implicar que esta persona deje de hacer sus funciones para estar totalmente disponible al equipo de desarrollo, razón por la cual se considera mejor la utilización de este proceso para desarrollos internos, pues debe haber una gran confianza entre el cliente y el equipo de desarrollo, como se hace mención anteriormente era poco probable que el cliente pueda prescindir de sus empleados esto incurriría en un coste adicional para el cliente. Por último como se podría representar todo lo que se debe sin documentación alguna (dependencia entre componentes por ejemplo) si no se anota ni se archiva nada y como alguien más puede tomar el lugar de uno de los miembros del equipo, o hacer mejoras en el sistema, esto crearía una dependencia con el equipo de desarrollo. (12)

OMT (Object Modeling Technique)

Es una de las metodologías de análisis y diseño orientados a objetos, más maduros y eficientes que existen en la actualidad. La gran virtud que aporta esta metodología es su carácter de abierta (no propietaria), que le permite ser de dominio público y, en consecuencia, sobrevivir con enorme vitalidad. Esto facilita su evolución para acoplarse a todas las necesidades actuales y futuras de la ingeniería de software. (13)

Ventajas

1. Proporciona una serie de pasos perfectamente definidos al desarrollador.
2. Tratamiento especial de la herencia.
3. Facilita el mantenimiento dada la gran cantidad de información que se genera en el análisis.
4. Es fuerte en el análisis.

Desventajas

1. Hay pocos métodos para encontrar inconsistencias en los modelos.
2. Interacción de objetos no soportada explícitamente en ninguna herramienta gráfica.
3. Al ser un análisis iterativo es difícil de saber cuando comenzar con el diseño.

4. Es débil en el diseño.

FDD (Feature Driven Development)

Este proceso se considera como punto medio entre los procesos pesados y ágiles, aunque en la práctica es más similar a XP. Pensado para proyectos relativamente cortos, también esta basado en iteraciones que producen un software funcional que puede ser visto, probado y monitorizado por el cliente.

Estas iteraciones son decididas en base a las funcionalidades que el software debe tener, funcionalidades definidas por el cliente, este proceso está dividido en cinco fases:

1. Desarrollo de un Modelo General.
2. Construcción de la Lista de Funcionalidades.
3. Diseñar en base a las funcionalidades definidas.
4. Implementar en base a las mismas funcionalidades.

Aquí en el equipo de trabajo si existen jerarquías, siempre debe haber un jefe de proyecto, y aunque es un proceso considerado ligero también incluye documentación (la mínima necesaria para que algún nuevo integrante pueda entender el desarrollo de inmediato).

Desventaja

Aunque también genera documentación es la mínima necesaria para comprender el código, si bien es cierto el quipo cuenta con cierta libertad también es cierto que estos dependen directamente del jefe de proyecto quien es el responsable directo de proyecto.

El principal problema de este proceso esta representado por la necesidad de contar con miembros experimentados en el equipo, miembros que puedan definir los roles y funciones de cada uno, que marquen el camino a seguir desde el principio con la elaboración del modelo global, el hecho de ser una combinación de XP y RUP es lo que lo hace más atractivo a la implementación, claro siempre y cuando la experiencia de esa gente de mayor jerarquía no cree dependencias.

RUP (Rational Unified Process)

RUP es un proceso de ingeniería de software que constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Provee lineamientos, templates para herramientas que guían una implementación efectiva para las mejores prácticas de un software.

Es un proceso basado en componentes, que utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software.

Entre sus principales características se encuentran :(14)

1. Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
2. Pretende implementar las mejores prácticas en Ingeniería de Software
3. Desarrollo interactivo
4. Administración de requisitos
5. Uso de arquitectura basada en componentes
6. Control de cambios
7. Modelado visual del software
8. Verificación de la calidad del software

Es uno de los procesos más generales que existe, se basa en la documentación generada en cada uno de sus cuatro fases:

1. Intercepción (puesta en marchar).
2. Elaboración (definición, análisis y diseño).
3. Construcción (implementación).
4. Transición (fin del proyecto y puesta en producción) en las cuales se ejecutarán varias iteraciones (según el tamaño del proyecto).

RUP se basa en Use Case (casos de uso) para describir lo que se tiene y lo que se espera del software, está muy orientado a la arquitectura del sistema a implementarse, documentándose de la mejor manera, gracias a su plan de desarrollo con el que se controla el desarrollo se pueden reconocer los problemas y fallos de forma temprana y corregirlos.

RUP define las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. El proceso de desarrollo se divide en cuatro fases.

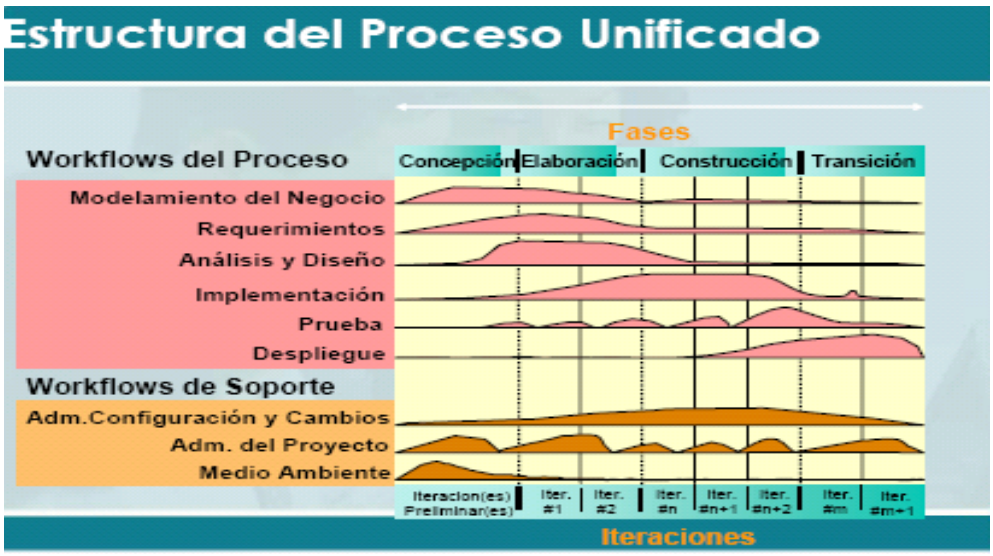

Figura 1. 2 RUP en dos dimensiones.

Flujos de trabajo:

- **Modelamiento del negocio:** Describe los procesos de negocio, mediante la identificación de participantes y actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos).
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.
- **Instalación:** Realiza actividades (empaque, instalación, asistencia a usuarios, etc.).
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios*:*** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

El ciclo de vida de RUP se caracteriza por:

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo.
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo, por lo que describe los elementos del modelo más importantes para su construcción. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.
- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, unos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

Tecnologías

1.4.2 AJAX.

AJAX, acrónimo de Java Script asíncrono y XML, es una técnica de desarrollo Web para crear aplicaciones interactivas.

AJAX es realmente muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas nuevas formas.

AJAX incorpora: (15)

Presentación basada en estándares usando XHTML y CSS.

Exhibición e interacción dinámicas.

Intercambio y manipulación de datos usando XML.

Recuperación de datos asincrónica usando XMLHttpRequest y Java Script poniendo todo junto [15].

1.4.3 SGBD: PostgreSQL.

PostgreSQL es un servidor de base de datos libre. Tiene soporte total para transacciones, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta

de software libre PostgreSQL tiene entre otras ventajas, su código fuente está disponible sin costo alguno y es multiplataforma.

Fue diseñado para ambientes de alto volumen. Escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Tiene mejor soporte para vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos.

Características

Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

Altamente Extensible

Soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios.

MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles. (16)

Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

Ventajas

1. Instalación ilimitada.
2. Mejor soporte que los proveedores comerciales.
3. Ahorros considerables en costos de operación.
4. Estabilidad y confiabilidad legendarias.
5. Extensible.
6. Multiplataforma.
7. Diseñado para ambientes de alto volumen.
8. Herramientas gráficas de diseño y administración de bases de datos.

1.5 Patrones de arquitectura.

1.5.1 Arquitectura Orientada a Servicios (SOA).

Una arquitectura SOA puede ser implementada usando cualquier tipo de infraestructura cliente/servidor, pero lo más común es que se implemente con Servicios Web.

Muchas veces se confunde una Arquitectura Orientada a Servicios con Servicios Web. SOA es una visión de arquitectura fruto de la evolución tecnológica y de las necesidades de negocio. Mientras que los Servicios Web son elementos, entre otros, que permiten su implementación.
Puesto que cada servicio web puede estar implementado en una tecnología heterogénea es necesario cumplir una serie de Estándares para hacer posible la comunicación entre ellos. Los más comúnmente utilizados son los siguientes:

1. **XML**: Es un lenguaje de marcado capaz de describir distintos tipos de datos. Es un estándar ampliamente aceptado y utilizado medio de descripción de datos.
2. **SOAP:** Es un protocolo de comunicación entre procesos basado en el intercambio de mensajes en formato XML dentro de una red. A su vez SOAP esta basado en XML y es completamente independiente de la plataforma y del lenguaje en el que estén implementados los procesos que se comunican.
3. **WSDL (***Web Services Description Language***)**: Es un lenguaje basado en XML que permite describir servicios web (como su nombre indica). Un documento WSDL especifica, entre otras cosas, dónde se encuentra el servicio así como las operaciones que pone accesibles a otros servicios.

4. **UDDI:** Es un directorio, basado en XML, en el que las distintas empresas dan de alta servicios web que ponen al servicio de otra empresas.

Ventajas

1. Reducción del tiempo de desarrollo: Los servicios SOA son fácilmente reusables y pueden ser ensamblados rápidamente. Gracias a la reusabilidad los tiempos de desarrollo de las aplicaciones se reducen drásticamente. Puesto que generalmente en las empresas la mayor parte de procesos de negocio ya estaban implementados, es mucho más rápido reutilizarlos que volver a crearlos.
2. Reducción de los costes de desarrollo y mantenimiento: La reducción del tiempo de desarrollo lleva consigo la reducción del coste de desarrollo. Además los Servicios reducen la complejidad interna de las aplicaciones ya que la funcionalidad está separada en bloques independientes más pequeños, el tiempo de mantenimiento se reduce así como los costes.
3. Más flexible y escalable: El crecimiento de la empresa ya no afecta a toda la aplicación, basta con añadir o modificar los módulos de funcionalidad (servicios) necesarios.
4. Más ágil: Debido al débil acoplamiento entre las interface de los distintos servicios se consiguen aplicaciones más ágiles debido al entorno en continuo cambio en el que están inmersas las empresas.
5. Mayor calidad en los servicios y reducción de riesgos: La reutilización de servicios es segura y fiable puesto que ya han sido probados a través de numerosos clientes que acceden a ellos y en muchos casos mejorados. Con lo que se aumenta la calidad del servicio y se reducen los riesgos. (17)

1.5.2 Modelo Cliente Servidor.

La arquitectura Cliente/Servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que, definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información. Se establece así un enlace de comunicación transparente entre los elementos que conforman la estructura. Entre las principales características de la arquitectura Cliente/Servidor, se pueden destacar las siguientes: (18)

1. El servidor presenta a todos sus clientes una interfaz única y bien definida.
2. El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

3. El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
4. Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Ventajas de la arquitectura cliente-servidor:

1. El servidor no necesita potencia de procesamiento, parte del proceso se reparte con los clientes.
2. Se reduce el tráfico de red considerablemente. Idealmente, el cliente se conecta al servidor cuando es estrictamente necesario, obtiene los datos que necesita y cierra la conexión dejando la red libre.

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor.

1.5.3 Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

Symfony es el framework que se utiliza en la aplicación, está basado en un patrón clásico conocido como arquitectura MVC, que está formado por tres niveles:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- 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.

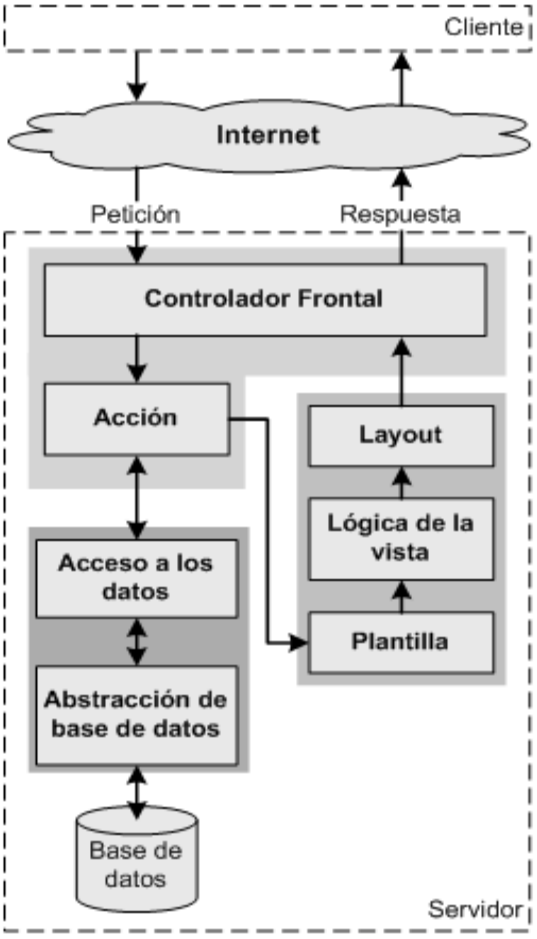

Figura 1.3 Patrón MVC en Symfony.

En Symfony, la capa del controlador, que contiene el código que liga la lógica de negocio con la presentación, está dividida en varios componentes que se utilizan para diversos propósitos [21]:

- El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse.
- Las acciones contienen la lógica de la aplicación. Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación.
- Los objetos request, response y session dan acceso a los parámetros de la petición, las cabeceras de las respuestas y a los datos persistentes del usuario. Se utilizan muy a menudo en la capa del controlador.
- Los filtros son trozos de código ejecutados para cada petición, antes o después de una acción. Por ejemplo, los filtros de seguridad y validación son comúnmente utilizados en aplicaciones web.

La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

- Los diseñadores web normalmente trabajan con las plantillas (que son la presentación de los datos de la acción que se está ejecutando) y con el layout (que contiene el código HTML común a todas las páginas). Estas partes están formadas por código HTML que contiene pequeños trozos de código PHP, que normalmente son llamadas a los diversos *helpers* disponibles.
- Para mejorar la reutilización de código, los programadores suelen extraer trozos de las plantillas y los transforman en componentes y elementos parciales. De esta forma, el layout se modifica para definir zonas en las que se insertan componentes externos. Los diseñadores web también pueden trabajar fácilmente con estos trozos de plantillas.
- Hasta ahora, la mayor parte de los contenidos se ha dedicado a la construcción de páginas y al procesado de peticiones y respuestas. Sin embargo, la lógica de negocio de las aplicaciones web depende casi siempre en su modelo de datos.

El componente que se encarga por defecto de gestionar el modelo en Symfony es una capa de tipo ORM[4] realizada mediante el proyecto Propel[5] .En las aplicaciones Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; de esta forma nunca se accede de forma explícita a la base de datos. Este comportamiento permite un alto nivel de abstracción y permite una fácil portabilidad. (19)

- En este capítulo se explica como crear el modelo de objetos de datos, y la forma en la que se acceden y modifican los datos mediante Propel. Además, se muestra la integración de Propel en Symfony.

1.6 Patrones de diseño.

1.6.1 Patrones GRASP

[4] Mapeo objeto-relacional.
[5] Es la librería externa que utiliza Symfony para realizar su ORM.

GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades) El nombre se eligió para indicar la importancia de ***captar*** estos principios, si se quiere diseñar eficazmente el software orientado a objetos. (20)

Bajo Acoplamiento

El Bajo Acoplamiento es un principio que se debe recordar durante las decisiones de diseño, es la meta principal que es preciso tener presente siempre. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. El Bajo Acoplamiento estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

El Bajo Acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto o Alta Cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

Beneficios:

1. No se afectan por cambios de otros componentes
2. Fáciles de entender por separado
3. Fáciles de reutilizar

Alta Cohesión

Alta Cohesión es un principio que se debe tener presente en todas las decisiones de diseño, es la meta principal que ha de buscarse en todo momento. Grady Booch[6] señala que se da una alta cohesión funcional cuando los elementos de un componente "colaboran para producir algún comportamiento bien definido".

El patrón Alta Cohesión presenta semejanzas con el mundo real, ya que si alguien asume demasiadas responsabilidades -sobre todo las que debería delegar-, no será eficiente.

Beneficios:

[6] Co-inventor de UML.

1. Mejoran la claridad y la facilidad con que se entiende el diseño.
2. Se simplifican el mantenimiento y las mejoras en funcionalidad.
3. A menudo se genera un bajo acoplamiento.
4. La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Experto

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. Nótese que el cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos [20].

Ello significa que hay muchos expertos "parciales" que colaboraron en la tarea. El patrón Experto ofrece una analogía con el mundo real.

Beneficios:

1. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
2. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases "sencillas" y más cohesivas que son más fáciles de comprender y de mantener.

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento [20]

Beneficios:

1. Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

2. Es probable que el acoplamiento no aumente, pues la clase creada tiende a ser visible a la clase creador, debido a las asociaciones actuales que llevaron a elegirla como el parámetro adecuado.

Controlador

Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad.

Beneficios:

1. Mayor potencial de los componentes reutilizables. Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Desde el punto de vista técnico, las responsabilidades del controlador podrían cumplirse en un objeto de interfaz, pero esto supone que el código del programa y la lógica relacionada con la realización de los procesos del dominio puro quedarían incrustados en los objetos interfaz o ventana.
2. Reflexionar sobre el estado del caso de uso. A veces es necesario asegurarse de que las operaciones del sistema sigan una secuencia legal o poder razonar sobre el estado actual de la actividad y las operaciones en el caso de uso subyacente.

1.6.2 Patrones GOF (Gang of Four)

Fábrica Abstracta:

Es un patrón creacional en la clasificación de los patrones GOF. Proporciona una interfaz para crear familias de objetos sin especificar su clase de forma concreta.

Consecuencias:

1. Se potencia el encapsulamiento, puesto que se aísla a los clientes de las implementaciones.
2. Se incrementa la flexibilidad del diseño, resultando fácil cambiar de familia de productos.
3. Se refuerza la consistencia (alta cohesión y bajo acoplamiento), puesto que se restringe el uso a productos de una sola familia cada vez.

Fachada:

Simplifica el acceso a un conjunto de clases o interfaces. Proporcionar una interfaz unificada de un subsistema sin ocultar sus interfaces. Permite acceder a elementos del sistema y realizar operaciones más complejas con ellos de forma transparente.

Consecuencias:

1. Oculta a los clientes parte de la complejidad de los subsistemas.
2. Disminuye el acoplamiento entre subsistemas y cliente.
3. Disminuye el acoplamiento (las interfaces de por sí reducen el acoplamiento).
4. Ocultación de componentes del subsistema.
5. Que el cliente pueda utilizar toda la funcionalidad del sistema.

Instancia única o Solitario (Singleton):

El patrón Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ésta instancia.

Consecuencias:

1. Acceso controlado a la única instancia. Puede tener un control estricto sobre cómo y cuándo acceden los clientes a la instancia.
2. Espacio de nombres reducido. El patrón Singleton es una mejora sobre las variables globales. Permite el refinamiento de operaciones y la representación. Se puede crear una subclase de Singleton.
3. Permite un número variable de instancias. El patrón hace que sea fácil cambiar de opinión y permitir más de una instancia de la clase Singleton. Más flexible que las operaciones de clase.

.

1.7 Servicios Web

Un Servicio Web es una colección de protocolos y estándares que sirve para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

El Servicio Web es un componente de software que se basa en las siguientes tecnologías:

1. Un formato que describa la interfaz del componente (sus métodos y atributos) basado en XML. Por lo general este formato es el WSDL, Lenguaje de descripción de servicios Web.
2. Un protocolo de aplicación basado en mensajes y que permite que una aplicación interaccione (use, instancia, llame, ejecute) al servicio. Por lo general este protocolo es SOAP.

Un servicio Web puede ser usado internamente por una aplicación o ser publicado en Internet. Estos servicios permiten la ejecución de sus funcionalidades sin importar la plataforma, sistema operativo, o lenguaje en el cual estén implementados. Gracias a los servicios Web se pueden hacer que sistemas heterogéneos que trabajen conjuntamente como una sola aplicación computacional. Estos servicios constituyen una potente herramienta para el desarrollo de aplicaciones distribuidas. (21)
La aplicación Directorio de Búsqueda utiliza los servicios implementados en la universidad publicados en el Directorio de Servicios Web UCI.

- Registro de Identidad UCI.
- ASSETS - Sistema de Capital Humano.

1.8 Lenguajes

1.8.1 Hypertext Markup Language (HTML).

HTML es un lenguaje de hipertexto cuyas siglas significan Lenguaje de Marcas Hipertexto. HTML consiste en códigos estándar o "marcas" que son usadas para definir la estructura de la información en una página Web. HTML es usado para crear documentos en la World Wide Web. Una página Web es la unidad básica de información accesible desde la World Wide Web[7].

