UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

Autores: Rosayne Valenzuela Aguilera

Yuraldy Díaz García

Tutor: Ing. Yurisbel Vega Ortiz

Asesora: Lic. Yaqueline Zamora Mora

Ciudad de la Habana, junio 2007.
"Año 49 de la Revolución"

DEDICATORIA

A nuestros padres, por todo el apoyo que nos han brindado, su comprensión, su amor, su guía y a los que les debemos hoy lo que somos.

AGRADECIMIENTOS

A nuestro tutor Ing. Yurisbel Vega Ortiz, por habernos ayudado y apoyado siempre, por las horas de entrega y de enseñanza para lograr la exitosa culminación de nuestra tesis.

Al Ing. Manuel Alejandro Gil Martín, por habernos orientado y dedicado parte de su tiempo en aclarar nuestras dudas, además de facilitarnos información muy importante para el desarrollo de esta tesis.

Al Ing. Yuniel Eliades Proenza, por su gran colaboración.

Al Ing. Juenlis Coss Piña por habernos brindado una gran ayuda en un momento crucial para la culminación de nuestra tesis.

A todos los profesores que durante estos cinco años nos orientaron, nos brindaron su conocimiento para lograr nuestra correcta formación, fueron como nuestros padres en los momentos difíciles en especial a Roberto Millet Luaces y Alcides Cabrera Campos.

A toda nuestra familia por estar siempre a nuestro lado, brindarnos su amor, su apoyo, un consejo en un momento difícil, por preocuparse tanto por nosotros como nuestros padres.

A todos nuestros amigos, con los que hemos pasados momentos muy agradables, que han sabido con su cariño llenarnos de felicidad y a los que recordaremos siempre, especialmente a Aylín, Marggie, Jacqueline, Annabell, Yadeny, Leonardo, Yunesti, y a todos aquellos que se han ganado un pedacito de nuestro corazón.

A todos, los que de una forma u otra se han preocupado por nosotros.

Muchas Gracias.

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 _____.

_____ _____

Rosayne Valenzuela Aguilera

Autor

Yuraldy Díaz García

Autor

_____

Ing. Yurisbel Vega Ortiz

Tutor

DATOS DE CONTACTO

Síntesis del Tutor Ing. Yurisbel Vega Ortiz

Profesión: Ingeniero Informático.

Años de graduado: 1

Síntesis de la Asesora Lic. Yaquelin Zamora Mora

Profesión: Licenciada en Información científico técnica y Bibliotecología.

Categoría docente: Instructor.

Años de graduada: 4

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado “Organización, reconcepción y migración de la fachada de los servicios Web de la Base de Datos Ciudadano”, fue realizado en la Universidad de las Ciencias Informáticas (UCI), a solicitud del Departamento de Informatización. El mismo considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

Totalmente

Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes:

_____

_____

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reporta un efecto económico que asciende a _____.

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____

_____ _____

Nombre del representante de la entidad Cargo

_____ _____

Firma Cuño

OPINIÓN DEL TUTOR

Título: Organización, reconcepción y migración de la fachada de los servicios Web de la Base de Datos Ciudadano

Autores: Rosayne Valenzuela Aguilera y Yuraldy Díaz García

El tutor del presente Trabajo de Diploma considera que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando la escala: muy alta, alta, adecuada) entre otras las cualidades siguientes:

- Independencia
- Originalidad
- Creatividad
- Laboriosidad
- Responsabilidad

< Además, debe evaluar la calidad científico-técnica del trabajo realizado (resultados y documento) y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de <nota>. <Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>

_____ _____

Firma Fecha

RESUMEN

Existen actualmente en la Universidad de las Ciencias Informáticas (UCI) diferentes aplicaciones que automatizan determinados procesos, facilitando el desarrollo de diferentes actividades en el Centro. Estas aplicaciones hacen uso de la información almacenada en la Base de Datos Ciudadano, que contiene la información común o general de todas las personas pertenecientes a la Universidad.

Actualmente existe implementada una fachada de servicios Web sobre esta Base de Datos, pero estos servicios en su gran mayoría no responden a las necesidades de las aplicaciones, son bastantes lentos, provocando que las aplicaciones no hagan uso de los mismos, y que por tanto accedan directamente a Ciudadano.

Debido a esta situación se hace necesario lograr una reconcepción, organización y migración de esta fachada, de forma tal, que satisfaga a las distintas aplicaciones. Para ello se hace necesario llevar a cabo una investigación profunda con el objetivo de obtener qué servicios responden verdaderamente a las necesidades de estas aplicaciones y de esta forma poder concebir una fachada de servicios Web con funcionalidades objetivas.

Palabras Claves:
Servicios Web, XML, WSDL, SOAP, UDDI.

ÍNDICE DE TABLAS

ÍNDICE FIGURAS

ÍNDICE

INTRODUCCIÓN

A nivel mundial se ha alcanzado un alto desarrollo tecnológico, Cuba a pesar de ser un país subdesarrollado y a su vez bloqueado hace más de 40 años, ha sabido de alguna manera seguir de cerca este desarrollo y tratar de con sus escasos recursos irse inmiscuyendo en este mundo de la tecnología y la informática. Con este propósito fue fundada la Universidad de las Ciencias Informáticas(UCI), creada al calor de la batalla de ideas, con el objetivo de preparar jóvenes capaces de informatizar cada rincón de nuestro país y de convertirlo en una pequeña pero sólida industria del software.

Actualmente nuestra Universidad cuenta con un gran número de estudiantes, trabajadores, profesores, provenientes de diferentes lugares de nuestro país, que día a día, hacen de ella el sueño esperado por nuestro Comandante en Jefe.

La información referente a cada una de estas personas se encuentra almacenada en diferentes Bases de Datos (BD), es decir se cuenta con una BD para almacenar la información de los estudiantes (Akademos), otra para la información de los trabajadores (Asset), además (Terciarizado) que almacena tanto a los trabajadores como al capital terciarizado, que no son más que aquellas personas que trabajan en la Universidad pero que son atendidos por la empresa a la que pertenecen. Además se cuenta con una BD nombrada Ciudadano que almacena la información común o general de todas las personas, obtenida de las BD mencionadas anteriormente.

Asimismo han sido desarrolladas en nuestra Universidad numerosas aplicaciones que facilitan el trabajo en el Centro, ya sea para la reservación del pase, el acceso a los comedores, la búsqueda de información relacionada con los resultados docentes de los estudiantes, y así, se pudieran enumerar diferentes sistemas que existen actualmente, además de que estos pudieran incrementarse a medida de que se vayan informatizando más procesos en la Universidad, y la mayoría de ellos para su funcionamiento necesitan obtener información de las personas, y por tanto necesitan acceder a Ciudadano cuando la información requerida es de carácter general para cualquier tipo de persona.

Para la obtención de información de la BD Ciudadano fue implementada una fachada de servicios Web con el objetivo de que las distintas aplicaciones hicieran uso de los mismos. Estos no fueron implementados eficientemente de forma tal que respondieran a las necesidades de las aplicaciones, por lo que actualmente las mismas acceden directamente a la BD, provocando esto ciertas deficiencias en el funcionamiento de estas aplicaciones, además problemas de seguridad y limitando la independencia de estas aplicaciones respecto a la estructura de la BD.

Por esta razón el Departamento de Informatización perteneciente a la Infraestructura Productiva (IP) de la UCI, que tiene como misión dirigir, organizar, coordinar, chequear, diseñar y definir la informatización de todos los procesos internos en cada una de las áreas que rigen la vida de la Universidad, y con ello velar por el buen funcionamiento de los sistemas que automatizan estos procesos ha solicitado desarrollar algún mecanismo que solucione todas estas problemáticas que han puesto freno a su misión dentro de nuestra Universidad.

Dada la situación antes planteada surge como necesidad a resolver la siguiente interrogante:

¿Cómo organizar, reconcebir y migrar la fachada de servicios Web de la Base de Datos Ciudadano para que las aplicaciones que la utilizan se abstraigan totalmente de su estructura?