HTML define varios aspectos de una página web, incluyendo títulos, negritas, itálicas, imágenes, párrafos y conexiones de hipertexto a otros recursos, es por ello que puede ser comparado con el procesamiento de palabras. (22)

1.8.2 Cascading Style Sheets (CSS).

[7] Red Global Mundial

CSS, Cascading Style Sheets, hojas de estilo en cascada, fue introducido en 1996 como el estándar para añadir información de estilo a los documentos HTML. El lenguaje y reglas de CSS hacen este trabajo de forma muy simple y clara, facilitando la tarea de diseñar páginas Web atractivas y vistosas.
Hojas de Estilo en Cascada es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. (23)

1.8.3 PHP (Hypertext Preprocessor).

Es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.
PHP es un lenguaje de programación de estilo clásico, con variables, sentencias condicionales, bucles, funciones. (24)

Diseñado para, entre otras cosas, aumentar, incrementar el dinamismo de las páginas Web. Originalmente se trataba de un conjunto de macros concebidas para ayudar en el mantenimiento de páginas Web. Desde entonces, sus características han ido creciendo hasta convertirse en un lenguaje de programación completo, capaz de manejar entornos que integran grandes bases de datos. Su popularidad se basa, en gran parte, a su sintaxis similar a la del lenguaje de programación C, su rapidez y simplicidad.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como Linux y Windows, y puede interactuar con los servidores de Web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI[8]. (25)
Es multiplataforma, funciona tanto para Unix (con Apache) como para Windows (con Microsoft Internet Information Server) de forma que el código que se haya creado para una de ellas no tiene porqué modificarse al pasar a la otra. (26)

Porqué utilizar PHP y no otra herramienta: (27)

1. PHP no soporta directamente punteros, como el C, de forma que no existen los problemas de depuración provocados por estos.

[8] Internet Server Application Programming Interface

2. Se pueden hacer grandes cosas con pocas líneas de código. Lo que hace que merezca la pena aprenderlo.
3. El código PHP es mucho más legible que el de PERL, todo el que haya programado PERL podrá corroborar esta afirmación.
4. Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, XML, creación de PDF ...)
5. Al poderse encapsular dentro de código HTML se puede recoger el trabajo del diseñador gráfico e incrustar el código PHP posteriormente.
6. Esta siendo utilizado con éxito en varios millones de sitios web.
7. Hay multitud de aplicaciones PHP para resolver problemas concretos (tiendas virtuales, periódico) listas para usar.
8. Es multiplataforma, funciona en todas las plataformas que soporten apache.
9. Es software libre. Se puede obtener en la web y su código esta disponible bajo la licencia GPL.

1.8.4 JavaScript

Se trata de un lenguaje de tipo script compacto y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.
Los programas Java Script van incrustados en los documentos HMTL, y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos. (28)

Es el lenguaje que nos permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida.

1.8.5 Lenguaje Unificado de Modelado (UML).

El Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Los diagramas son entes importantes de UML, cuya finalidad es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementarlo. El modelo gráfico de UML tiene un vocabulario en el que se identifican: elementos, relaciones y diagramas.

1.9 ¿Por qué utilizar PHP y PostgreSQL?

El departamento de Informatización sigue una política de utilización de software libre, es por esto que se plantea como propuesta de solución en este sentido cumpliendo con estos principios. Además como se expuso anteriormente PHP y PostgreSQL dadas sus características constituyen una buena alternativa en este sentido.

1.10 Herramientas

1.10.1 Macromedia Dreamweaver, herramienta de diseño web.

Macromedia Dreamweaver es uno de de los programas más utilizados en todo el mundo para la creación de páginas web. Es empleado tanto por profesionales como por personas que se inician en la creación de su primera página web. (29)

Es una de las herramientas más completas y estandarizadas a nivel mundial, con varios años de experiencia. Incluye una gran cantidad de funcionalidades que le permiten al diseñador explotar al máximo sus potencialidades, se integra con múltiples lenguajes y plataformas incluyendo PHP, incluye herramientas para trabajar aplicaciones que manejan XML, así como mejoras a su manejo de hojas de estilo (CSS) y guía para revisar los diseños y una barra de código para accesar funciones frecuentes. Además de sus ventajas existía dominio de la herramienta por parte del equipo de trabajo por lo que se decide su utilización.

1.10.2 Eclipse

Eclipse es un IDE (Entorno de Desarrollo Integrado) tan potente como popular que incorpora un sin fin de utilidades para simplificar la labor de los programadores. Aparte de ser un entorno de desarrollo completo, una de las particularidades más interesantes para la comunidad es que es de código libre y gratuito. (30)

Para Eclipse existen diversos plug-ins o añadidos para proveer de nuevas utilidades al programa, enfocadas a diversos usos que los distintos tipos de programadores pueden necesitar. Unos de los añadidos de Eclipse que más pueden interesar a los desarrolladores de páginas web sería el módulo para programación en PHP.

Entre estos esta PDT (PHP Development Tools, Eclipse) trabaja para proveer un IDE completamente funcional para PHP para la plataforma Eclipse.
PDT es independiente de plataforma, estando disponible en Windows o Linux.
Entre las características en la versión actual (1.0) se encuentran:

1. Editor sensible al contexto, el cual provee de resaltamiento de código, asistente de código y autocompletado de código.
2. Ayuda para eliminar errores de código de PHP adicionales.
3. Integración con el modelo del proyecto Eclipse, que permite para inspeccionar el uso de las vistas del contorno del fichero y del proyecto.
4. Soporte para el debug incremental del código de PHP
5. Extensos frameworks que permiten a los desarrolladores fácilmente extender PDT para crear nuevas e interesantes herramientas orientadas al desarrollo de PHP.

1.10.3 Visual Paradigm

Visual Paradigm para UML es una herramienta CASE profesional, fácil de utilizar y que soporta el ciclo de vida completo del desarrollo de software; usa al UML como lenguaje de modelado, característica que ayuda a la construcción de aplicaciones con calidad, de manera intuitiva, a menor coste; además de que facilita la comunicación entre los miembros del equipo de desarrollo, al garantizar el uso de un lenguaje estándar común. (31)

Se decidió utilizar esta herramienta CASE para el modelado del diseño de la aplicación, ya que entre sus principales características y facilidades se encuentran: (32)

1. Brinda la posibilidad de crear un conjunto bastante amplio de artefactos utilizados con mucha frecuencia durante la confección de algún diagrama en el desarrollo del software. Todos estos, cumpliendo con el Estándar UML 2.0.

2. Disponibilidad en múltiples plataformas (multiplataforma): Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
3. Puede ser utilizado en varios idiomas, sus componentes se encuentran relacionados, por lo que se hace muy fácil la creación de cualquier tipo de diagrama, no se requieren grandes conocimientos para su manejo.
4. Brinda un número considerable de estereotipos, lo que permite un mayor entendimiento de los diagramas.
5. Integración con distintos Ambientes de Desarrollo Integrados (IDE): se integra fácilmente con varios IDEs de desarrollo, entre los que se encuentra Eclipse, que es el que se utilizará para el desarrollo de la aplicación. Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.
6. Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

1.11 Framework

1.11.1 Symfony

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas Linux como en plataformas Windows [19].

1. Fácil de instalar y configurar en la mayoría de plataformas
2. Independiente del sistema gestor de bases de datos
3. Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como por ejemplo [19]:

1. La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
2. La capa de presentación utiliza plantillas y *layout* que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework. Los *helpers* incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.
3. Los formularios incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
4. Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.
5. Las interacciones con Ajax son muy fáciles de implementar mediante los *helpers* que permiten encapsular los efectos Java Script compatibles con todos los navegadores en una única línea de código.

Es un framework maduro, bien documentado, recomendable para aplicaciones web complejas con mucha lógica de negocio pues permite el mantenimiento y las ampliaciones futuras de la aplicación, con un código ligero, legible y efectivo.

1.12 Conclusiones

En este capítulo se exponen las condiciones y problemas que rodean el objeto de estudio a través de los conceptos y definiciones planteadas. Se evidencia la necesidad de implementar un software que permita la interoperabilidad, organización y consumo de los servicios Web en el contexto de la arquitectura SOA de la UCI. Se realizó la justificación de cada herramienta, metodología a utilizar y los lenguajes que se emplearon para el desarrollo del sistema.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se realiza la descripción de la solución propuesta para la implementación del sistema, se describen los procesos y se decide la utilización de un Modelo de Dominio, donde se recrea el entorno del problema. Se explica el comportamiento del sistema a través de los requisitos funcionales y no funcionales, casos de uso y su relación con los actores del sistema, a través del diagrama de casos de usos.

2.1.1 Problema y Situación Problémica.

Actualmente, en la universidad se utiliza el Directorio de Persona desarrollado en la plataforma .NET. Este sistema, aunque automatiza algunas actividades relacionadas con la búsqueda de personas del centro, no satisface las demandas de sus usuarios al exigir rapidez, facilidad a la hora de realizar una operación determinada y en muchos casos satisfactorios resultados.

Gran parte de estas molestias son ocasionadas debido a que el sistema interactúa directamente con la bases de datos lo que imposibilita su óptimo funcionamiento.

2.1.2 Objeto de automatización.

Será objeto de automatización el proceso de búsqueda de personas por parte del personal de la universidad así como los procesos relacionados con la misma como salvar y cargar la misma agilizando el mismo. De las actividades en las que está inmerso el sistema propuesto, se proponen automatizar las siguientes:

1. Autenticar al usuario que acceda al sistema

Este proceso se llevara a cabo una vez que el usuario desee utilizar la aplicación otorgándole los privilegios necesarios.

2. Realizar búsquedas personalizadas.

Al ingresar al sistema el usuario tendrá la opción de realizar una búsqueda simple o avanzada.

3. Salvar búsquedas

Una vez realizadas las búsquedas puede salvar las mismas bajo un nombre determinado, el usuario podrá acceder y realizar posteriores cambios a las mismas.

4. Cargar búsquedas

El usuario una vez entrado al sistema puede cargar las búsquedas que ha guardado anteriormente y efectuar cambios a la misma.

5. Exportar resultados.

Luego de obtener el resultado deseado el usuario puede exportar el mismo hacia un formato cvs.

Estos procesos se basan en la búsqueda y presentación de información, por lo que para su mayor comprensión se utiliza uno de los modelos que brinda RUP, el Modelo de Dominio, mas adelante se hará una descripción mas detallada de los mismos.

2.2 Entorno del sistema. Propuesta del sistema.

Para resolver esta situación se propone implementar un sistema automatizado, seguro, flexible y ágil desarrollado en un ambiente SOA que mejore la calidad y el tratamiento de la información del servicio de Directorio de Persona.

El Directorio de Personas que se propone implantar se desarrolla sobre una Arquitectura Orientada a Servicios consumiendo servicios web que le dan agilidad y escalabilidad al sistema final.

Entre las funcionalidades que tiene el mismo se encuentra la Autenticación del usuario o cliente, quien tiene acceso al resto de las funcionalidades del sistema una vez autenticado. De ahí va directo a la sección de Búsqueda Simple donde puede realizar la misma a través de los criterios usuario, nombre o solapin .Si el usuario desea realizar la búsqueda en la sección Búsqueda Avanzada accede a ella a través de un link, la misma se realiza escogiendo cualquier combinación de los campos usuario, nombre, carnet de identidad, solapin, municipio, provincia especificando si es trabajador o estudiante. En ambas secciones una vez realizada la búsqueda puede Exportar este resultado a un formato cvs. Permite Salvar las búsqueda, esta se realiza una vez que se escojan las combinaciones de campos en la sección Búsqueda Avanzada, esta combinación se guarda en la base de datos del sistema con un nombre entrado por el usuario y así una vez que el usuario quiera realizar esta misma búsqueda y modificar solo tiene que ir a la opción Cargar, y el sistema carga nuevamente estos campos.

2.3 Modelo de Dominio.

Al desarrollar un sistema es necesario comprender la organización bajo estudio y los procesos que en ella tienen lugar, a fin de lograr una mejor comprensión del problema a resolver y el común entendimiento entre clientes y desarrolladores. En la presenta propuesta del sistema, al tener un negocio tan pequeño, no se logra determinar este proceso con fronteras bien establecidas donde se logren ver claramente, quienes son las personas que lo inician, ni quienes son los beneficiados con cada uno de los procesos definidos, pero además no se logra precisar quienes son las personas que desarrollan las actividades en cada uno de estos.

Como parte de este análisis se llega a la conclusión de realizar un modelo de Dominio para la aplicación del directorio del centro en lugar de realizar un modelo de negocio. Se considera un escenario apropiado para la alternativa de Modelo de Dominio, donde el objetivo primario es la búsqueda y presentación de información.

Durante esta fase del modelo del dominio se centra en conocer el dominio para el que se va a desarrollar un sistema. Se trata de una fase de análisis en la que aún se piensa en los objetos y las relaciones del mundo real en vez de en los conceptos de programación.Este modelo permite la representación visual de los conceptos u objetos significativos del entorno o área del problema capturando los tipos más importantes de objetos que se presentan en el contexto descrito que permiten mostrar los principales conceptos que se manejan en el dominio del sistema en desarrollo. Para ello se presenta una tabla con los conceptos y descripciones que ayudarán a una mejor comprensión del problema.

Concepto	**Descripción**
Universidad	Institución donde se desarrolla este trabajo. La cual tiene una aplicación para realizar la búsqueda de cualquier personal del centro.
Áreas	Diferentes esferas donde trabajan o laboran el personal del centro, las mismas pueden ser facultades, áreas de trabajo, departamentos.
Aplicación Directorio de Búsquedas	Es la aplicación que permite que los estudiantes, y demás trabajadores del centro realizar las diferentes búsquedas de personas que están en la universidad.
Personal	Todas las personas que pertenecen a la Universidad de Ciencias Informáticas.
Estudiantes	Las personas que reciben clases en el centro.
Trabajador	Las personas que laboran en el centro.

Profesor	Aquellos que imparten clases y tiene que ver con la docencia en la universidad.
Dirigente	Es un trabajador de la universidad que ocupa cargos en la misma.
Tercerizado	Personas que trabajan dentro de la uci, pero no pertenecen a la ella y no se ajustan a su escala salarial.
Familiar	Personas cuyos familiares son trabajadores del centro y conviven en el mismo.
Huésped	Persona externa a la UCI pero que esta hospedada (puede ser en el hotel universitario), una visita que viene invitado por cuestiones de trabajo
Eventual	Personas que trabajan para la UCI pero temporalmente.
Búsqueda	Es la principal operación que realiza la institución, a través de diferentes campos nombre, usuario, solapin, carnet de identidad, provincia, municipio y tipo de persona.
Información	Son los campos a través de los cuales se realizan las diferentes búsquedas.

Tabla 2.1 Descripción de los Concepto del dominio del problema.

2.3.1 Diagrama de clases del Modelo de Dominio.

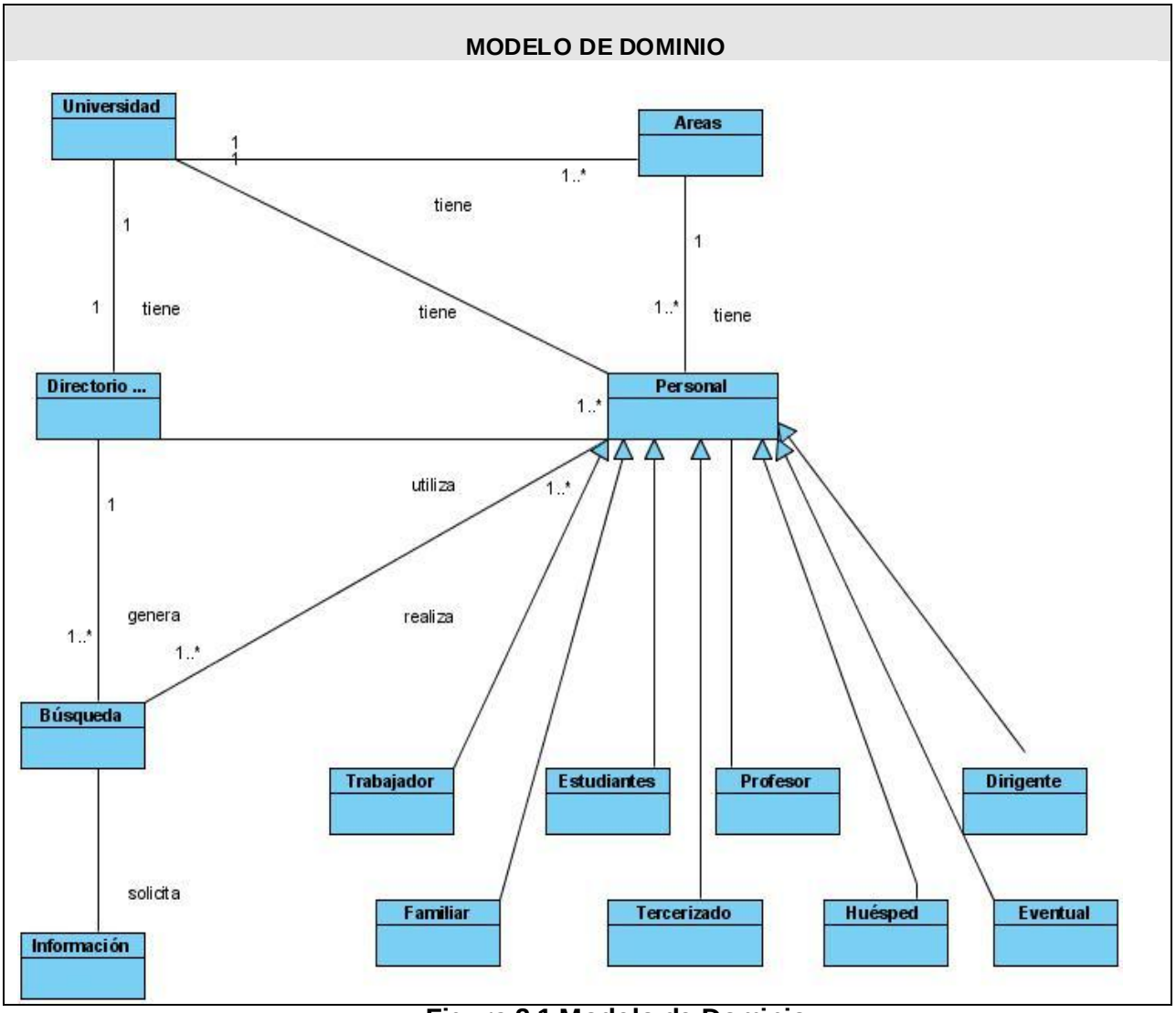

Figura 2.1 Modelo de Dominio.

2.3.2 Requerimientos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Ellos no alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen.

De acuerdo a las necesidades que presenta el sistema actual y las funcionalidades que se desean implementar contribuyendo a la mejora del mismo y teniendo en cuenta a las diferentes especificaciones del cliente, se han definido los siguientes requisitos funcionales.

R1: Autenticar usuario.

R2: Brindar el servicio de búsqueda a cualquier personal del centro.

R2.1.: Mostrar sección Búsqueda Avanzada.

R2.1.2.: Permitir filtrar por todos los campos que tenga cada persona.

R2.2.: Mostrar sección Búsqueda Simple.

R3: Permitir exportar el resultado de una búsqueda a un formato cvs, etc.

R4: Salvar búsquedas personalizadas.

R5: Cargar Búsquedas.

2.3.3 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Rendimiento

El sistema debe ser lo más eficiente y rápido posible para poder lograr en menor tiempo una respuesta adecuada. La disponibilidad de trabajo en red contra el servidor debe ser constante. Se debe garantizar que la respuesta a solicitudes de los usuarios del sistema sea en un período de tiempo breve. El sistema deberá de ser lo más estable y confiable posible.

Soporte

El producto debe recibir mantenimiento ante cualquier fallo que ocurra.

Se debe lograr una solidez de los datos realizando mantenimientos automatizados en la base de datos, orientados a la actualización y salvas de seguridad, a horas del día donde haya la menor cantidad de usuarios conectados.

Software

Para el funcionamiento del sistema en el servidor es necesario el S.O. Linux o Windows. Para el funcionamiento del sistema en las terminales cliente es necesario el S.O. Windows 95 o superior, Linux o Unix aunque puede usarse cualquier SO que soporte un navegador y que por supuesto esté conectada a la red.

Navegadores (Microsoft Internet Explorer, Netscape, Mozila Firefox)
Servidor Web (Apache para ambas).

Portabilidad

El sistema debe ser compatible con todos los sistemas operativos sin necesidad de cambios significativos.

Seguridad

El sistema debe encargarse de controlar los diferentes niveles de acceso y funcionalidad de usuarios al sitio de identificar al usuario antes de que pueda realizar cualquier acción sobre el sistema. Debe garantizar que la información sea vista únicamente por quien tiene derecho a verla.
Debe mantenerse la corrección e integridad de los datos.

- **Confidencialidad**: La información manejada por el sistema está protegida de acceso no autorizado y divulgación. Debe mantenerse la consistencia de los datos en correspondencia con la realidad.

Las herramientas a usar deben ser confiables con soporte para la recuperación ante fallos y errores.

- **Integridad**: La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos.
- **Disponibilidad:** Significa que los usuarios autorizados se les debe garantizar el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no deben ocultar o retrasar a los usuarios para obtener los datos deseados en un momento dado.

La seguridad de un sistema no solo tiene en cuenta la seguridad del sistema propiamente dicho sino, además, el ambiente en el que se usará el sistema. Por lo que se tiene que contemplar la seguridad física del lugar donde se usa la aplicación.

Usabilidad

El sistema ha sido diseñado de tal forma que permita a los usuarios tener control en todo momento del sitio web, evitando para eso la sobrecarga de información y para cada proceso una muestra a través

de mensaje de las acciones hechas por los usuarios. Los colores usados permiten el acceso a usuarios con problemas de distinción de color, se reduce el tiempo de latencia, además, requiriendo un mínimo proceso de aprendizaje.

Hardware.
Para el cliente:

- Se puede usar cualquier PC que soporte un navegador y este conectada a la red.