Para darle solución a este problema se ha identificado como **objeto de estudio** el *proceso de comunicación entre aplicaciones Web*, enmarcando la investigación en el *proceso de comunicación entre aplicaciones mediante* servicios *Web, el universo de aplicaciones que requieren información de la BD Ciudadano, la BD Ciudadano, y la fachada actual de servicios Web de la misma* como campo de acción.

Como idea a defender se ha planteado:

Si se logra una buena organización y reconcepción de la fachada de servicios web de la Base de Datos Ciudadano, las aplicaciones que interactúan con esta lograrán mayor eficiencia en sus funcionalidades.

Constituyendo el **objetivo principal**:

Desarrollar la fachada de servicios Web para la BD Ciudadano que brinden funcionalidades objetivas para las diferentes aplicaciones de la UCI que lo necesiten.

Identificando además los siguientes **objetivos específicos:**

- Migrar los servicios Web que actualmente estén bien implementados.
- Optimizar la implementación de los servicios Web existentes y que están en uso.
- Descartar los servicios Web que no brindan funcionalidades objetivas.
- Concebir nuevos servicios Web necesarios para las distintas aplicaciones.
- Lograr organización y eficiencia en la concepción de la fachada de servicios Web.

Para lograr el cumplimiento de estos objetivos se hace necesario desarrollar un grupo de actividades:

- Búsqueda de información sobre la comunicación entre Aplicaciones mediante servicios Web.
- Investigación de los servicios necesarios para las diferentes aplicaciones en nuestra Universidad.
- Selección de las herramientas a utilizar para la implementación de los diferentes servicios Web.
- Presentación del perfil de proyecto para su aprobación.
- Elaborar el análisis y diseño.
- Desarrollar los servicios Web fundamentales para estas aplicaciones.

Como **métodos de investigación científica** se utilizaron:

Métodos Teóricos:

- Histórico lógico: Posibilitó el análisis histórico del proceso de comunicación entre las aplicaciones.
- Análisis y síntesis: Se analiza la bibliografía y se realiza síntesis de la misma.
- Modelación: Se modelan diagramas para la implementación de la solución propuesta.

Métodos Empíricos:

- Entrevistas: Se realizan entrevistas con el fin de precisar el problema a resolver, para obtener información de los servicios que necesitan las diferentes aplicaciones.
- Análisis de documentos: Se basa en la revisión de documentos utilizados en la investigación.

El cumplimiento de los objetivos planteados y con ello el desarrollo de la fachada de estos servicios Web permitirán que las distintas aplicaciones funcionen correctamente, logrando una menor dependencia entre ellas y con la BD, por tanto las diferentes actividades llevadas a cabo en nuestra Universidad que dependen del buen funcionamiento de las mismas lograrán ser realizadas con mayor facilidad, obteniéndose resultados óptimos con un tiempo de respuesta aceptable.

Asimismo se logrará dar cumplimiento a la misión del Departamento de Informatización, que tiene entre sus principales objetivos lograr automatizar todos los procesos en nuestra Universidad, integrarlos, lograr un buen funcionamiento de los mismos, haciendo un correcto uso de los recursos, tecnologías, servicios informáticos con los que cuenta nuestro Centro y de esta forma convertir nuestra institución en el prototipo para la Informatización de la Sociedad Cubana.

El presente trabajo se estructura en cuatro capítulos:

El primer capítulo describe el objeto de estudio, hace una valoración de los principales conceptos relacionados con los servicios Web, las tecnologías utilizadas para el desarrollo de los mismos. Además se fundamenta el uso de las herramientas, lenguajes y metodologías a utilizar.

El segundo capítulo describe los principales procesos involucrados en el objeto de estudio. Se definen los conceptos en los que se enmarca nuestra aplicación en un Modelo del Dominio, se enumeran los requisitos funcionales y no funcionales que debe tener la aplicación Web que se propone. Además incluye las descripciones de los actores y casos de uso del sistema, así como los diagramas que ilustran esta información.