Para el servidor:

- Ubuntu Gutsy server, 2 Gb de memoria RAM.
- Con 160 Gb de disco duro.
- Tarjeta de red.

Debe estar instalada en la computadora donde se ejecuta el terminal de comunicación al menos la versión cliente de la aplicación PostgreSQL para que esta pueda acceder a la base de datos donde se almacenará la información que se obtendrá a partir de los datos enviados la tarjeta interfaz.

Interfaz externa
Interfaz con un diseño sencillo pero amigable, con vista a acelerar la velocidad de respuesta hacia el usuario debido a la complejidad de los procesos llevados a cabo en el terminal. La interfaz debe limitarse a presentar las funcionalidades netamente de la estación logrando la concentración del usuario en las tareas que esté realizando. Su diseño gráfico debe ser acorde con las pautas de diseño de la Intranet universitaria, adaptada para una resolución de 800x600.

Confiabilidad.
Es decisiva la precisa ejecución de la aplicación para la actualización de la base de datos. Esta característica es fundamental debido que la precisión de los datos que se reciben y en función de su valor garantizan que el articulo este de alta o baja.

En caso de alguna falla que pudiera provocar la detención de la aplicación se debe proceder a su inmediata reparación y realización nuevamente de todos los chequeos que se realizaron en ese tiempo.
Todas las salidas del sistema tienen que tener el 100% de veracidad y precisión.

Restricciones en el diseño y la implementación.

Implementación del sistema en el lenguaje PHP.

PostgreSQL como sistema de gestor de base de datos.

Eclipse como herramienta de desarrollo.

Legales

El producto está regido por las normativas que se establecen en los diferentes reglamentos de la Universidad de las Ciencias Informáticas (UCI).

2.4 Modelo de Casos de uso del Sistema.

2.4.1 Descripción de los actores del sistema.

Definición de los actores del sistema

Los actores del sistema no son parte del mismo, pero sí pueden intercambiar información con él. Un actor puede:

- Sólo brindar información de entrada al sistema.
- Sólo recibir información del sistema.
- Brindar y recibir información.

Actor	Descripción
Cliente	Es el usuario a quien se le brindan los diferentes servicios para realizar la búsqueda de personas.

Tabla 2.2 Actor del sistema

2.4.2 Diagrama de paquetes.

DIAGRAMA DE PAQUETES

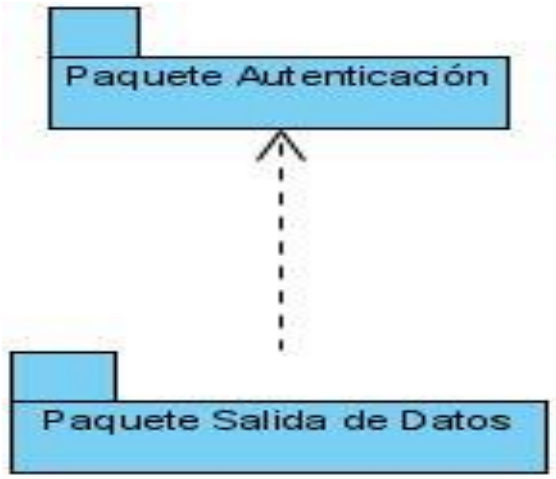

Figura 2.2 Diagrama de Paquetes.

2.4.3 Listado de casos de uso.

CU-1	Autenticar
Actor	Cliente
Descripción	El cliente debe autenticarse antes de realizar cualquier operación en el sistema de forma que quede previamente autorizado a usar el mismo.
Referencia	R1

Tabla 2.3 Caso de uso Autenticar

CU-2	Realizar Búsqueda
Actor	Cliente
Descripción	El cliente solicita uno de los servicios destinados a la búsqueda de personas (Búsqueda Simple o Avanzada) Estas búsquedas pudieran realizarse por los diferentes campos de cada persona. Introducidos los datos necesarios el sistema realiza la búsqueda según lo especificado por el cliente y devuelve un listado con los resultados.
Referencia	R2

Tabla 2.4 Caso de uso Realizar Búsqueda

CU-3	Exportar Resultado (caso de uso extendido)
Actor	Cliente
Descripción	Una vez realizada la búsqueda el cliente puede solicitar exportar el resultado de la misma hacia un formato destino cvs.
Referencia	R3

Tabla 2.5 Caso de uso Exportar Resultado.

CU-4	Salvar Búsqueda(caso de uso extendido)
Actor	Cliente
Descripción	El cliente desea salvar una búsqueda determinada y luego de llenar los campos correspondientes solicita el servicio de salvar búsquedas personalizadas bajo un nombre determinado.
Referencia	R4

Tabla 2.6 Caso de uso Salvar Búsqueda.

CU-5	Cargar Búsqueda (caso de uso extendido)
Actor	Cliente
Descripción	El cliente carga una búsqueda personalizada que ha guardado anteriormente.
Referencia	R5

Tabla 2.7 Caso de uso Cargar Búsqueda.

2.4.4 Diagrama del caso de uso del sistema.

A continuación se muestra el diagrama de casos de uso del sistema y posteriormente se realiza una breve descripción de cada uno de los casos de uso involucrados.

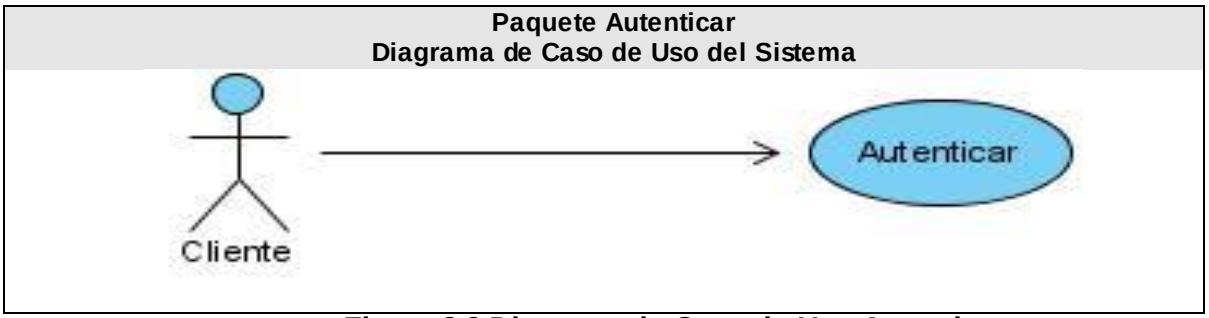

Figura 2.3 Diagrama de Caso de Uso Autenticar.

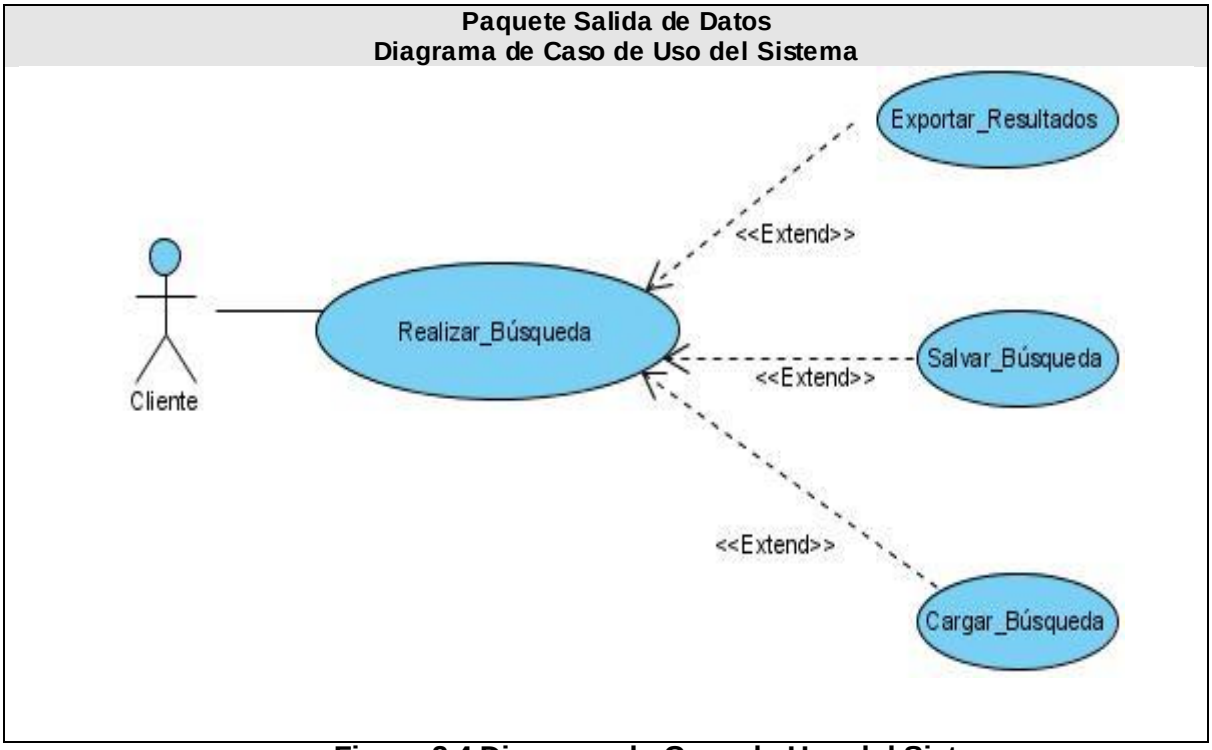

Figura 2.4 Diagrama de Caso de Uso del Sistema.

2.4.5 Descripción de los casos de uso del sistema.

"Los casos de uso son fragmentos de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales."[9](JACOBSON *et al.* 2000)

2.4.6 Descripción Caso de uso 1 Autenticar.

Nombre del caso de uso	**Autenticar**
Actores	Cliente(inicia)
Propósito	Permitir el acceso al sistema, otorgando privilegios de acceso a sus funcionalidades.

[9] Jacobson, Ivar; Booch, Grady; Rumbaugh, James.El Proceso Unificado de Ingenieria de Software. La Habana.Cuba.Editorial Felix Varela.2004

Resumen El caso de uso se inicia cuando el cliente desea utilizar el sistema para los servicios que el mismo ofrece.	
Referencia	R1
Precondición	
Poscondición	Se autentica el usuario.
Prioridad	Critico
Caso de uso asociado	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El Usuario introduce el usuario y la contraseña.	
2. El Usuario selecciona la opción “Aceptar”.	
	3. Valida que no ha dejado campos vacios. • Los campos para introducir el usuario y contraseña no se pueden dejar vacíos.
	4. Comprueba en la BD, según el servicio web que el usuario existe y verifica que la contraseña pertenezca a ese usuario.
	5. Se autentica el usuario.
	6. Se almacena los datos del usuario en la BD del sistema.
Cursos alternativos 3a Campos vacios.	
Acción del actor	**Respuesta del sistema**
	3a. 1Comprueba que ha dejado campos vacíos mostrando un mensaje de error.
Cursos alternativos 4ª Datos Incorrectos	
Acción del actor	**Respuesta del sistema**
	4a.1 Comprueba en la BD del servicio web utilizado que los datos

	de autenticación no son correctos mostrando mensaje de error. 4a.2. Indica que el nombre de usuario y/o la contraseña proporcionados no son correctos mostrando mensaje de error.

Tabla 2.8 Descripción del caso de uso Autenticar.

2.4.7 Descripción Caso de uso 2 Realizar Búsqueda.

Nombre del caso de uso	**Realizar Búsqueda**
Actores	Cliente(inicia)
Propósito	Ofrecer los servicios necesarios para la búsqueda de personas
Resumen El caso de uso se inicia cuando se solicita uno de los servicios destinados a la búsqueda de personas existente en la universidad. Estas búsquedas pueden realizarse por los diferentes campos de cada persona: nombre, usuario, CI, provincia, municipio, solapin, y tipo de persona o por cualquier combinación de campos. Introducidos los datos necesarios el sistema realiza la búsqueda según lo especificado por el cliente y devuelve un listado con los resultados.	
Referencia	R2
Precondición	El cliente está previamente autenticado.
Poscondición	Obtener los resultados devueltos por el servicio solicitado.
Prioridad	Critico
Caso de uso asociado	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
	1. Muestra las opciones de búsqueda.
2. El cliente realiza una petición de búsqueda. 2.1 Si elige la opción Búsqueda Simple ver sección:"Búsqueda Simple."	

2.2 Si elige la opción Búsqueda Avanzada ver sección:"Búsqueda Avanzada."	
Caso de uso	**Realizar Búsqueda (Sección: Búsqueda Simple**)
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
	1. Muestra el formulario para realizar la búsqueda.
2. El cliente escribe los criterios de búsqueda simple por los que desea realizar la búsqueda de una persona determinada, estos son usuario, solapin y nombre.	3. Recibe los datos y consulta los mismos con los servicios Web.
	4. Se comprueba en la BD, dado el servicio web que se consume que los datos introducidos estén activados.
	5. Obtiene el resultado y visualiza la información.
Cursos alternativos	
Acción del actor	**Respuesta del sistema**
	4a.1. Si los datos entrados están escritos incorrectamente el sistema muestra una sugerencia de los nombres que coincidan con los entrados.
	4a.2 Si los datos entrados no existen muestra un mensaje de error.
Caso de uso	**Realizar Búsquedas (Sección: Búsqueda Avanzada)**
Curso normal de eventos	

Acción del actor	**Respuesta del sistema**
	1. Muestra la ventana Búsqueda Avanzada previamente solicitada por el usuario con la cadena de búsqueda por filtros.
2. El Usuario selecciona los filtros que le quiera aplicar a la búsqueda y presiona el botón buscar.	3. Recibe los datos y consulta los Servicios Web.
	4. Se comprueba en la BD, dado el servicio web que se consume que los datos introducidos estén activados.
	5. Obtiene el resultado y visualiza la información.
6. El usuario presiona el botón Exportar. **Invoca CU Exportar Resultado.** 7. El usuario presiona el botón Salvar. **Invoca CU Salvar Búsqueda** 8. El usuario presiona el botón Cargar **Invoca CU Cargar Búsqueda.**	
Cursos alternativos 4 a Datos incorrectos	
Acción del actor	**Respuesta del sistema**
	4a.1Si los datos entrados están escritos incorrectamente el sistema muestra una sugerencia.
	4a.2 Si los datos entrados no existen muestra Resultados (0).

Tabla 2.9 Descripción del caso de uso Realizar Búsqueda.

2.4.8 Descripción Caso de uso 3 Exportar Resultado.

Nombre del caso de uso	**Exportar Resultado(caso de uso extendido)**
Actores	Cliente(inicia)
Propósito	Exportar la búsqueda realizada hacia un formato deseado.
Resumen El caso de uso se inicia cuando el cliente solicita exportar el resultado de una búsqueda realizada hacia un formato destino.	
Referencia	R3
Precondición	El cliente esta previamente autentificado y se ha realizado una determinada búsqueda.
Poscondición	Exportar la búsqueda realizada.
Prioridad	Critico
Caso de uso asociado	CU2
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El cliente presiona el botón exportar resultado de una búsqueda determinada.	2. El sistema muestra una ventana para escoger el formato al cual quiere exportar este resultado.
2. El cliente escoge el formato y presiona el botón aceptar.	3. El sistema exporta el resultado de la búsqueda realizada hacia el formato escogido.

Tabla 2.10 Descripción del caso de uso Exportar Resultado.

2.4.9 Descripción Caso de uso 4 Salvar Búsqueda.

Nombre del caso de uso	**Salvar Búsquedas (caso de uso extendido)**

Actores	Cliente(inicia)
Propósito	Salvar la combinación de campos para lo cual se realiza una determinada búsqueda, y así la próxima vez que el cliente desee realizar la misma solo la llama con su nombre guardado en la base de datos.
Resumen El caso de uso se inicia cuando el cliente desea realizar una búsqueda determinada y luego de llenar los campos correspondientes solicita el servicio de salvar búsquedas personalizadas bajo un nombre determinado.	
Referencia	R4
Precondición	El cliente esta previamente autentificado y ha llenado campos en el formulario de búsqueda avanzada.
Poscondición	Salvar las búsquedas realizadas.
Prioridad	Critico
Caso de uso asociado	CU2
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El cliente llena los campos para realizar una búsqueda en la sección Avanzada.	2. El sistema muestra la ventana bajo el nombre que se quiere salvar la búsqueda.
3. Entra el nombre de la búsqueda y presiona el botón Salvar.	4. Guarda la combinación de campos para el cual se realiza la búsqueda bajo este nombre en la base de datos.

Tabla 2.11 Descripción del caso de uso Salvar Búsquedas.

2.4.10 Descripción Caso de uso 5 Cargar Búsqueda.

Nombre del caso de uso	**Cargar Búsquedas(caso de uso extendido)**
Actores	Cliente(inicia)

Propósito	Cargar las búsquedas salvadas con anterioridad por el cliente.
Resumen El caso de uso se inicia cuando el cliente desea cargar una búsqueda personalizada ya bajo un nombre determinado que ha guardado anteriormente.	
Referencia	R5
Precondición	El cliente esta previamente autentificado y la debe existir alguna búsqueda guardada.
Poscondición	Cargar las búsquedas salvadas.
Prioridad	Secundario.
Caso de uso asociado	CU2
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El cliente selecciona la búsqueda que quiere abrir o cargar	
2. El cliente presiona el botón cargar.	3. El sistema muestra la combinación de campos en la sección Avanzada que ha sido guardadas por el cliente, con la opción de modificar la misma.

Tabla 2.12 Descripción del caso de uso Cargar Búsquedas.

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

"Análisis designa la comprensión de un problema o situación, mientras que diseño se relaciona con la creación de una solución para el problema analizado; un modelo es una simplificación utilizada para comprender mejor el problema (modelo de análisis) o la solución (modelo de diseño)" (33)

El binomio análisis-diseño ha sido tradicionalmente utilizado en ingeniería del software para estructurar las actividades del proceso de desarrollo de software. En el presente capítulo se desarrolla los distintos diagramas de clases y de interacción que se generan en ambos flujos de trabajo.

3.2 Modelo de Análisis.

3.2.1 Definición del modelo de clases de análisis.

El Modelo de Análisis puede definirse de varias formas. Los modelos representan a pequeña escala y de manera gráfica fenómenos que suceden en la vida real. Para diseñar un modelo es necesario establecer una serie de categorías y definir el tipo de relaciones que se establecen entre ellas.

Las clases del análisis siempre encajan en uno de tres estereotipos básicos: de interfaz; de control o de entidad. Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores, las clases entidad se utilizan para modelar información que posee una vida larga y que es a menudo persistente y las clases de control representan coordinación, secuencia, transacciones y control de otros objetos.

3.2.2 Diagrama de clases de análisis.

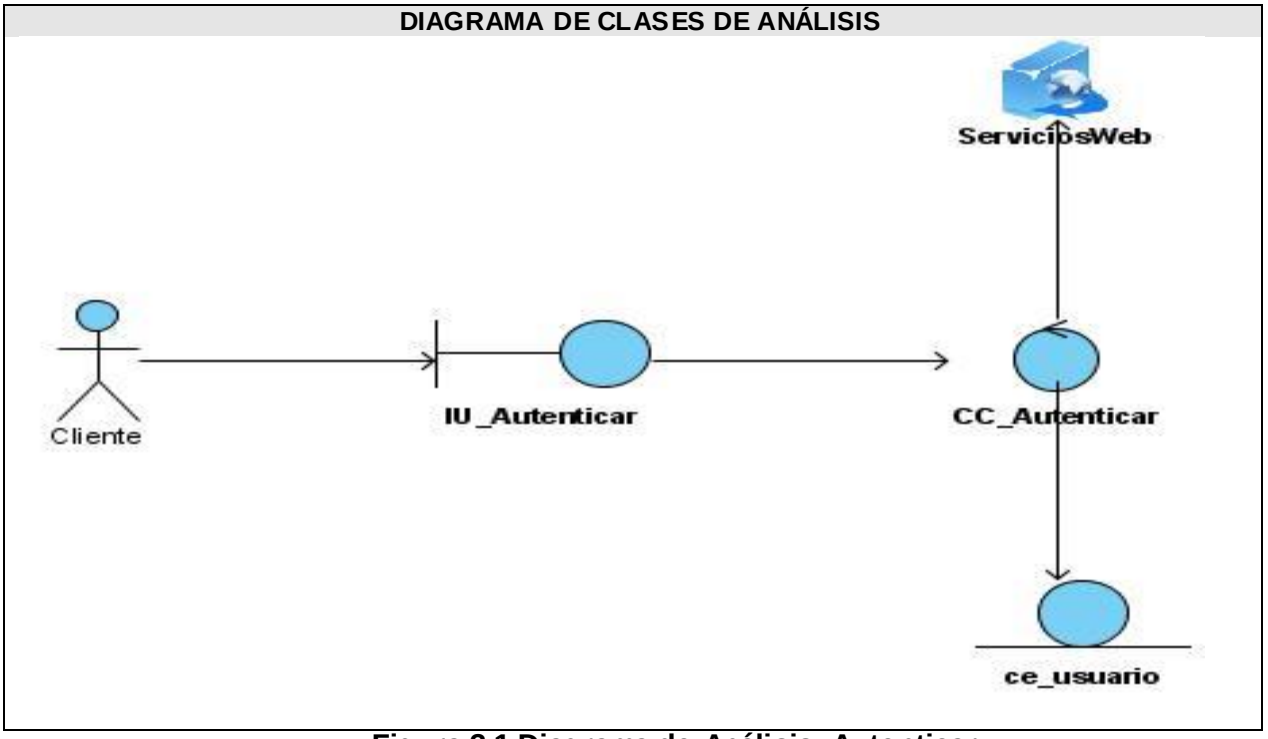

Figura 3.1 Diagrama de Análisis Autenticar.