El tercer capítulo plantea todas las líneas de descripción del diseño y construcción de la solución propuesta, basadas en el futuro funcionamiento del sistema a través de la representación de diagramas de clases del diseño, de componentes y despliegue. También se propone un caso de prueba para la aplicación.

El cuarto y último capítulo describe la estimación de costos del sistema propuesto y sus beneficios, basada en las técnicas de Análisis de Puntos de Casos de Uso.

Capítulo 1

Fundamentación Teórica

1.1 Introducción

En este capítulo se brinda una descripción de los principales conceptos relacionados con el objeto de estudio, indagaciones que ratifican a los servicios Web (SW) como la mejor opción cuando se habla de comunicación entre aplicaciones. Asimismo se hace un breve estudio de las principales tecnologías a utilizar para el desarrollo de SW, que las convierten en las más adecuadas dada las potencialidades que reportan.

1.2 Conceptos relacionados con el dominio del problema

12.1 Surgimiento de los servicios Web

A comienzos de este siglo se evidencia con más intensidad la necesidad de permitir a las computadoras comunicarse entre ellas e intercambiar información. Desde entonces se ha estado pensando y dando forma al nuevo modelo de computación distribuida llamado SW basados en XML.

Los SW surgieron ante una necesidad de estandarizar la comunicación entre distintas plataformas y lenguajes de programación. Anteriormente se habían realizado intentos de crear estándares pero fracasaron o no tuvieron el suficiente éxito, entre los que se encuentran DCOM (Distributed Component Object Model) de Microsoft, RMI (Remote Method Invocation) de SUN y CORBA de OMG (Object Management Group). Un gran problema es que se hacía uso de RPC (Remote Procedure Call) para realizar la comunicación entre diferentes nodos, esto además de presentar ciertos problemas de seguridad, tiene la desventaja de que su implementación en un ambiente como Internet, es casi imposible (muchos firewalls bloquean este tipo de mensajes, lo que hace prácticamente imposible a dos computadoras conectadas por Internet comunicarse). Los SW surgieron entonces para finalmente poder lograr la tan esperada comunicación entre diferentes plataformas (1).

Los SW constituyen hoy en día la arcilla que sirve de base, y como tal juega un papel muy importante en la transición al proceso distribuido en Internet. El auge de los estándares abiertos y la necesidad de comunicación y colaboración entre las personas y aplicaciones han creado un entorno donde los SW se están convirtiendo en la plataforma para la integración de aplicaciones. Estas se construyen utilizando múltiples SW distribuidos en diferentes fuentes pero que pueden ser invocados e integrados para que trabajen conjuntamente con independencia de dónde residen o cómo hayan sido implementados (2).

1.2.2 ¿Qué se entiende por servicio Web?

Un SW es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en diferentes lenguajes de programación y ejecutadas sobre cualquier plataforma pueden utilizar los SW para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos (2).

Los SW realizan funciones, las cuales van desde simples peticiones hasta procesos de negocios complicados. Una vez que un SW es publicado, otra aplicación u otro SW pueden localizarlo e invocarlo. Un SW también puede ser visto como una aplicación que existe en un ambiente distribuido, como Internet, que acepta una petición, realiza su función de acuerdo a la petición y devuelve una respuesta. No es necesario que el proveedor y el usuario del SW tengan en cuenta los detalles de la implementación, y acceso al servicio, ya que se basan en estándares como XML, HTTP y SMTP y siguen una estructura común para su descripción e invocación.

Con los SW se logra que las aplicaciones puedan compartir información, además que puedan invocar a otras funciones brindadas por diferentes aplicaciones, sin importar cómo y dónde estas hayan sido creadas, en qué lenguaje de programación, en qué plataforma se ejecutan, cuáles son los dispositivos utilizados para obtener acceso a ellas, etc. Un aspecto importante a destacar en los SW es su capacidad de agregación o composición con otros SW para proveer un conjunto de mayores características, funcionalidades más complejas que a la vez se hacen más sencillas gracias a la capacidad de estos de poder integrarse.

1.2.3 Ventajas de los servicios Web

Sus principales beneficios radican en su profunda estandarización que los vuelve independientes de la plataforma, su disponibilidad abierta al público y la facilidad y uniformidad de acceso.
Se enumeran aquí un conjunto de ventajas que resumen las superioridades que hacen de los SW en la actualidad la tecnología más importante cuando se habla de comunicación e interacción en la red.

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los SW fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar.
- Los SW se basan en HTTP sobre TCP en el puerto 80. Dado que las organizaciones protegen sus redes mediante firewalls que filtran y bloquean gran parte del tráfico de Internet, cierran casi todos los puertos TCP salvo el 80, que es precisamente, el que usan los navegadores. Los SW se vinculan por este puerto, por la simple razón de que no resultan bloqueados, además al apoyarse en HTTP, los SW pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Pueden aportar gran independencia entre la aplicación que usa el SW y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más acusada.
- Soporte para múltiples Lenguajes**:** Usted puede escribir SW en cualquier lenguaje (3).

1.2.4 Arquitectura de los servicios Web

Son muchos los esfuerzos que se han hecho y se están realizando en la actualidad para definir las diferentes especificaciones y arquitecturas que posibiliten la expansión de este nuevo tipo de aplicaciones denominadas SW. Grandes empresas como Microsoft, IBM, Intel, y consorcios como W3C están profundamente involucradas en esta investigación.

La arquitectura de SW se puede estudiar examinando sus componentes y examinando la arquitectura de capas de los protocolos. Examinando los componentes de un SW se pueden encontrar los siguientes elementos:

- **Servicio:** La aplicación es ofrecida para ser utilizada por solicitantes que completan los requisitos especificados por el proveedor de servicios. La implementación se realiza sobre una plataforma accesible en la red. El servicio se describe a través de un lenguaje de descripción de servicio. Tanto la descripción como las políticas de uso han sido publicadas de antemano en un registro.
- **Proveedor de Servicio:** Desde el punto de vista de la arquitectura, es la plataforma que provee el servicio.
- **Registro de Servicios.** Es un depósito de descripciones de servicios que puede ser consultado, donde los proveedores de servicios publican sus servicios y los solicitantes encuentran los servicios y detalles para utilizar dichos servicios.
- **Solicitante de servicios**: Desde el punto de vista de la arquitectura, la aplicación o cliente que busca e invoca un servicio (1).

Figura 1: Relaciones operacionales entre los componentes

En la arquitectura también puede aparecer otro componente denominado en algunos medios “broker”, el cual provee al solicitante de servicios de un mecanismo de descubrimiento para encontrar el servicio requerido y al mismo tiempo, también provee a los proveedores de servicios de un medio de publicitar y gestionar las descripciones sobre los servicios ofertados.

Otra forma de ver la arquitectura de un SW es examinando la pila de protocolos. Esta pila está todavía evolucionando, pero actualmente posee cinco capas principales. La figura 2 muestra esta pila de tecnologías y el actual estándar elegido, correspondiente para cada una de estas capas.