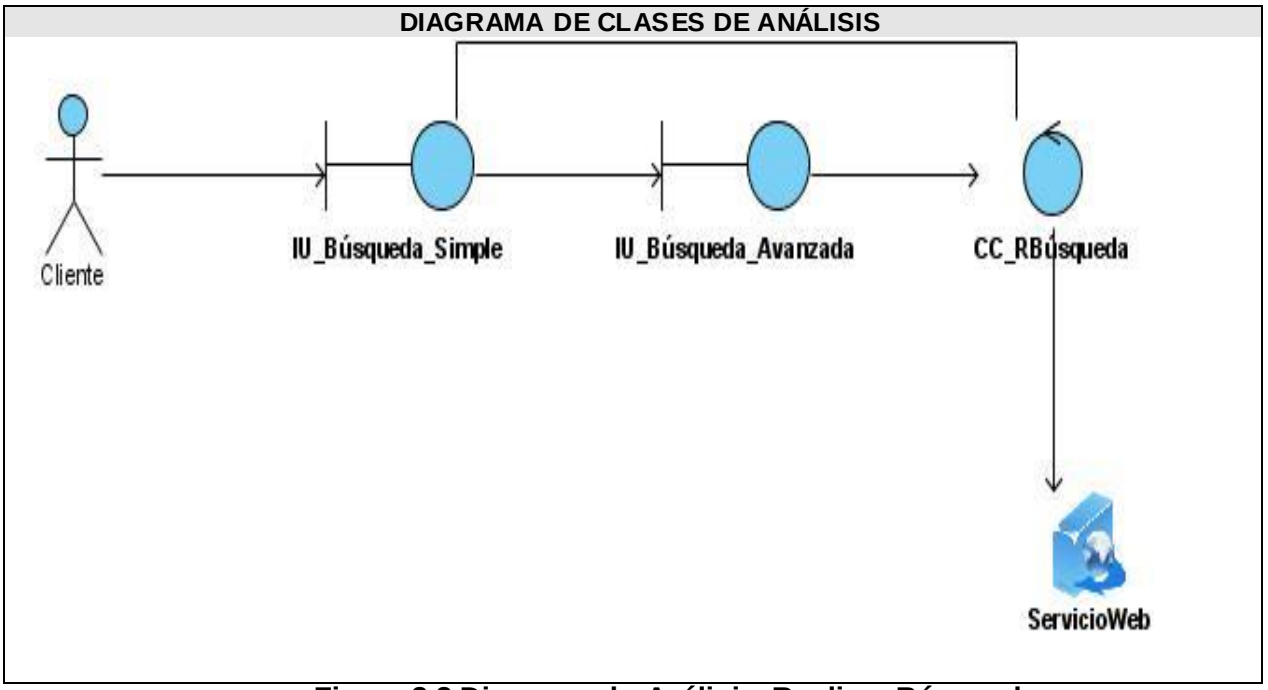

Figura 3.2 Diagrama de Análisis Realizar Búsqueda.

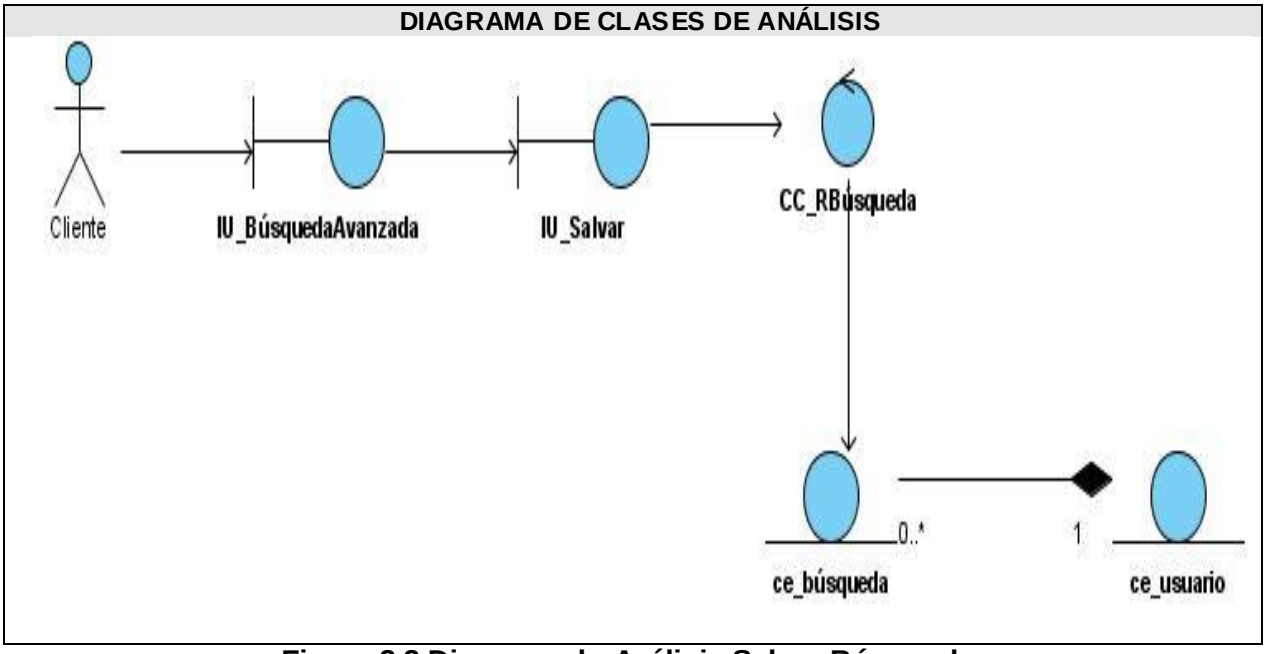

Figura 3.3 Diagrama de Análisis Salvar Búsqueda.

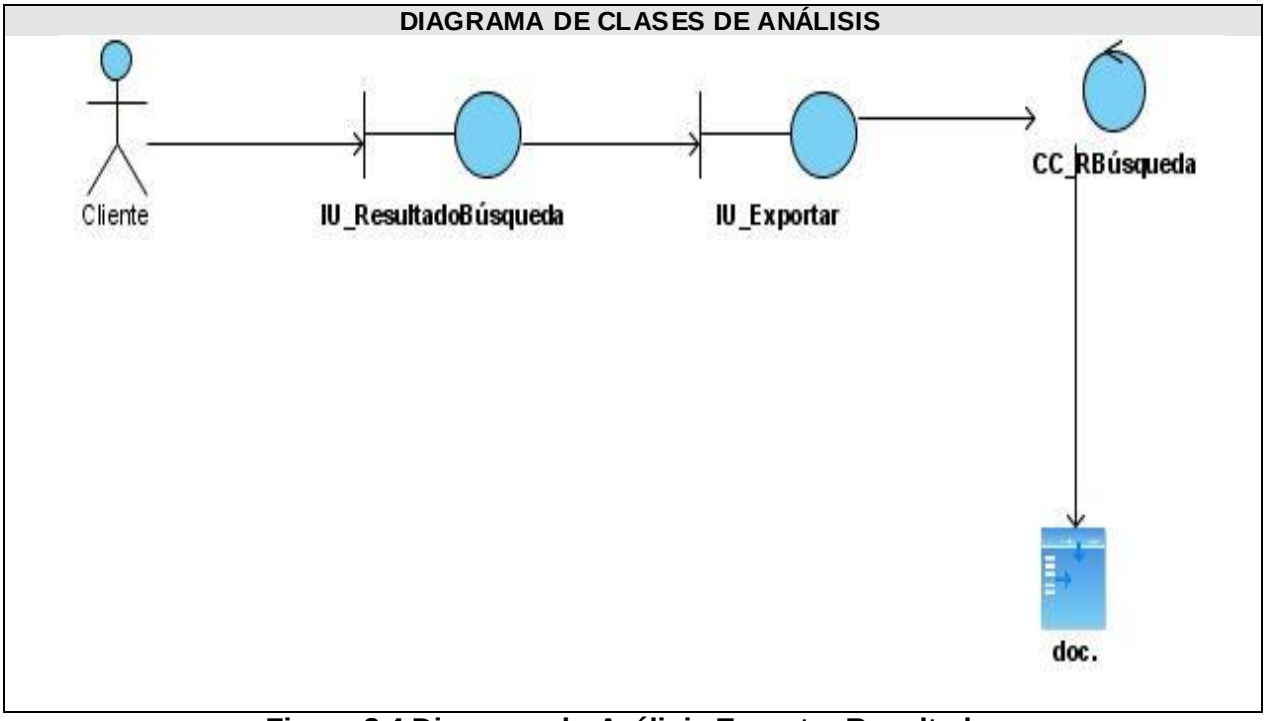

Figura 3.4 Diagrama de Análisis Exportar Resultado.

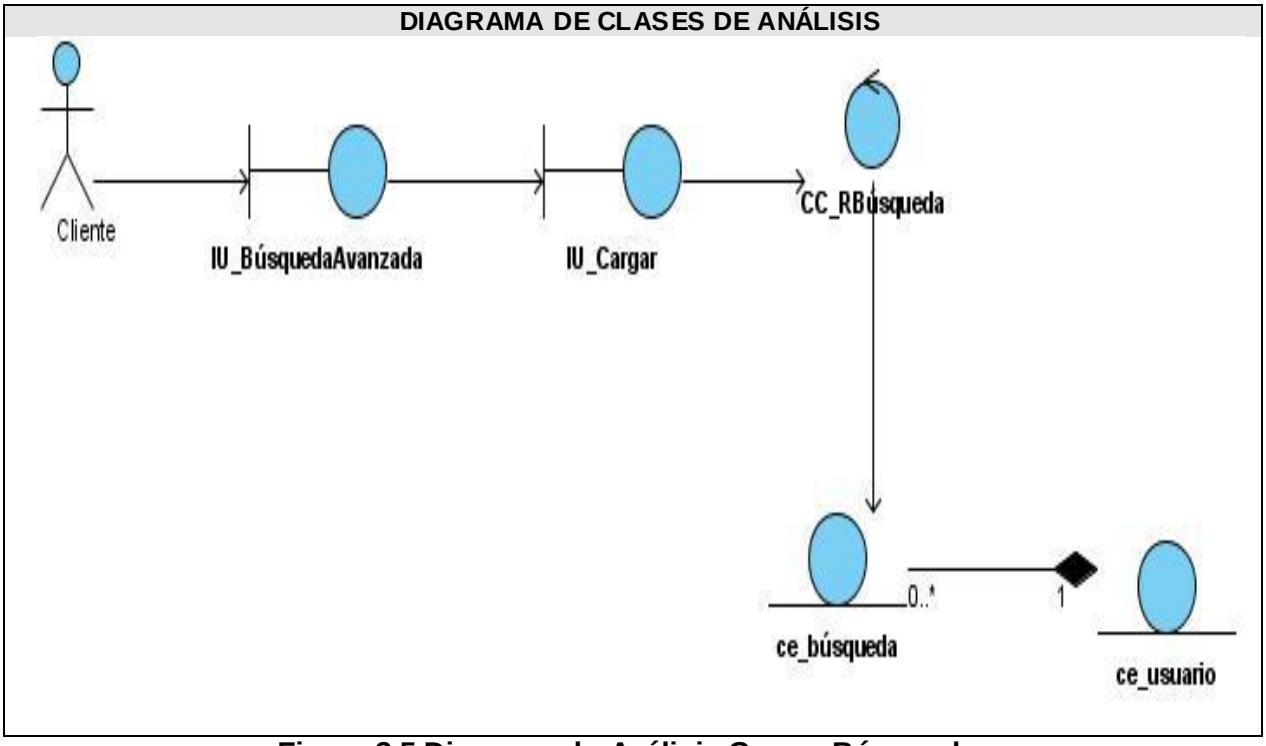

Figura 3.5 Diagrama de Análisis Cargar Búsqueda.

3.3 Diagrama de clases del diseño

3.3.1 Diagrama de interacción.

Los diagramas de interacción describen la forma en que cada operación detectada en los diagramas de secuencia lleva a cabo sus responsabilidades y modifica el estado del sistema, mostrando el modo en que los objetos interactúan a través de mensajes. En UML los diagramas de interacción pueden representarse a través de los Diagramas de Colaboración y/o de los Diagramas de Secuencia Los Diagramas de Secuencia muestran interacciones entre objetos basadas en el tiempo.
Ver **Anexo 1.**

3.3.2 Diagrama de clases web del Diseño.

El diagrama de clases es una descripción de los modelos de objetos, contiene clases y las relaciones estructurales y de herencia existentes entre ellas. Este se obtiene como resultado del refinamiento del modelo conceptual y de los diagramas de secuencia. La definición de clase incluye definiciones para atributos y responsabilidades.

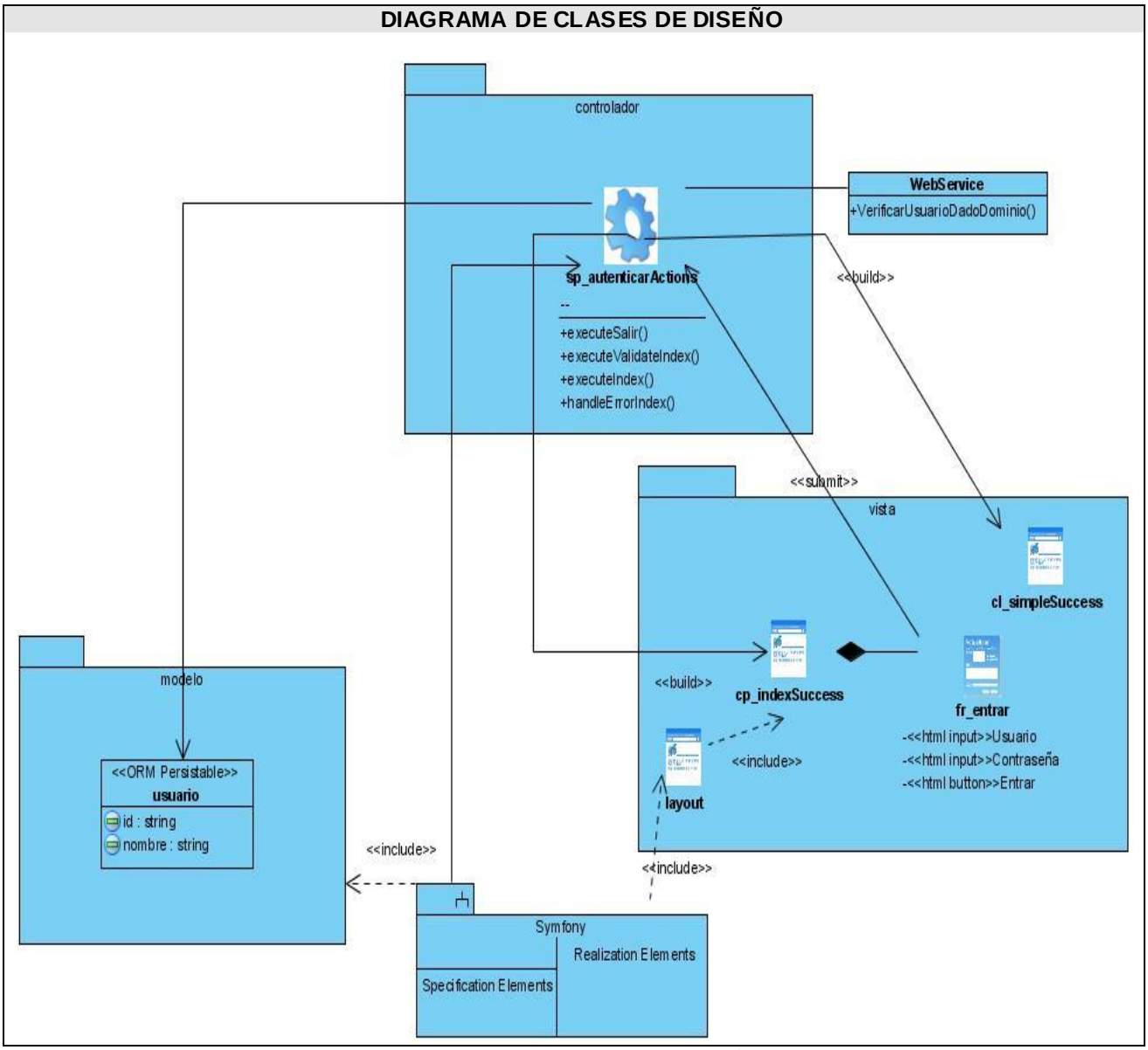

Figura 3.12 Diagrama de Diseño Autenticar.

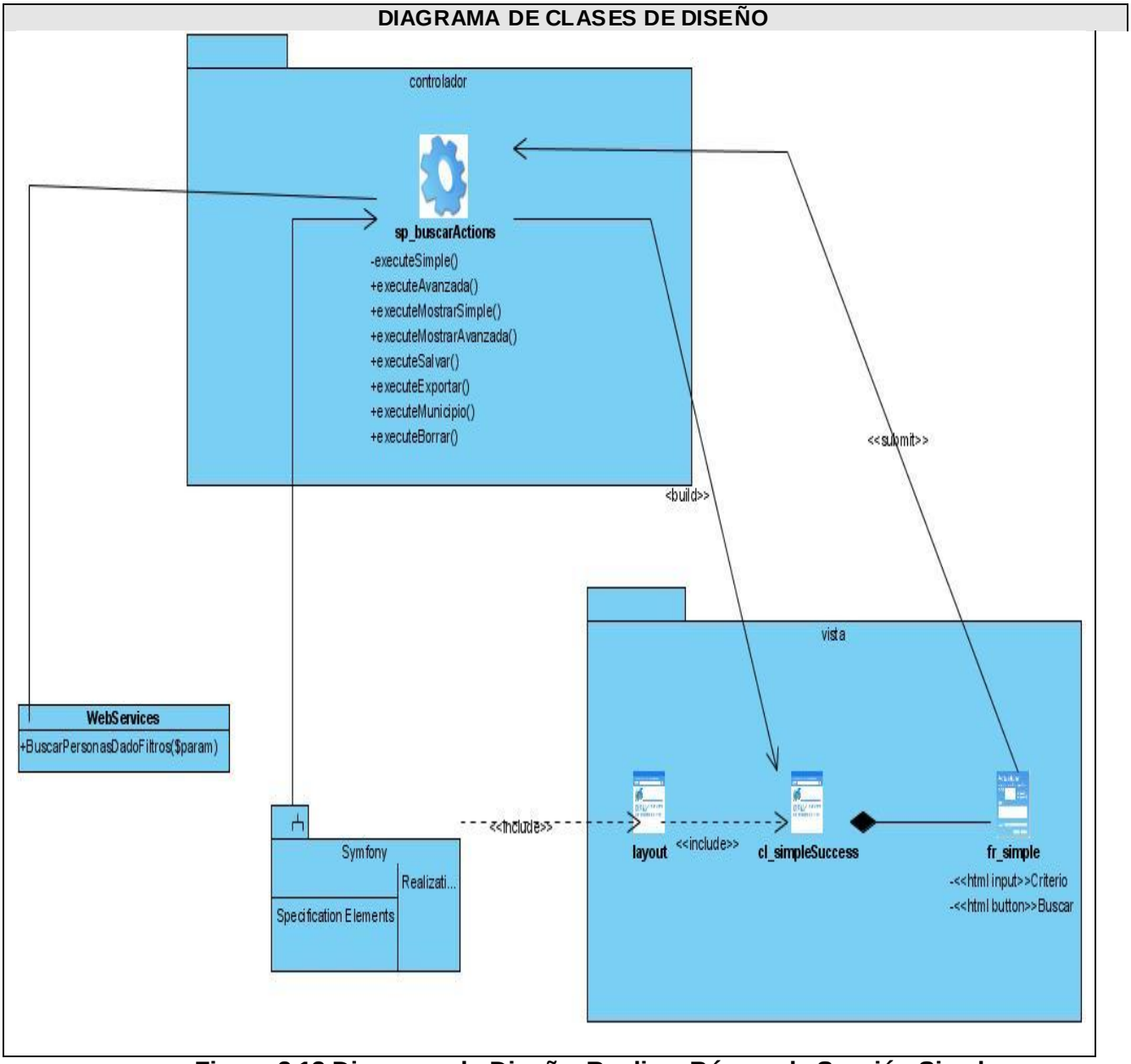

Figura 3.13 Diagrama de Diseño Realizar Búsqueda Sección Simple.

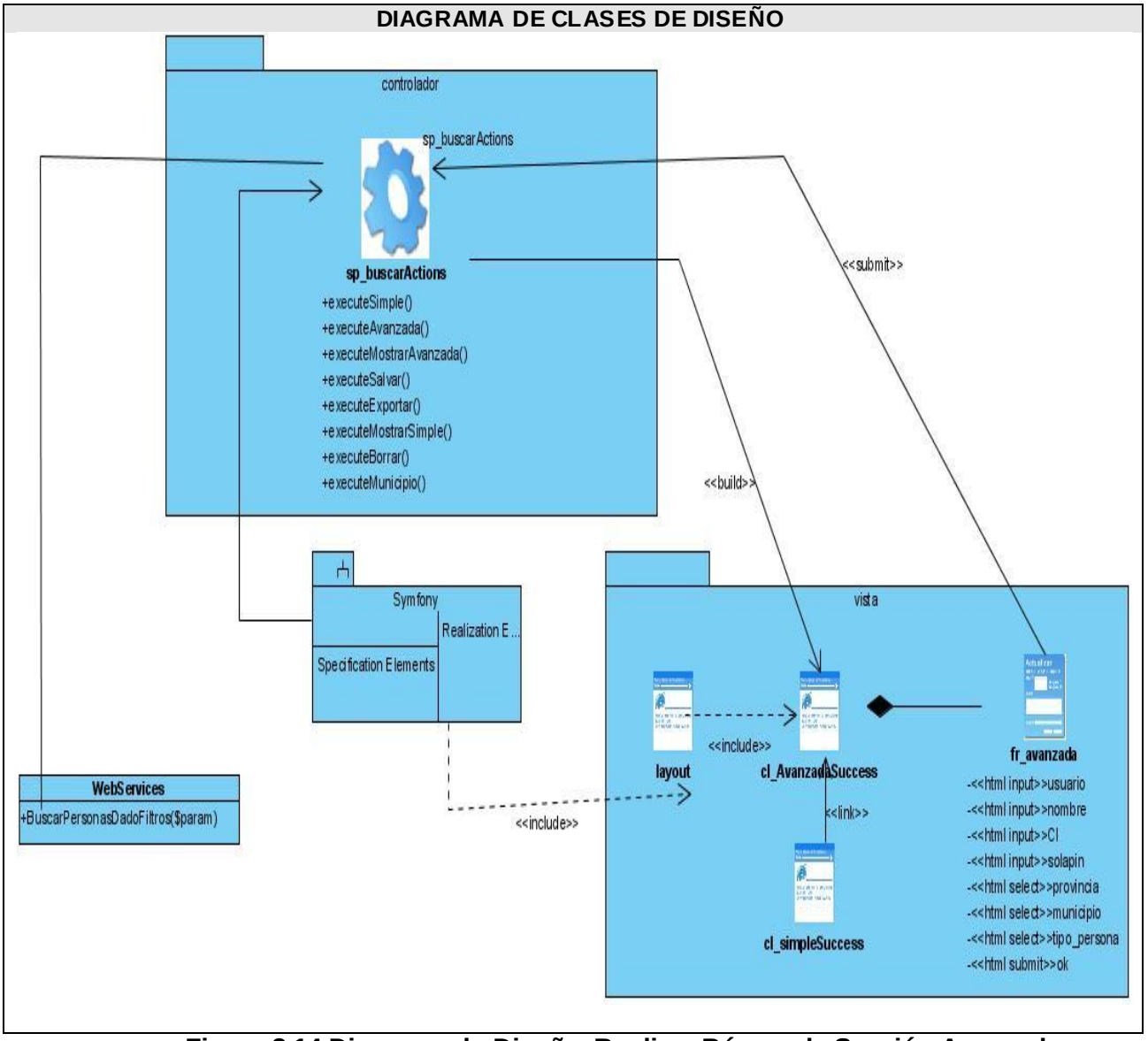

Figura 3.14 Diagrama de Diseño Realizar Búsqueda Sección Avanzada.

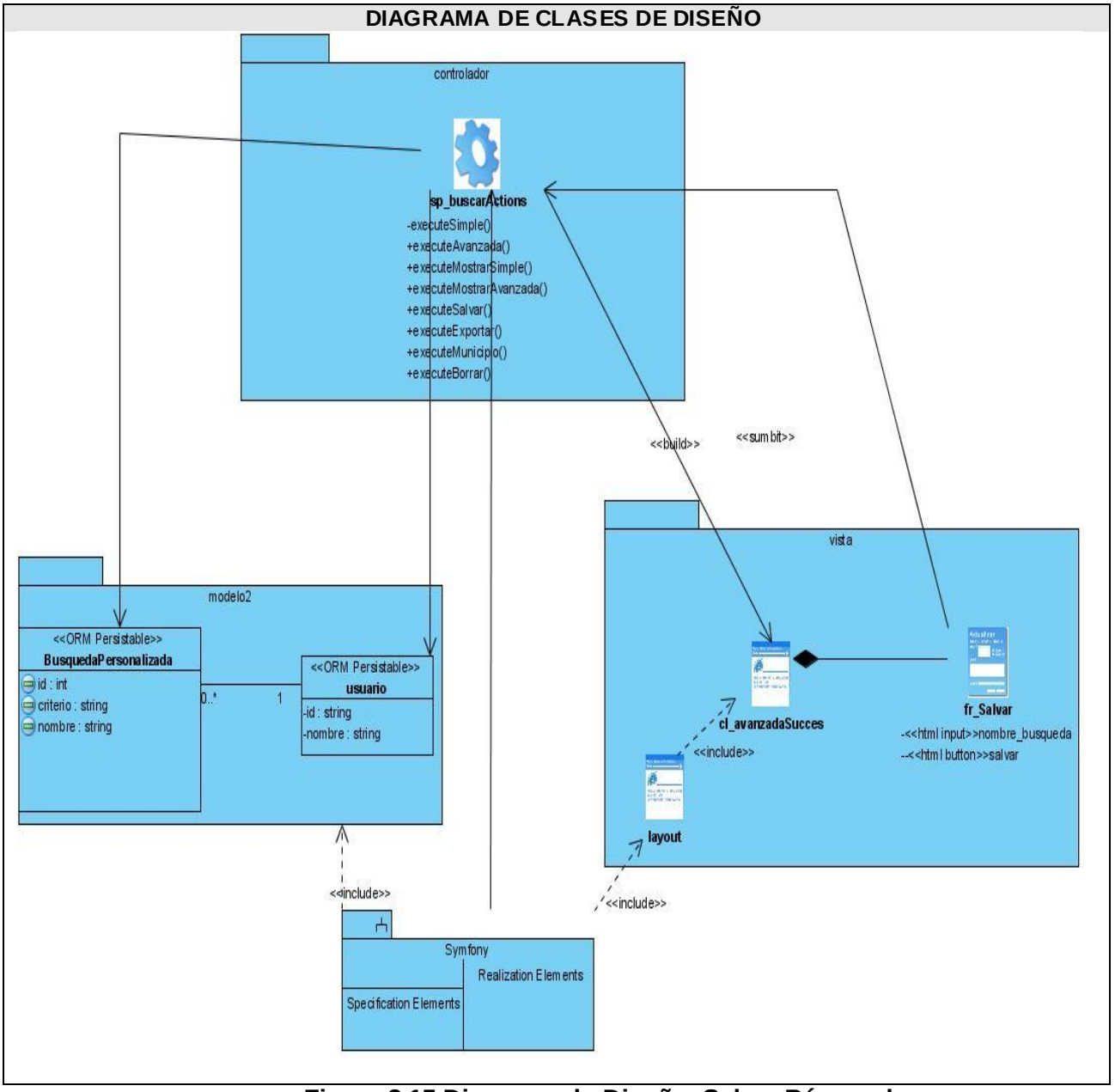

Figura 3.15 Diagrama de Diseño Salvar Búsqueda.

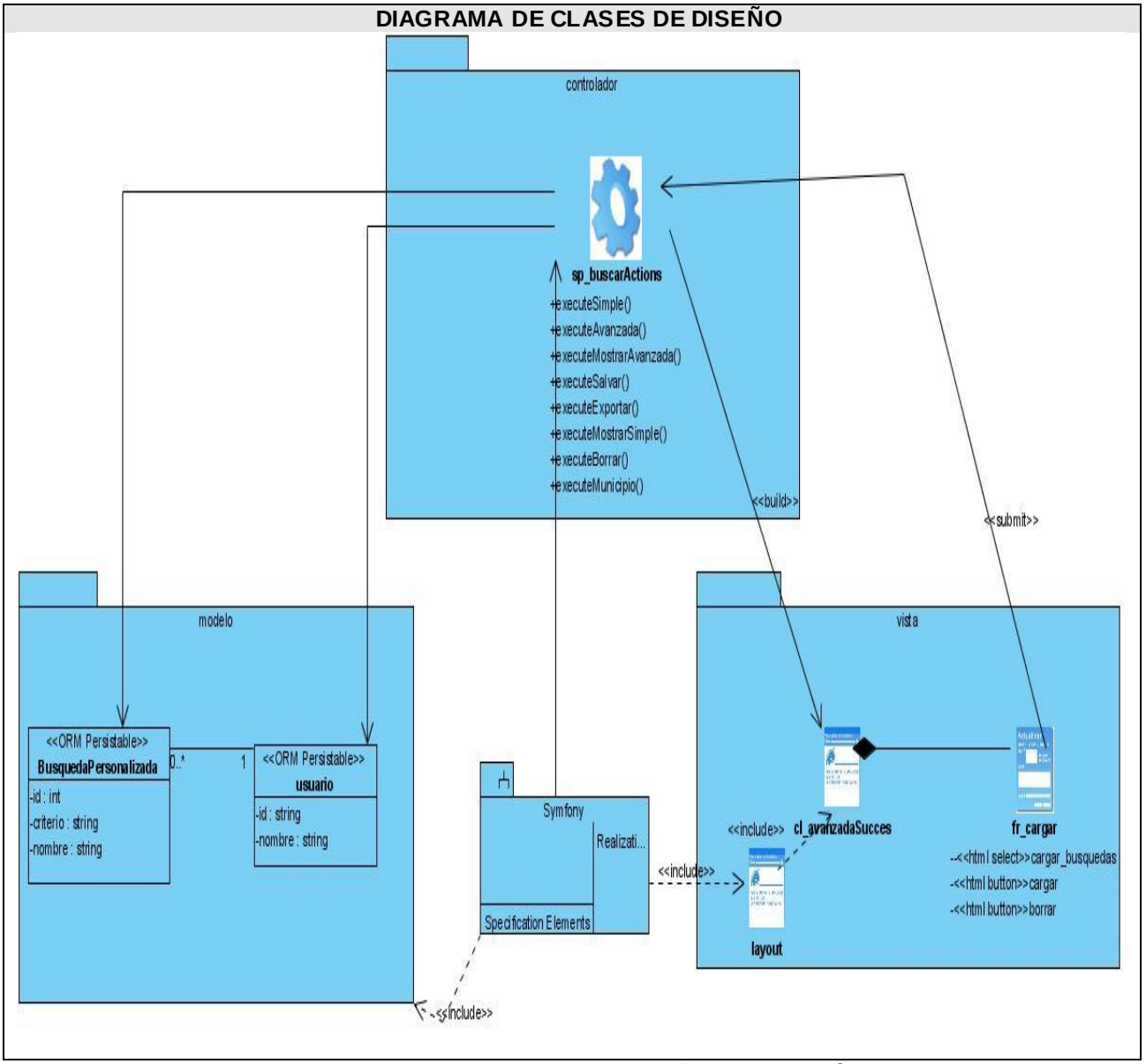

Figura 3.16 Diagrama de Diseño Cargar Búsqueda.

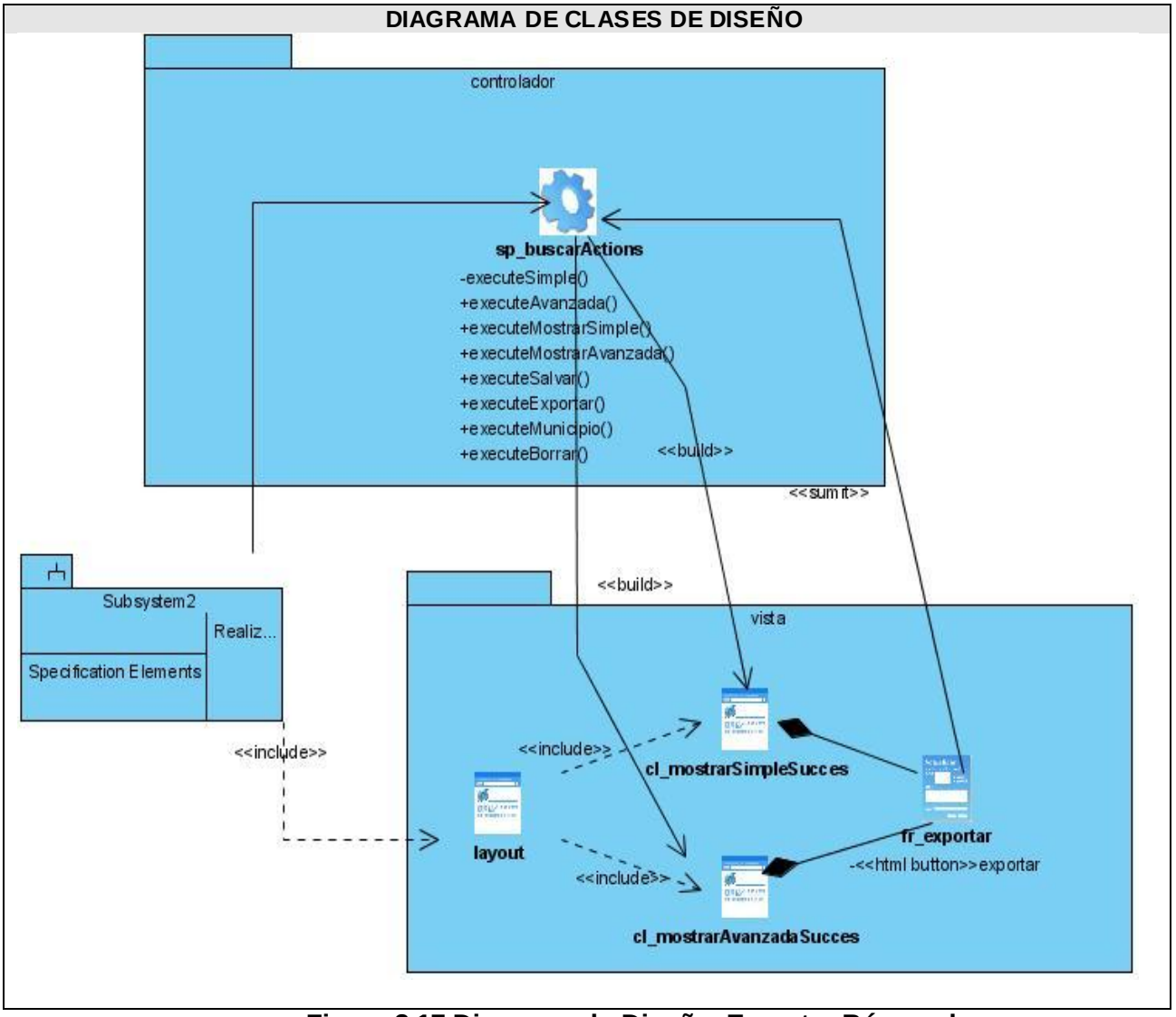

Figura 3.17 Diagrama de Diseño Exportar Búsqueda.

3.3.3 Descripción de las clases web del diseño.

Clases controladoras.

Nombre:	**Sp_autenticarActions**
Tipo de clase	**Server Page (Clase Servidora que tiene los métodos principales y funciones que hacen llamada a las clases de acceso a datos generada por el framework)**
Atributo	**Tipo**
No aplica	
Para cada responsabilidad:	
Nombre:	**executeIndex()**
Descripción:	**Levanta la página principal para que el usuario permita autenticarse, si es correcto levanta la página de Búsqueda Simple.**
Nombre:	**validateIndex()**
Descripción:	**Devuelve si el usuario autenticado es correcto o no comprobando con el servicio web, posteriormente salva el id del usuario en la BD**
Nombre:	**executeSalir()**
Descripción:	**Destruye los datos de la sesión del usuario y este no puede acceder al sitio hasta que no se autentique otra vez.**
Nombre:	**handleErrorIndex()**
Descripción:	**Redirecciona la página de autenticación si hay un error en el proceso de autenticarse.**

Tabla 3.1 Clase controladora Autenticar.

Nombre:	**Sp_buscarActions**
Tipo de clase	**Server Page (Clase Servidora que tiene los métodos principales y funciones que hacen llamada a las clases de acceso a datos generada por el framework)**

Atributo	**Tipo**
No aplica	

Para cada responsabilidad:	
Nombre:	**executeSimple()**
Descripción:	**Levanta la página de Búsqueda Simple si el usuario ha sido autenticado satisfactoriamente**
Nombre:	**executeMostrarSimple()**
Descripción:	**Devuelve el resultado de la búsqueda simple mediante los criterios nombre, solapin y usuario, comprobando su existencia con el servicio web.**
Nombre:	**executeAvanzada()**
Descripción:	**Levanta la página de Búsqueda Avanzada y realiza la funcionalidad de cargar los valores de una búsqueda avanzada guardada con la opción de modificarlos.**
Nombre:	**executeSalvar()**
Descripción:	**Salva la combinación de campos de la Búsqueda Avanzada en la BD con un nombre.**
Nombre:	**executeExportar()**
	Exporta el resultado de cualquier búsqueda hacia un formato de texto.
Nombre:	**executeBorrar()**
Descripción:	**Borra las búsquedas salvadas por el usuario.**
Nombre:	**executeMunicipio()**
Descripción:	**Devuelve dada la provincia los municipios de esta.**
Nombre:	**executeMostrarAvanzada()**
Descripción:	**Devuelve el resultado de la Búsqueda Avanzada por los campos seleccionados por el usuario.**

Tabla 3.2 Clase controladora Buscar.

Clases de Acceso a Datos

Nombre :	**BaseBusquedaPersonalizada**
Tipo de clase	**Clases de Acceso a Datos (Clases generadas por el framework que tiene los métodos de acceso a datos)**

Atributo	**Tipo**
$id	**protected**
$usuario_id	**protected**
$criterio	**protected**
$nombre	**protected**

Para cada responsabilidad:	
Nombre:	**getId()**
Descripción:	**Devuelve el identificador de la búsqueda.**
Nombre:	**getNombre()**
Descripción:	**Devuelve el nombre de la búsqueda realizada por el usuario.**
Nombre:	**getUsuarioId()**
Descripción:	**Devuelve el identificador del usuario que ha salvado esa búsqueda.**
Nombre:	**getCriterio()**
Descripción:	**Devuelve la combinación de campos que escogió el usuario para salvar su Búsqueda personalizada.**
Nombre:	**save()**
Descripción	**Salva la búsqueda personalizada.**
Nombre	**setId($v)**
Descripción	**Modifica el valor del atributo identificador.**
Nombre	**setNombre($v)**
Descripción	**Modifica el valor del atributo nombre.**
Nombre	**setUsuarioId($v)**
Descripción	**Modifica el valor del atributo identificador del usuario.**
Nombre	**setCriterio($v)**
Descripción	**Modifica el valor del atributo criterio de búsqueda.**

Tabla 3.3 Clase Acceso a Datos BaseBusquedaPersonalizada.

Nombre :	BaseBusquedaPersonalizadaPeer
Tipo de clase	Clases de Acceso a Datos (Clases generadas por el framework que tiene los métodos de acceso a datos)
Atributo	Tipo
No aplica	
Para cada responsabilidad:	
Nombre:	retrieveByPK($pk, $con = null)
Descripción:	Busca las diferentes búsquedas por identificador.

Tabla 3.4 Clase Acceso a Datos BaseBusquedaPersonalizadaPeer

Nombre :	BaseUsuario
Tipo de clase	Clases de Acceso a Datos (Clases generadas por el framework que tiene los métodos de acceso a datos)
Atributo	Tipo
$id	protected
$nombre	protected
Para cada responsabilidad:	
Nombre:	getId()
Descripción:	Devuelve el identificador del usuario.
Nombre:	getNombre()
Descripción:	Devuelve el nombre del usuario.
Nombre:	save()
Descripción	Salva el usuario.
Nombre	setId($v)
Descripción	Modifica el valor del atributo identificador.
Nombre	setNombre($v)
Descripción	Modifica el valor del atributo nombre.

Tabla 3.5 Clase Acceso a Datos BaseUsuario

Nombre :	**BaseUsuarioPeer**	
Tipo de clase	**Clases de Acceso a Datos (Clases generadas por el framework que tiene los métodos de acceso a datos)**	
Atributo		**Tipo**
No aplica		
Para cada responsabilidad:		
Nombre:	**retrieveByPK($pk, $con = null)**	
Descripción:	**Busca los diferentes usuarios por identificador.**	

Tabla 3.6 Clase Acceso a Datos BaseUsuarioPeer

3.4 Diseño de la Base de Datos.

3.4.1 Modelo Lógico de Datos.

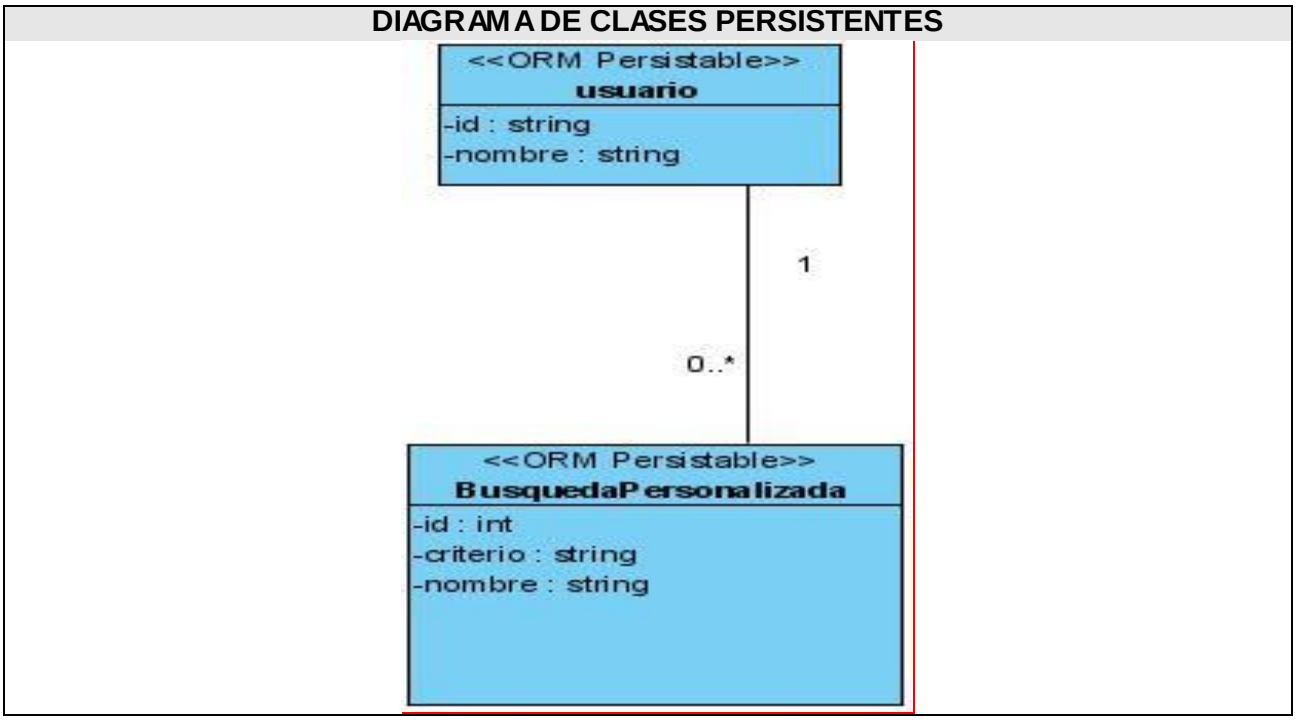

Figura 3.18 Diagrama de Clases Persistentes.