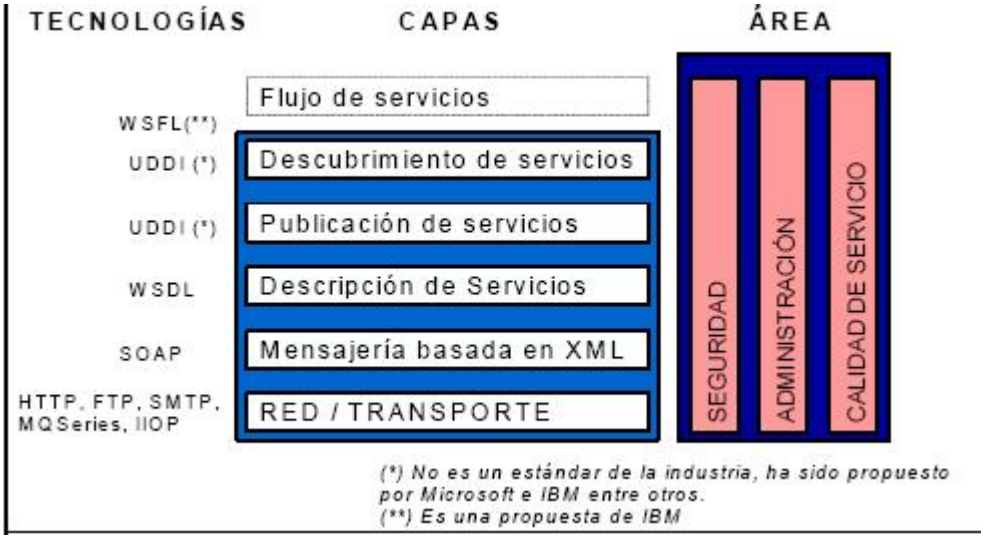

Figura 2: Pila de protocolos de la arquitectura de servicios Web

1.2.5 Tecnologías y Estándares de servicios Web

La necesidad de comunicación y colaboración entre las aplicaciones ha dado lugar a que sean muchas las empresas enfrascadas en el desarrollo y especificación de estándares con características relevantes que satisfagan la necesidad de hacer que los SW sean interoperables, distribuibles, integrables, constituyendo estos la plataforma para la integración de aplicaciones.

Con este propósito, grandes de la industria de las Telecomunicaciones y la Informática como Microsoft, IBM, Sun se han visto forzados a desarrollar nuevas técnicas que dieran respuestas a estos retos, siendo los estándares que más se destacan con esta finalidad los siguientes:

XML (Extensible Markup Language): Lenguaje de definición de documentos que permite una forma estándar de representar datos para su fácil intercambio. Es un subconjunto simplificado del SGML el cual fue diseñado principalmente para documentos Web. Deja a los diseñadores crear sus propias “etiquetas” o "tags", habilitando la definición, transmisión, validación, y la interpretación de datos entre aplicaciones y entre organizaciones. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

- Comunicación de datos. Si la información se transfiere en XML cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.
- Migración de datos. Si trabajamos en formato XML sería muy sencillo mover datos de una Base de Datos (BD) a otra (4).

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible.

SOAP (Simple Object Access Protocol**):** Es un protocolo de comunicación, el cual permite la comunicación entre aplicaciones a través de mensajes por medio de Internet. Es independiente de la plataforma, y del lenguaje. Está basado en XML y es la base principal de los SW, los mensajes SOAP son documento XML propiamente dicho.

Se enumeran aquí sus principales ventajas:

- Protocolo abierto: SOAP es una especificación abierta, construido sobre tecnologías también abiertas como XML y HTTP.
- Simplicidad: La especificación de SOAP está bien definida y es sumamente simple.
- Independiente de plataformas y lenguajes: Hoy en día HTTP es un protocolo de transporte aceptado por la industria y es usado en forma general sobre todas las plataformas. XML está siguiendo un camino parecido, dado que actualmente existen versiones de XML para casi cualquier plataforma o lenguaje de uso común. El uso de XML y HTTP hace a SOAP completamente independiente de plataformas, sistemas operativos o lenguajes de programación.
- Interoperabilidad: SOAP enfoca la interoperabilidad entre los sistemas distribuidos en el nivel de serialización de datos, introduciendo XML en reemplazo de los formatos propietarios de serialización, lo cual permite adoptar una posición neutral entre las distintas plataformas.
- Firewalls: Al usar HTTP como transporte puede pasar fácilmente a través de los firewalls, los cuales dejan pasar habitualmente el tráfico que les llega por el puerto HTTP.

- Transporte: La versión 1.0 de la especificación de SOAP mandataba el uso de HTTP como transporte, en la versión 1.1 de la especificación se flexibilizó esta posición permitiendo el uso de protocolos de transporte como FTP, SMTP, además de HTTP y sus variantes (HTTPS).

  Puede pensarse que en algún momento se usen otros protocolos como ser IIOP, RMI, etc. y de esta manera aumentar su usabilidad e interoperabilidad.