3.4.2 Modelo Físico de Datos.

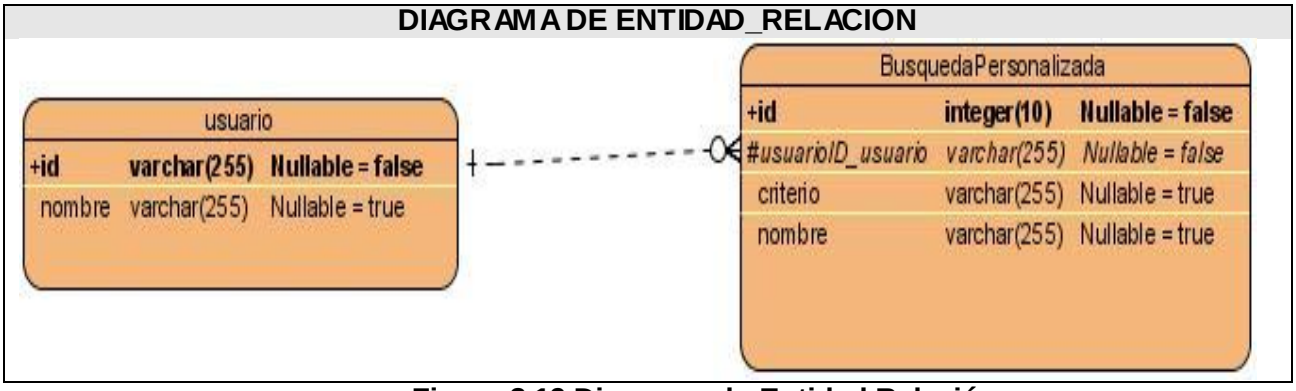

Figura 3.19 Diagrama de Entidad Relación.

3.4.3 Descripción de las tablas.

Nombre: Usuario		
Descripción: En esa tabla se almacenan los datos de los usuarios.		
Atributo	**Tipo**	**Descripción**
id	string	Identificador del usuario
nombre	string	Nombre del usuario

Tabla 3.7 Tabla Entidad Usuario

Nombre: BúsquedaPersonalizada		
Descripción: En esa tabla se almacenan las diferentes búsquedas realizadas por los usuarios.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador de la búsqueda.
criterio	string	Criterio de búsqueda.
nombre	string	Nombre de la búsqueda.
usuario_id	string	Identificador del usuario correspondiente a la búsqueda salvada.

Tabla 3.8 Tabla Entidad Búsqueda Personalizada.

3.5 Conclusiones.

En este capítulo se han llevado a cabo los flujos de trabajo de Análisis y Diseño que propone la metodología RUP, modelando los diagramas fundamentales de ambos flujos, se realizó la descripción de las clases del diseño y de las tablas de la base de datos al ser definido el modelo de datos.

CAPÍTULO IV: IMPLEMENTACIÓN

4.1 Introducción.

En este capítulo se adentra en la implementación del sistema propuesto donde se abordan las características del mismo. Se estructura el modelo de implementación a través del diagrama de despliegue y el diagrama de componentes por lo que dentro de los elementos está el modelado de los elementos de procesado en tiempo de ejecución, y de los componentes, procesos y objetos de software que viven en ellos y sus relaciones.

4.2 Diagrama de Despliegue.

Los Diagramas de Despliegue se usan para modelar la configuración de los elementos de procesado en tiempo de ejecución y de los componentes, procesos y objetos de software que viven en ellos. En el diagrama 'Despliegue', se modelan nodos físicos y las asociaciones de comunicación que existen entre ellos. Para cada nodo, puedes indicar qué instancias de componentes viven o corren (se ejecutan) en el nodo.

Un nodo se representa como un cubo, un nodo es un elemento donde se ejecutan los componentes, representan el despliegue físico de estos componentes. El diagrama de despliegue muestra la topología del hardware sobre el que se ejecuta el sistema.

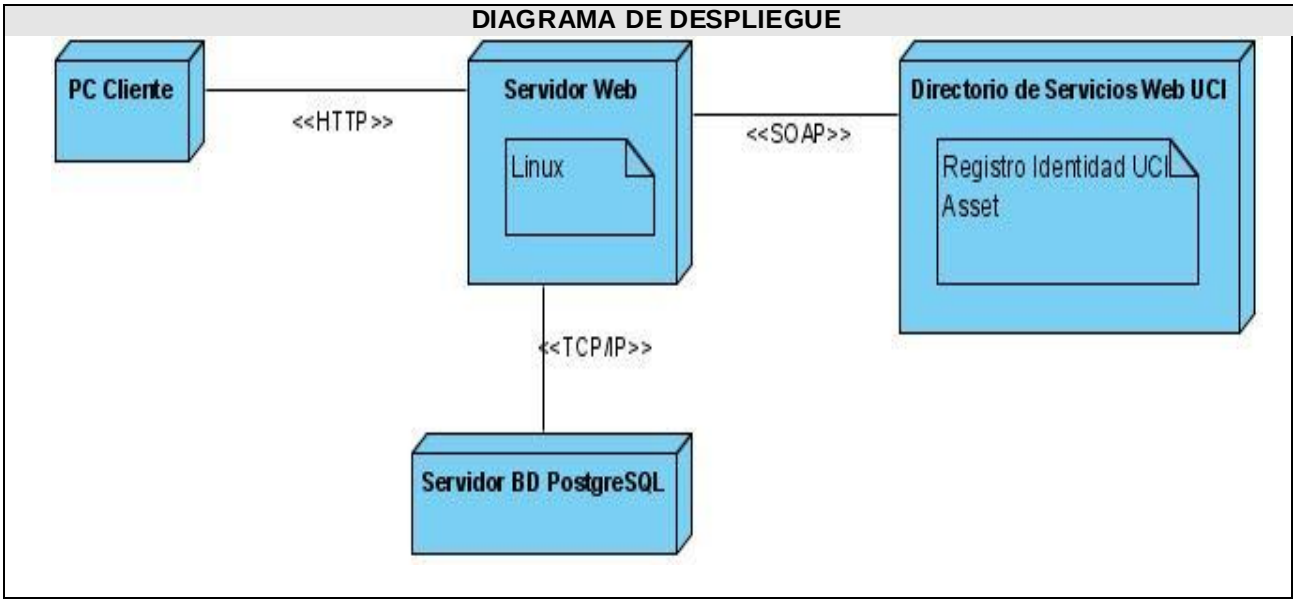

Figura 4.1 Diagrama de Despliegue.

4.3 Diagrama de Componente.

Los diagramas de componentes describen los elementos físicos del sistema y su relación muestra las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables. El diagrama de componente hace parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su despliegue en nodos de ejecución. Esta vista proporciona la oportunidad de establecer correspondencias entre las clases y los componentes de implementación y nodos. La vista de implementación se representa con los diagramas de componentes.

Se puede señalar que son en la arquitectura física la implementación de los conceptos y en la arquitectura lógica representan las funcionalidades descritas (clases, objetos, sus relaciones, y colaboraciones).

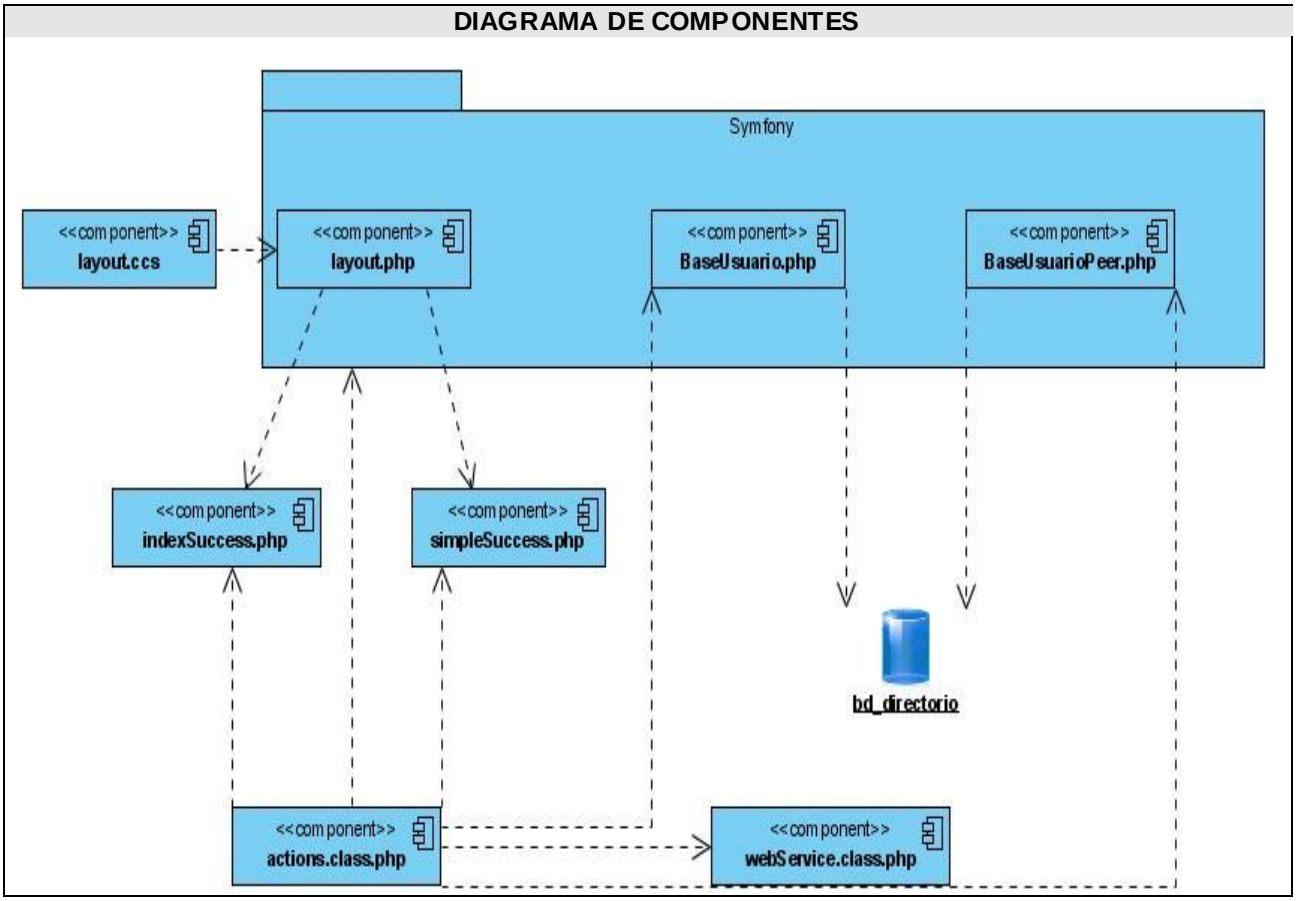

Figura 4.2 Diagrama de Componente subsistema Autenticar.

DIAGRAMA DE COMPONENTES

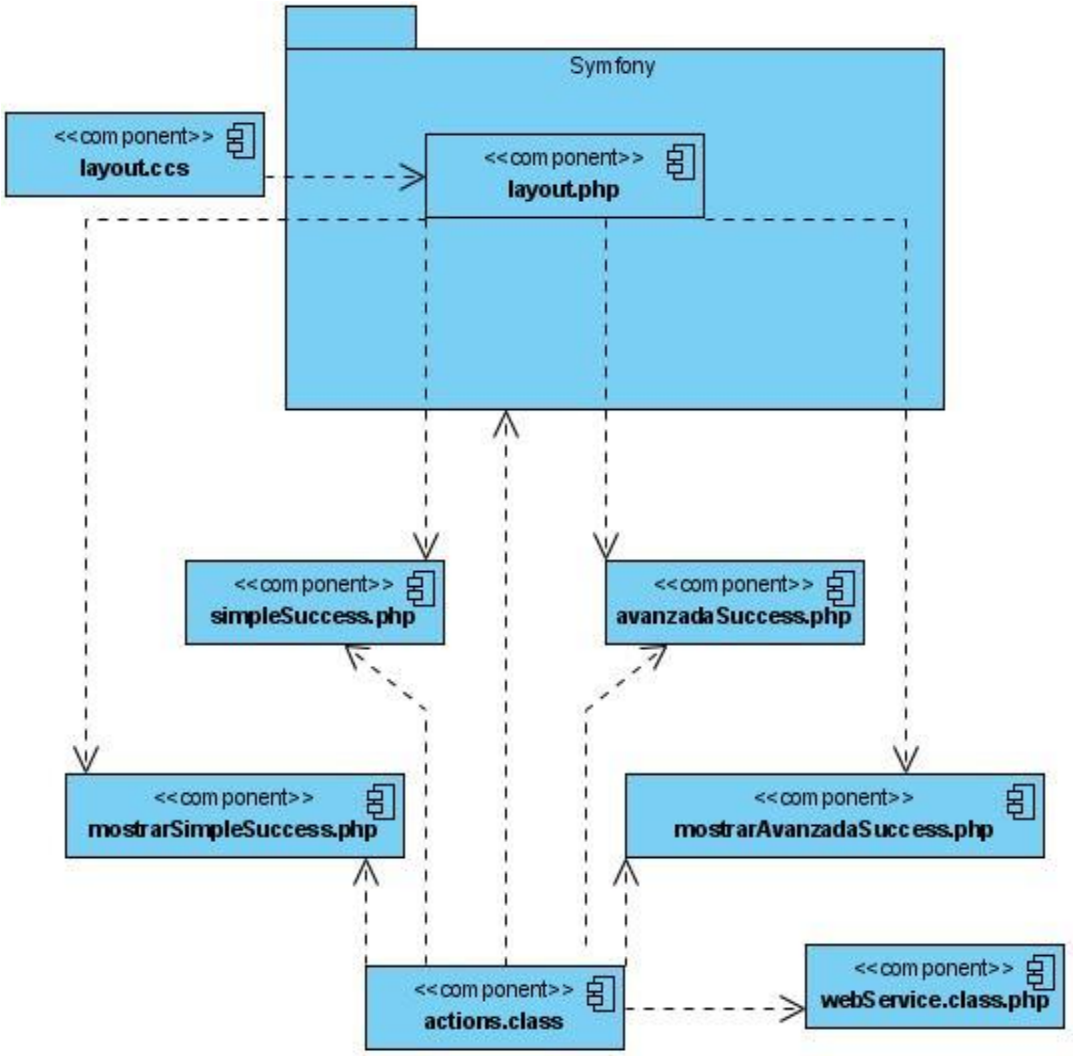

Figura 4.3 Diagrama de Componente subsistema Realizar Búsqueda.

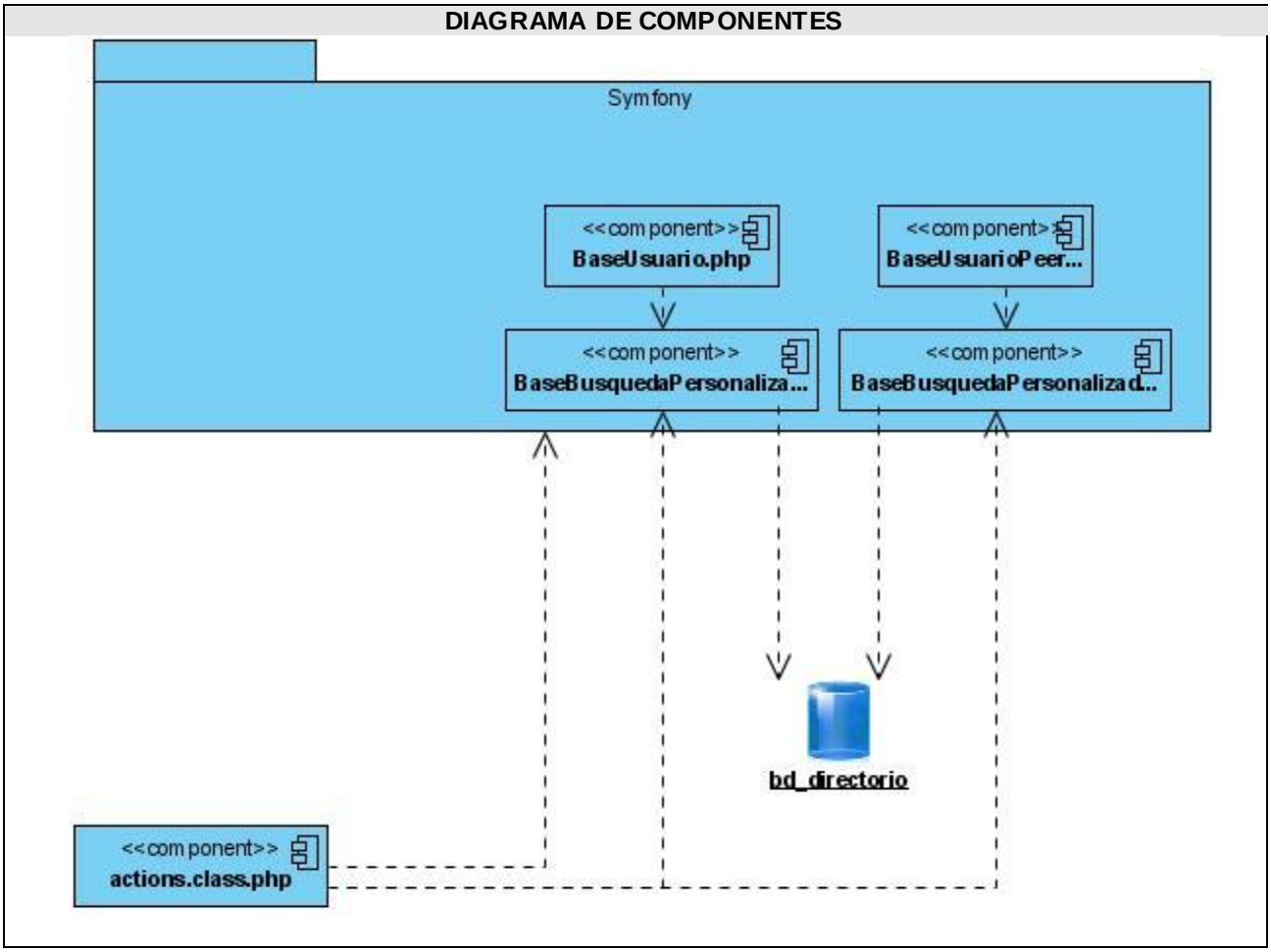

Figura 4.4 Diagrama de Componente subsistema Datos.

4.4 Conclusiones.

Con la culminación de este capítulo se han obtenido los diferentes diagramas que dieron lugar a la obtención del modelo de implementación final. El modelo de despliegue, que describe las configuraciones sobre las cuales deberá implementarse el sistema y el diagrama de componentes que ilustra los componentes de software que se usan para construir el sistema .en el cual se describe su organización y dependencia entre nodos físicos en los que funcionará a aplicación.

CAPÍTULO V: ESTUDIO DE FACTIBILIDAD

5.1 Introducción

Para poder planificar un proyecto software, es imprescindible previamente estimar su tamaño y el costo y esfuerzo que se emplea para lograrlo; siguiendo el adagio de que un buen proyecto es aquel que fue bien planificado; cobra importancia capital la estimación. (34)

Para la estimación del tamaño de un sistema a partir de sus requerimientos, entre las técnicas más difundidas se encuentran el Análisis de Puntos de Función, COCOMO II y una variante más reciente Análisis de Puntos de Casos de Uso.

En este capítulo se describe la estimación de costos del sistema propuesto y sus beneficios empleando esta última.

5.2 Estimación basada en análisis de Puntos de Casos de Uso.

La estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner de Objectory AB, y posteriormente refinado por muchos otros autores. Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. (35)

A continuación, se detallan los pasos a seguir para la aplicación de éste método.

Paso1 Cálculo de Puntos de Casos de Uso sin ajustar

UUCP = UAW + UUCW

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin ajustar

Factor de Peso de los Actores sin ajustar (UAW)

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece teniendo en cuenta en

primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema.

Tipo de Actor	Descripción	Actor	Factor de Peso
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	1	3

Tabla 5.1 Tabla Factor de Peso de los Actores sin ajustar (UAW)

UAW = 1 x 3 = 3

Factor de Peso de los Casos de Uso sin ajustar (UUCW)

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia.

Tipo de Caso de Uso	Descripción	Cant.	Factor de Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	5

Tabla 5.2 Tabla Factor de Peso de los Casos de Uso sin ajustar (UUCW)

UUCW = 5 x 5 = 25

UUCP = UAW + UUCW

UUCP = 3 + 25 = 28

Paso 2 Cálculo de Puntos de Casos de Uso ajustados

UCP = UUCP x TCF x EF donde,

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar
TCF: Factor de complejidad técnica
EF: Factor de ambiente

Factor de complejidad técnica (TCF)

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante.

Factor	Descripción	Peso	Valor	Total	Comentario
T1	Sistema distribuido	2	0	0	El sistema no es distribuido
T2	Tiempo de respuesta	1	4	4	Debe ser bastante rápido
T3	Eficiencia del usuario final	1	4	4	
T4	Procesamiento interno complejo	1	2	2	No es complejo
T5	El código debe ser reutilizable	1	5	5	Se requiere que el código sea reutilizable
T6	Facilidad de instalación	0.5	4	2	Escasos requerimientos de facilidad de instalación
T7	Facilidad de uso	0.5	4	2	Buena
T8	Portabilidad	2	5	10	
T9	Facilidad de cambio	1	3	3	Se requiere un costo normal de mantenimiento
T10	Concurrencia	1	5	5	No hay concurrencia
T11	Incluye objetivos especiales de seguridad	1	3	3	Seguridad normal
T12	Provee acceso directo a	1	5	5	Los usuarios web del

	terceras partes				centro tienen acceso.
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	1	1	Sistema fácil de usar

Tabla 5.3 Tabla Factor de complejidad técnica (TCF)

TCF = 0.6 + 0.01 x Σ (Pesoi x Valor asignadoi)

TCF = 0.6 + 0.01 x 46

TCF = 1.06

Factor de ambiente (EF)

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del Factor de ambiente. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5.

Factor	Descripción	Peso	Valor	Total	Comentario
E1	Familiaridad con el modelo de proyecto utilizado	1.5	0	0	El grupo está regularmente familiarizado con el modelo
E2	Experiencia en la aplicación	0.5	0	0	El equipo ha trabajado poco tiempo en ésta aplicación
E3	Experiencia en orientación a objetos	1	3	3	El equipo programa en objetos
E4	Capacidad del analista líder	0.5	3	1.5	El analista tiene experiencia media.
E5	Motivación	1	3	3	El grupo está altamente motivado

E6	Estabilidad de los requerimientos	2	2	4	Se esperan pocos cambios
E7	Personal part-time	-1	1	-1	El equipo es mitad y mitad.
E8	Dificultad del lenguaje de programación	-1	1	-1	Se usará lenguaje PHP

Tabla 5.4 Tabla Factor de ambiente (EF)

EF =1.4 - 0.03 x Σ (Pesoi x Valor asignadoi)

EF =1.4 - 0.03 x 9.5

EF =1.4 - 0.285

EF = 1.115

Finalmente

UCP = UUCP x TCF x EF

UCP = 28 x 1.06 x 1.115

UCP = 33.0932

Paso 3 De los Puntos de Casos de Uso a la estimación del esfuerzo

Karner originalmente sugirió que cada Punto de Casos de Uso requiere 20 horas-hombre. Posteriormente, surgieron otros refinamientos que proponen una granularidad algo más fina, según el siguiente criterio:

- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.
- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.
- Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.
- Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre.

Esfuerzo en horas-hombre

E = UCP x CF

Donde,

E: esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: factor de conversión

E = UCP x CF

E = 33.0932 x 20

E = 926.60 Horas-Hombre

Paso 4 Cálculo del esfuerzo total

Actividad	Porcentaje	Horas-Hombre
Análisis	**10%**	**231,6524**
Diseño	**20%**	**463,3048**
Programación	**40%**	**926,6096**
Pruebas	**15%**	**347,4786**
Sobrecarga (otras actividades)	**15%**	**347,4786**
Total	**100%**	**2316,524**

Tabla 5.5 Tabla Cálculo del esfuerzo total

Si 2316,524 horas-hombre y se estima que cada mes tiene como promedio 192 horas laborables, eso daría un **ET = 12.0652292 mes-hombre**.

Esto quiere decir que 1 persona puede realizar el problema analizado en 12 meses aproximadamente.

Paso 5 Costo del Proyecto.

Salario promedio mensual $150.00

CH: Cantidad de hombres**.**

Tiempo: Tiempo total del proyecto.

CH = 2 hombres

CHM = 2 * Salario Promedio

CHM = 150.00 $/mes

Costo = CHM * ET / CH
Costo = 150.00 * **12.0652292** /2
Costo = $ 904.892
Tiempo = ET / CH
Tiempo = 12.0652292 /2
Tiempo = 6.032 meses

De los resultados obtenidos se interpreta que con 2 hombres trabajando en el proyecto el mismo se desarrolla 6 **meses** y su costo total se estima que sea **$**904.892**.**

5.3 Beneficios tangibles e intangibles.

El sistema Directorio de Búsqueda para la Universidad de Ciencias Informáticas no es un producto con fines comerciales, su principal objetivo es resolver los problemas que existen con el Directorio anterior e integrarlo a los sistemas implementados bajo una Arquitectura Orientada a Servicios, arquitectura que se encuentra en su etapa inicial actualmente en el centro.

El beneficio fundamental del sistema es contar con una aplicación web flexible, dinámica y de interfaz agradable que permita la realización de búsqueda del personal del centro de una forma eficiente y rápida, basada en SOA.

Por tanto, los beneficios inmediatos son intangibles:

1. Fácil y rápido acceso al proceso de búsqueda.
2. Óptimo funcionamiento de la aplicación.
3. Implementada bajo la nueva arquitectura concebida en el centro SOA.

5.4 Análisis de costos y beneficios.

Desarrollar un producto informático cuesta. Justificar entonces su desarrollo depende de los beneficios que reportarían su implantación y utilización [3].

Los beneficios pueden ser económicos y de orden social, estos últimos son de tanta importancia como los primeros. El sistema que se propone está dirigido a la Universidad de las Ciencias Informáticas, para la satisfacción del personal del centro, por tanto su mayor beneficio es de orden social.

Una vez implantado el sistema éste contribuirá a aumentar la eficiencia en el proceso de búsqueda de personas por un usuario determinado del centro, al disminuir el tiempo de respuesta del sistema y obtener estos resultados con mayor rapidez y certeza.
La tecnología utilizada para el desarrollo del sistema es totalmente libre, por tanto no hay que incurrir en gastos en el pago de licencias de uso. El sistema es portable por lo que un cambio de plataforma para la implantación del mismo es viable y factible, y no hay que incurrir en muchos cambios. Analizando el costo del proyecto, los numerosos beneficios que reporta, detallados con anterioridad, se puede concluir que su implementación es realmente factible.

5.5 Conclusiones.

En este capítulo se describió el estudio de factibilidad correspondiente al sistema propuesto: Directorio de Búsqueda, utilizando la técnica de estimación por Puntos de Caso de Uso, la cual resulta muy efectiva para estimar el esfuerzo requerido en el desarrollo de los primeros Casos de Uso de un sistema, si se sigue una aproximación iterativa como el Proceso Unificado de Rational. En éste tipo de aproximación, los primeros Casos de Uso a desarrollar son los que ejercitan la mayor parte de la arquitectura del software y los que a su vez ayudan a mitigar los riesgos más significativos.
Teniendo en cuenta el costo estimado y los beneficios que reportará al ser implantado dicha proposición, al contribuir a mejorar el proceso de búsqueda para satisfacción del usuario final y formar parte de los sistemas desarrollados en la nueva arquitectura adoptada por la universidad, (SOA), se llega a la conclusión de que es factible implementar la herramienta propuesta.

CONCLUSIONES

El presente trabajo finaliza dando cumplimiento al objetivo trazado de desarrollar un nuevo Directorio de Personas en software libre ,basado en la Arquitectura Orientada a Servicios , para mejorar el proceso de búsqueda de personas en la Universidad de Ciencias Informáticas.

Se efectuó un estudio de las herramientas y tecnologías de desarrollo que se utilizaron en la confección de la solución. Se definió el diseño que modela el sistema y brinda soporte a todos los requisitos funcionales y no funcionales.

Un aspecto de gran utilidad en la confección y concepción del diseño fue la comprensión y utilización de patrones, como el patrón que emplea Symfony ,patrón Modelo Vista Controlador que permitieron aplicar buenas prácticas y brindar soluciones probadas a problemas comunes.

Se estructuró un diagrama de componentes que representa la implementación del sistema en términos de componentes, quedando construida una aplicación basada en la arquitectura SOA, ajustándose a la estrategia que se desarrolla en la UCI, capaz de garantizar rapidez y efectividad en el proceso de búsqueda de personas.

RECOMENDACIONES

Seguir mejorando la apariencia de la aplicación con efectos visuales que hagan la interfaz para el usuario aún más amigable.

Agregar funcionalidades nuevas a la aplicación como un módulo de administración y un módulo de estadísticas que permita recoger datos de las búsquedas efectuadas por los usuarios.

REFERENCIA BIBLIOGRÁFICA

1. **Estrada, C. V y Mola, K.V.** *Proceso de Desarrollo basado en la Arquitectura Orientada a Servicios para el proyecyo ASP.* Ciudad de La Habana : s.n., 2007.

2. Estructura SOA Alcance SOA . [En línea] [Citado el: 20 de Marzo de 2008.] Disponible en http://www.tecnologiahechapalabra.com/datos/soluciones/negocio/articulo.asp?i=961.

3. **DE Leon, Yara y Rabi Cespedes, Michel.** *Aplicación del Plug-in SOA-RUP.* Ciudad de La Habana : s.n., 2007.

4. **Otón Tortosa, Salvador y Gutiérrez Gómez, Isaac.** Arquitecturas Orientadas a Servicios. [En línea] [Citado el: 10 de Marzo de 2008.]
Disponible en http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-132/paper09.pdf.

5. **Edo del Moral, Isabel y Fernández Lloria, Susana.** Estudio del metabuscador Kartoo. [En línea] [Citado el: 16 de Marzo de 2008.]
Disponible en http://personales.upv.es/ccarrasc/doc/2003-2004/KartOO/Estudio_KartooFINALI.

6. ORACLE IDENTITY MANAGEMENT. [En línea] [Citado el: 20 de Marzo de 2008.] Disponible en http://www.ayto-gijon.es/documentos/Departamentos%5CCoordinacion%5CJornadas.

7. Directorio X.500. [En línea] [Citado el: 20 de Marzo de 2008.]
Disponible en http://www2.uca.es/serv/ai/servicios/serv-x500.html.

8. **Gomez Reyes, Marta.** PROYECTO DIRECTORIO ELECTRONICO DE CUBA. [En línea] [Citado el: 20 de Marzo de 2008.]
Disponible en http://www.congreso-info.cu/Userfiles/File/Info/Info97/Ponencias/150.pdf.

9. **Mezquita, Yoel Ledo y Bergantiños Hidalgo, Damián.** PORTALES PARA LA INFORMACIÓN UNIVERSITARIA. [En línea] [Citado el: 25 de Marzo de 2008.]
Disponible en http://www.congreso-info.cu/UserFiles/File/Info/Info2002/Ponencias/30.pdf.

10. Metodologías de Desarrollo de Software. [En línea] [Citado el: 14 de Febrero de 2008.]
Disponible en http://www.scribd.com/doc/2050925/metodologias-de-desarrollo-software.

11. *Metodologías De Desarrollo De Software.* [En línea] [Citado el: 15 de Febrero de 2008.] Disponible en http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

12. Metodologias RUP y XP - [PROCESOS DE DESAROLLO]. [En línea] [Citado el: 13 de Mayo de 2008.] Disponible en http://jackopc.blogspot.com/2007/05/metodologias-rup-y-xp-procesos-de.html.

13. Metodología OMT. [En línea] [Citado el: 15 de Mayo de 2008.] Disponible en http://www.monografias.com/trabajos13/metomt/metomt.shtml.

14. Wikipedia.Proceso Unificado de Rational. [En línea] [Citado el: 6 de Junio de 2008.] Disponible en http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational#Demostrar_valor_.

15. **López, Pedro Enrique y Domínguez Medina, Lisdanay.** *SISTEMA PARA EL CONTROL DEL PERSONAL MOVILIZADO EN LAS FUERZAS .* Ciudad de La Habana : s.n., 2007.

16. PostgreSQL Práctico. [En línea] [Citado el: 17 de Abril de 2008.] Disponible en: http://www.sobl.org/traducciones/practical-postgres/node19.html.

17. SOA. [En línea] [Citado el: 10 de Enero de 2008.] Disponible en: http://www.google.com/search?q=cache:zcIMBj36cUsJ:md2.dei.inf.uc3m.es:8000/.

18. **Tamayo Ramos, Alberto y Zamora Aguilar, Ronny.** *Sistema para la informatización del Convenio Cuba-Venezuela. Módulo ficha mixta.* Ciudad de La Habana : s.n., 2007.

19. Symfony, la guía definitiva. [En línea] [Citado el: 6 de Mayo de 2008.] Disponible en: :http://www.librosweb.es/symfony/index.html.

20. UML y Patrones. [En línea] [Citado el: 16 de Mayo de 2008.] Disponible en: http://bibliodoc.uci.cu/pdf/reg00061.pdf.

21. SMTP. [En línea] [Citado el: 18 de Mayo de 2008.] Disponible en: http://www.programacionweb.net/articulos/articulo/?num=412.

22. DESARROLLO DEL CONTENIDO cap 4. [En línea] [Citado el: 5 de Mayo de 2008.] Disponible en: http://aragua.una.edu.ve/SITE%20COMPLETO/Inicio%20de%20CD/trabajo/Capitulo4.

23. Guía Breve de CSS. [En línea] [Citado el: 6 de Mayo de 2008.] Disponible en: http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo.

24. Lenguaje PHP. [En línea] [Citado el: 6 de Junio de 2008.] Disponible en: http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap11-2.html.

25. Programación en PHP. [En línea] [Citado el: 10 de Febrero de 2008.] Disponible en: http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_PHP.

26. **Hernandez, C. G.** *Módulo Alojamiento del Sistema Automatizado para la Gestión de Información de la Misión Milagro.* Ciudad de La Habana : s.n., 2007.

27. **Fonseca Pérez, Ricardo y Carvajal Vila, Carlos Alberto.** *Servicios Web para el Sistema de Reservación del Pase.* Ciudad de La Habana : s.n., 2007.

28. Java Script. [En línea] [Citado el: 6 de Junio de 2008.] Disponible en: http://www.webestilo.com/javascript/.

29. Curso de Macromedia Dreamweaver. [En línea] [Citado el: 6 de Junio de 2008.] Disponible en: http://www.aulafacil.com/AulaDream/Dream/temario.htm.

30. PDT: Eclipse + PHP. [En línea] [Citado el: 6 de Junio de 2008.] Disponible en: http://www.desarrolloweb.com/articulos/pdt-eclipse-php.html.

31. **Vizcaíno, Aurora, García, Felix Oscar and Caballero, Ismael.** *Una Herramienta CASE para ADOO: Visual Paradigm.* s.l. : UNIVERSIDAD DE CASTILLA-LA MANCHA.

32. Visual Paradigm. [En línea] [Citado el: 23 de Enero de 2008.] Disponible en: http://www.visual-paradigm.com/product/vpuml.

33. **GONZALO GÉNOVA, M. C. V., MÓNICA MARRERO.** *Sobre la diferencia entre análisis y diseño, y por qué es relevante para la transformación de modelos.* 2006.

34. Estimación de Líneas de Código, Puntos de Función y Esfuerzo Una visión de su naturaleza probabilística. [En línea] [Citado el: 5 de Mayo de 2008.] Disponible en: http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/PAGANINI-ESTIMACION.pdf.

35. **Peralta, Mario.** *ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO.* Instituto Tecnológico de Buenos Aires : Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS).

BIBLIOGRAFÍA

1. Estimación de Líneas de Código, Puntos de Función y Esfuerzo Una visión de su naturaleza probabilística. [En línea] [Citado el: 5 de Mayo de 2008.] Disponible en: http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/PAGANINI-ESTIMACION.pdf.

2. **Gamma, Erich, y otros.** *Design Patterns. Elements of Reusable Object Oriented Software, Adison-Wesley Longman.* 1995.

3. **Jacobson, I. and Booch, G. , Rumbaugh, J.** *El Proceso Unificado de Desarrollo de software.* 2000.

4. **Larman, Craig.** Uml y Patrones.Introducción al análisis y diseño orientado a objetos. [Online] 1999. [Cited: Enero 24, 2008.] Disponible en: http://bibliodoc.uci.cu/pdf/reg00061.pdf ISBN 970-17-0261-1.

5. Los mejores IDEs para Php. [Online] [Cited: Junio 6, 2008.] Disponible en: http://www.urlfan.com/local/los_mejores_ides_para_php/57473579.html.

6. Lenguaje PHP. [En línea] [Citado el: 6 de Junio de 2008.] Disponible en: http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap11-2.html.

7 .Metodologias RUP y XP - [PROCESOS DE DESAROLLO]. [En línea] [Citado el: 13 de Mayo de 2008.] Disponible en http://jackopc.blogspot.com/2007/05/metodologias-rup-y-xp-procesos-de.html.

8. Red Cenia Internet. [Online] [Cited: Junio 8, 2008.] Disponible en: http://www.cu.ipv6tf.org/2004entrevista_jesus.htm.

9 .Symfony, la guía definitiva. [En línea] [Citado el: 6 de Mayo de 2008.] Disponible en: :http://www.librosweb.es/symfony/index.html.

10. Visual Paradigm. Visual Paradigm for UML Model –Code-Deploy Platform. [Online] [Cited: Junio 8, 2008.] Disponible en: http://www.visual-paradigm.com/product/vpuml/.

ANEXOS

Anexo 1 Diagramas de interacción.

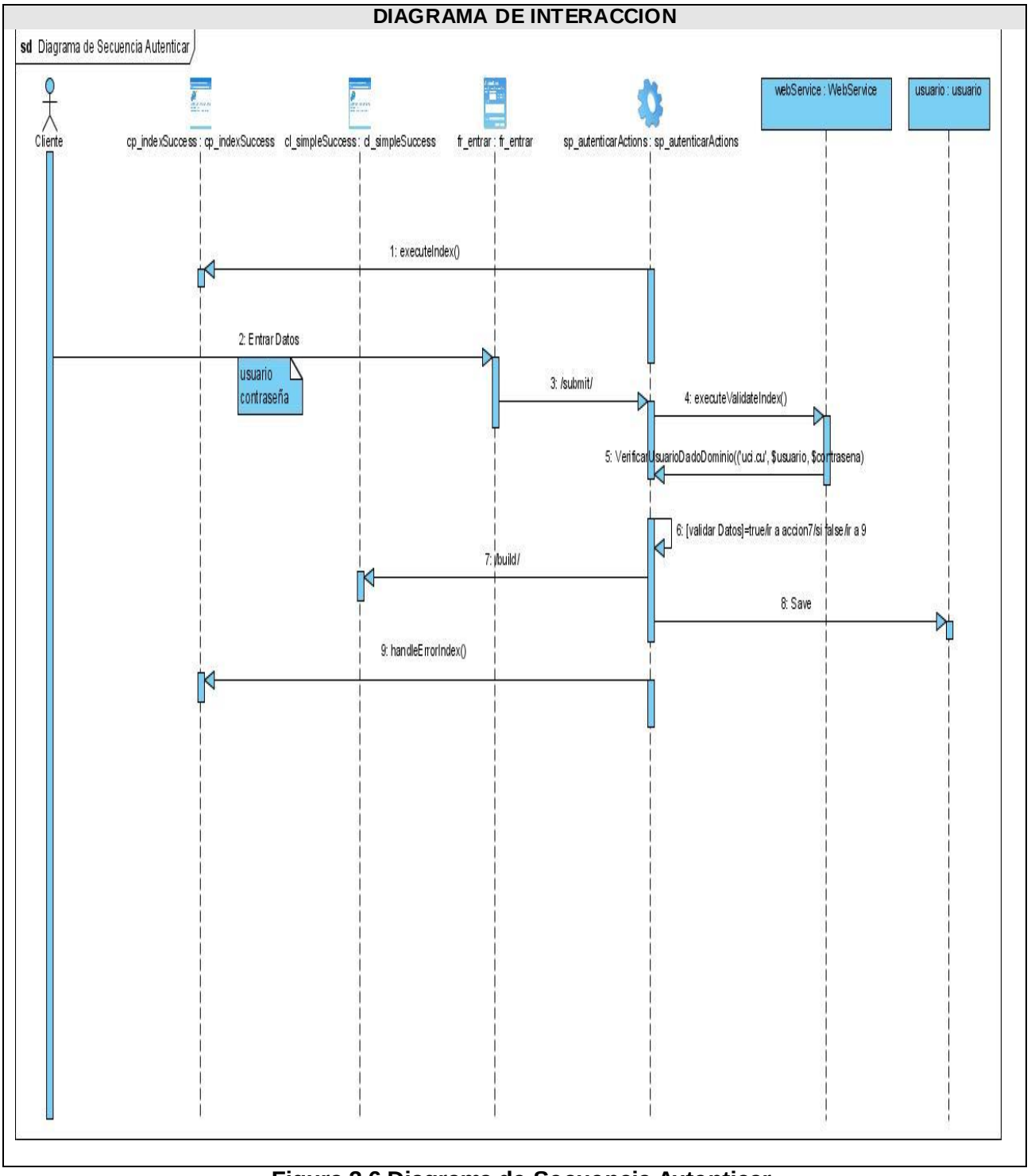

Figura 3.6 Diagrama de Secuencia Autenticar.