- Bajo acoplamiento: Los sistemas distribuidos basados en SOAP tienen un bajo acoplamiento, por lo cual pueden ser fácilmente mantenidos dado que pueden ser modificados independientemente de otros sistemas (5).

WSDL (Web Services Description Language**):** Es un protocolo basado en XML que describe los accesos al SW. Se puede decir que es el manual de operación del SW, porque indica cuáles son las interfaces que provee el SW y los tipos de datos necesarios para la utilización del mismo, la forma de accederlos e interactuar con ellos.

UDDI (Universal Description, Discovery and Integration**):** Es un modelo de directorios para SW. Es una especificación para mantener directorios estandarizados de información acerca de los SW, sus capacidades, ubicación, y requerimientos, en un formato reconocido universalmente. Utiliza WSDL para describir las interfaces de los SW.

1.2.6 Seguridad en servicios Web

Los SW facilitan la comunicación e interacción entre diferentes aplicaciones que necesitan intercambiar información, por lo que hoy en día se han convertido en la herramienta principal para lograr tales objetivos. Estos al estar basados en estándares lo hacen accesibles para todo el mundo, además de que son publicados para esos fines, permitir que puedan ser utilizados por todas las aplicaciones que lo necesiten. Esto reporta una gran ventaja ya que se logra interoperabilidad entre las distintas aplicaciones, pero a su vez, al estar disponibles para todo el mundo, se plantean nuevos retos de seguridad, temas en lo que se encuentran enfrascados los grandes desarrolladores, con el objetivo de eliminar la incertidumbre que causa en muchos este aspecto.

Son muchas las preguntas que se hacen los desarrolladores en cuanto a este tema de la seguridad de los SW, y que hacen que muchos duden en si emplearlos o no, preguntas como: ¿Cómo proteger adecuadamente esos servicios?, ¿De qué forma se verán amenazados el servidor y la red interna que ofrecen los servicios?, ¿Sirven de algo las tecnologías defensivas tradicionales basadas en cortafuegos y detección de intrusos?

Estas y otras muchas interrogantes parecidas provocan la incertidumbre de muchas compañías a la hora de decidirse a ofrecer SW. Los desarrolladores se enfrentan a una serie de desafíos complejos de abordar:

- Asegurar la autenticación mutua entre el consumidor que accede a los SW y el proveedor de dichos servicios.
- Permitir la autorización del acceso a recursos y, más importante, a operaciones y procesos en un entorno en el que debe administrarse y controlarse el acceso de clientes, proveedores, vendedores, competidores y hackers.
- Ofrecer la posibilidad de que un consumidor se identifique una sola vez y pueda acceder a servicios en diversos sistemas, sin tener que identificarse nuevamente en cada uno de ellos. Se está trabajando activamente en el estándar SAML para la codificación de información de autenticación y autorización en formato XML.
- Garantizar la confidencialidad de los datos intercambiados, ya que SOAP no cifra la información, la cual viaja en claro a través de Internet. El estándar ya firmemente establecido de creación de canales seguros SSL y el cifrado de partes específicas de documentos mediante el cifrado XML son las direcciones que se están siguiendo en este terreno.
- Garantizar la integridad de los datos, protegiéndolos frente a alteraciones fortuitas o deliberadas. En este campo se está utilizando el estándar de firmas XMLDSIG, que permite la firma de partes específicas del documento XML.
- Impedir el repudio de las operaciones, para lo cual nuevamente se necesitan las firmas XML.

- Protegerse frente a todo tipo de ataques. Los cortafuegos de aplicaciones XML son capaces de comprender los SW, las peticiones de servicio y el contenido de los mensajes XML intercambiados. Su uso en combinación con los cortafuegos tradicionales puede mejorar sustancialmente la seguridad de los SW ofrecidos (6).