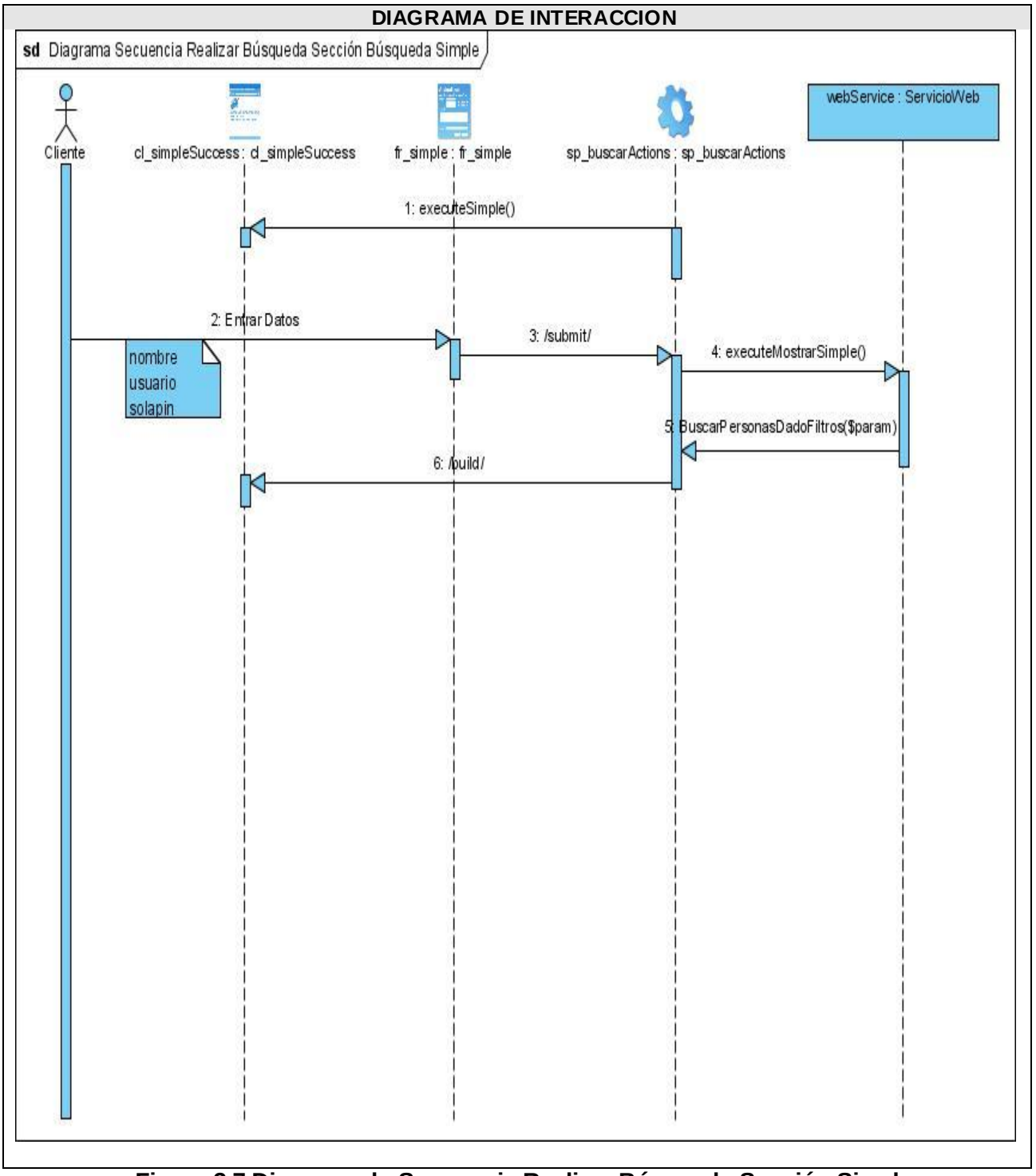

Figura 3.7 Diagrama de Secuencia Realizar Búsqueda Sección Simple.

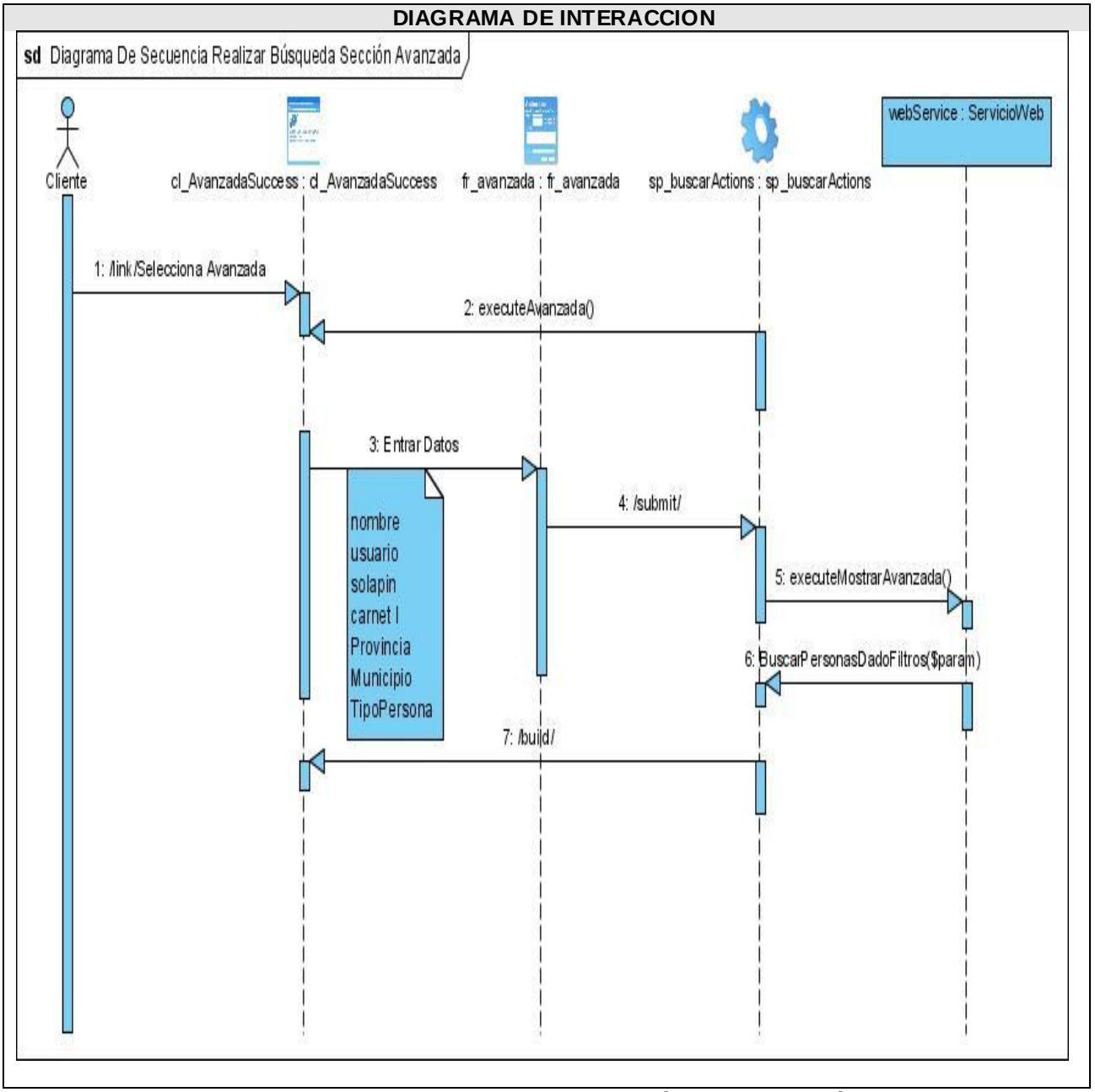

Figura 3.8 Diagrama de Secuencia Realizar Búsqueda Sección Avanzada.

DIAGRAMA DE INTERACCION

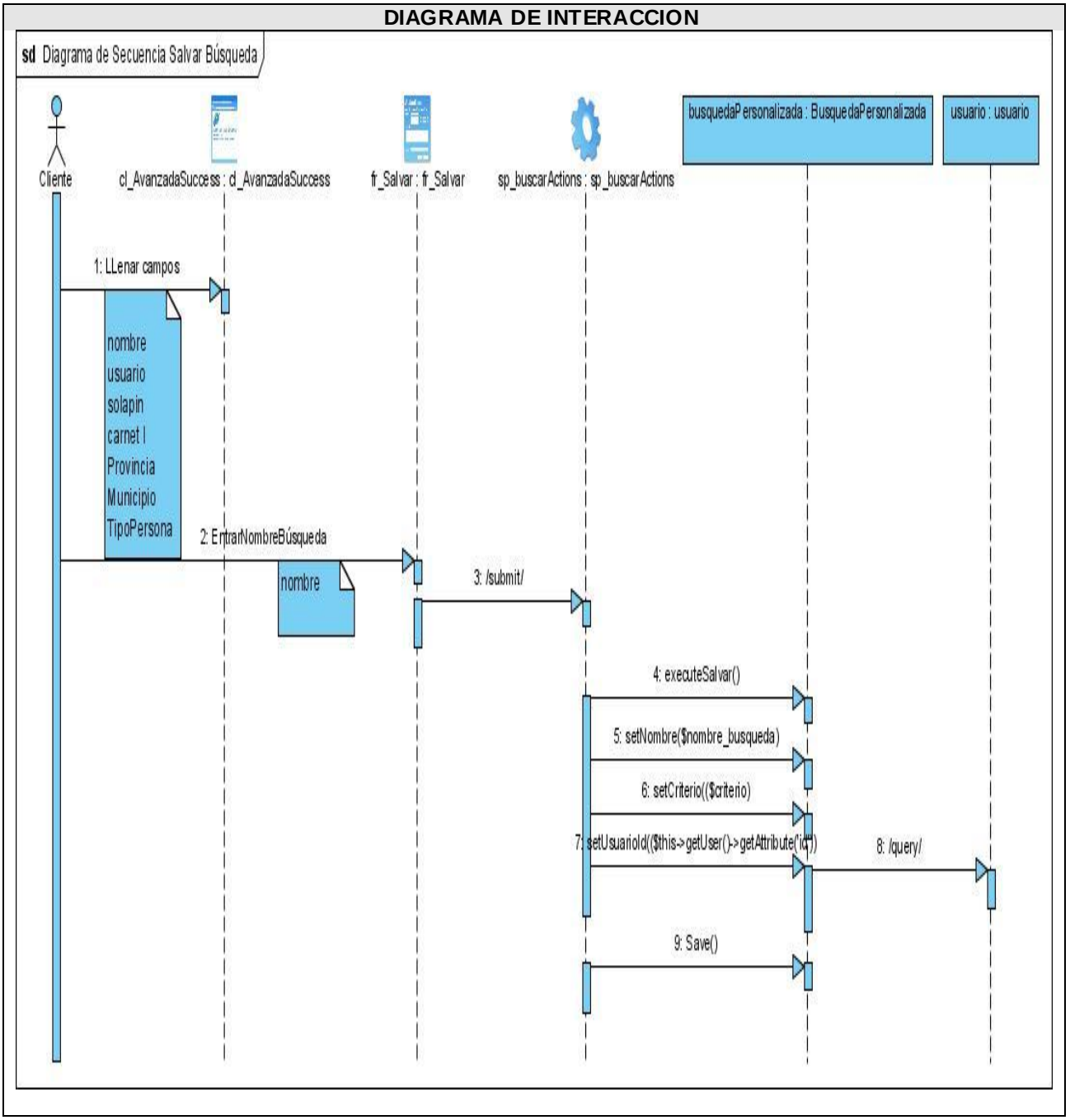

Figura 3.9 Diagrama de Secuencia Salvar Búsqueda.

DIAGRAMA DE INTERACCION

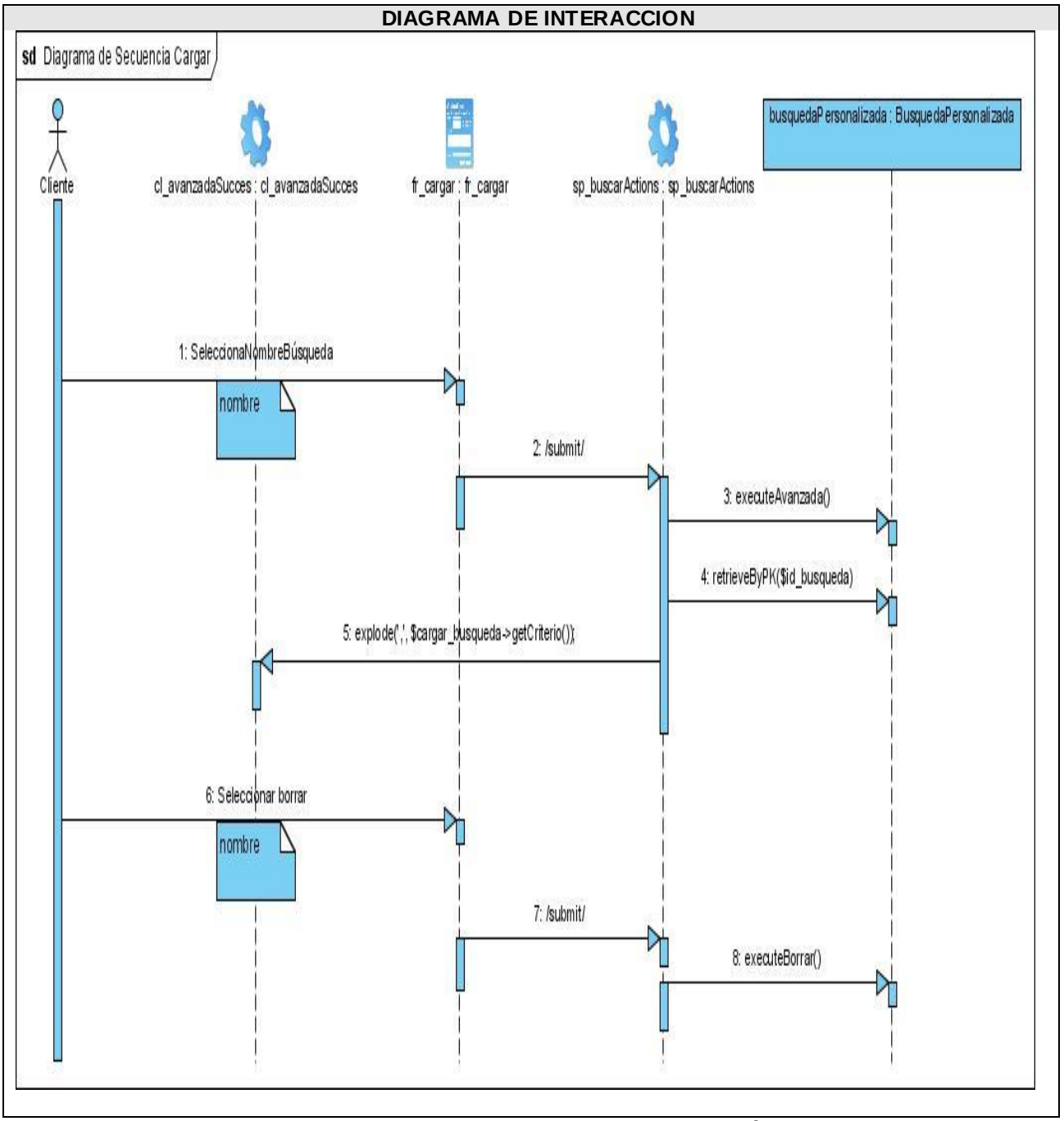

Figura 3.10 Diagrama de Secuencia Cargar Búsqueda.

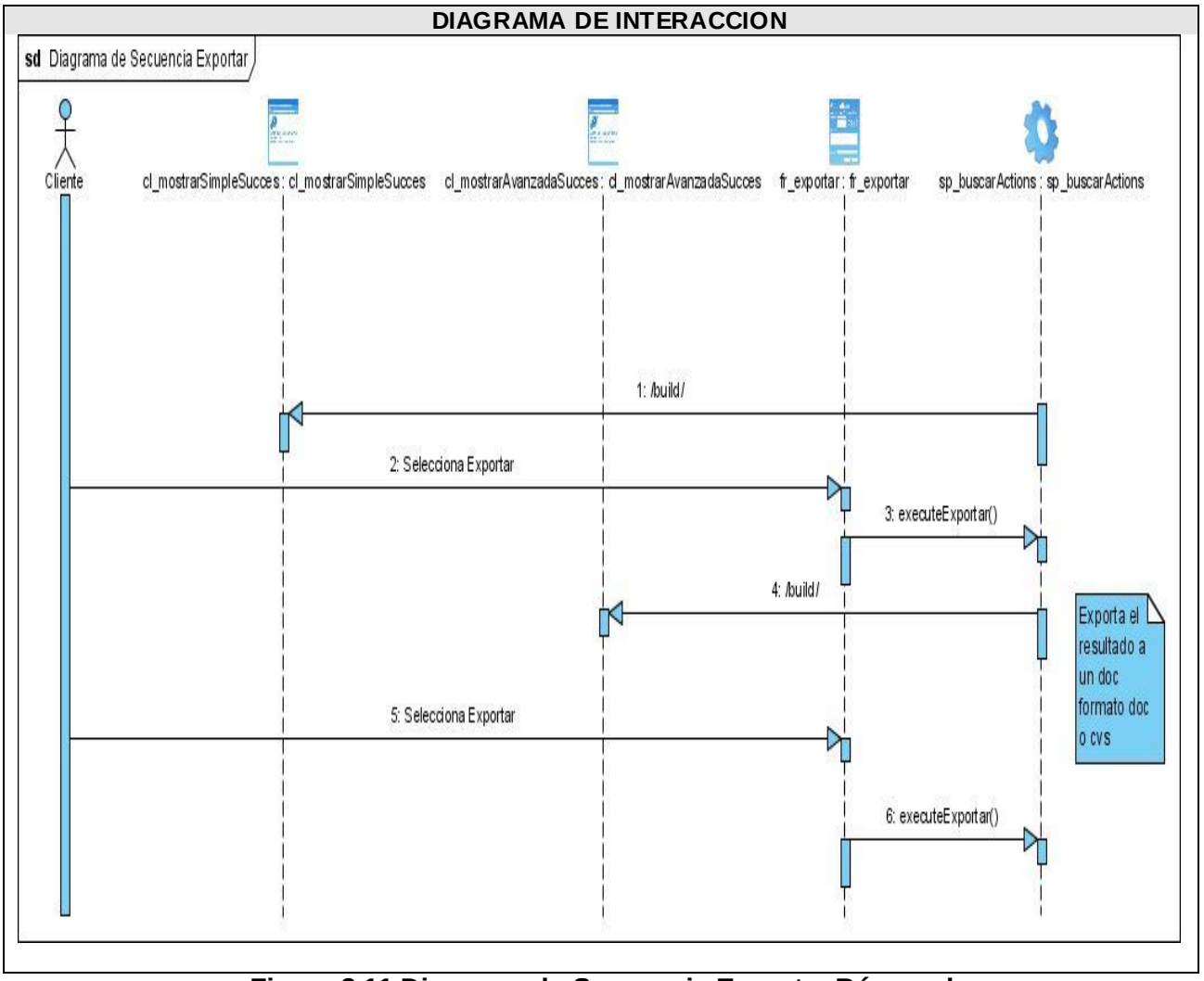

Figura 3.11 Diagrama de Secuencia Exportar Búsqueda.

Prototipo de Interfaz.

Autenticar

Búsqueda Simple

Búsqueda Avanzada


Directorio de Personas
Búsqueda Simple
Búsqueda Avanzada
Cerrar Sessión
Búsqueda
Usuario atamayor
Nombre alberto
Ci
Solapin t14439
Provincia GRANMA
municipio BAYAMO
Tipo de Persona Profesor
Buscar
Salvar Búsqueda
Nombre búsqueda1
Salvar
Cargar Búsqueda
Nombre
Cargar
Borrar
Resultado de la Búsqueda(1)
Nombre: ALBERTO TAMAYO RAMOS
Usuario: atamayor
Exportar

GLOSARIO

1. **APACHE:** es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1.
2. **Internet:** Sistema de redes de computación ligadas entre sí, con alcance mundial, que facilita servicios de comunicación de datos como registro remoto, transferencia de archivos, correo electrónico y grupos de noticias
3. MVC: Modelo Vista Controlador.
4. **PostgreSQ**L: es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre.
5. **XML**: son las siglas en ingles de eXtensible Markup Languaje (lenguaje de marcado ampliable y extensible) desarrollado por el World Wide Web Consortium (W3C).No es mas que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados. XML tiene otras aplicaciones entre las que destaca su uso como estándar para el intercambio de datos entre diversas aplicaciones o software con lenguajes privados como en el caso de SOAP.
6. .**UML:** *Unified Modeling Language*. Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.
7. **WSDL** son las siglas de *Web Services Description Language*, un formato XML que se utiliza para describir servicios Web. WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo.
8. **SOAP** (siglas de *Simple Object Access Protocol*) es un protocolo estándar creado por Microsoft y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML es uno de los protocolos utilizados en los servicios Web.
9. **REST** son las siglas en inglés de *Representational State Transfer*. Es un estilo de arquitectura y no un estándar definido. REST se basa en las URIs para identificar recursos, los cuales se retornan en XML puro.