Otros estándares aún no mencionados, también asociados a la seguridad, en los que diversos grupos de fabricantes están trabajando son XKMS, para la gestión de firmas, certificados digitales y claves públicas; XACML, para la definición de reglas y políticas de control de acceso a la información; y las especificaciones WS-Security, que representan una colección de todos los estándares XML anteriores, cubriendo casi todos los aspectos de la seguridad en las transacciones de SW.

1.3 Empleo de los servicios Web por las empresas para el desarrollo de software

Actualmente, los SW están siendo ampliamente aceptados por las empresas para el desarrollo de software de uso interno. De este modo, los servicios pueden implementar toda su funcionalidad y permanecer seguros tras el cortafuegos de la compañía.

Las principales compañías del mundo han empezado a desarrollar soluciones mediante la tecnología de los SW, algunos ejemplos son:

- **Microsoft:** Recientemente ha anunciado la disponibilidad de su primer SW, llamado MapPoint .Net. Mediante este servicio, el usuario podrá conocer su localización exacta y otros datos adicionales relacionados con su posición actual, como información de tráfico, rutas posibles o puntos comerciales cercanos.
- **IBM:** Ha implementado una solución basada en SW llamada e-Business on Demand. Esta solución permite la construcción de Extranets que ayuden a las empresas a ver los catálogos de productos, realizar y localizar pedidos o chequear el estado del inventario en tiempo real.
- **Líneas Aéreas Escandinavas:** Estas líneas aéreas han desarrollado un SW que permite a los usuarios comprar billetes y chequear el estado de los vuelos, mediante el uso del teléfono móvil (7).

1.4 Problema y situación problemática

Actualmente en nuestra Universidad se cuenta además de las tres BD: Akademos, Asset y Terciarizado, que almacenan información de estudiantes, trabajadores y profesores, y personal terciarizado respectivamente, con otra BD Ciudadano que posee una copia de la información común y general de las BD mencionadas anteriormente, la cual se actualiza dado los cambios que sufran cada una de ellas de forma independiente.

La BD Ciudadano se creó debido a que en nuestra Universidad no se cuenta con un sistema inteligente que sepa dirigirse por sí solo a una BD en específico dado una solicitud de información, es decir, cuando se necesita obtener información de un trabajador, un estudiante, o un profesor, no se cuenta con un sistema que sepa dirigirse por sí mismo a la BD donde se encuentra almacenada la información que necesita.

Existen también en nuestra Universidad diferentes sistemas: Identificación, Control de Acceso, Comedores, Akademos, Asset, entre otros, que para su funcionamiento necesitan acceder a la información almacenada en la BD Ciudadano. Para este fin fueron implementados un gran número de SW conformando la fachada actual de SW de la BD Ciudadano, pero estos en su gran mayoría no tienen funcionalidades objetivas, además servicios muy lentos que no responden a las necesidades de las aplicaciones de poder brindar una respuesta inmediata o aceptable, además algunos de ellos son implementados en .Net, por lo que el tipo de dato de la información de respuesta es específica a esta plataforma y por tanto muchas de estas aplicaciones no pueden o no saben como interpretarla. Debido a esto, las distintas aplicaciones no hacen uso de los servicios que actualmente se encuentran implementados y por tanto el acceso a la BD lo realizan directamente, no siendo estas aplicaciones dueñas de la BD y que por tanto deberían abstraerse de la estructura e implementación de la misma.

Esta dependencia de las diferentes aplicaciones con la BD no resulta conveniente, pues al estas conocer la estructura interna de la misma cuando se haga necesario algún cambio en el diseño de la BD Ciudadano, todas estas aplicaciones deben sufrir cambios en su implementación, y más aún, teniendo en

cuenta que el número de estas puede ir en aumento a medida que se vayan informatizando más procesos dentro de nuestra Universidad.

Asimismo estas aplicaciones pudieran alterar la información de la BD, situación que resultaría muy lamentable, pues de esta dependen otras aplicaciones que pudieran en un momento determinado dar respuestas inadecuadas provocadas por estos cambios, es decir esta dependencia pudiera afectar la integridad de los datos así como su confidencialidad, además el adecuado funcionamiento de los diferentes sistemas.

Por tal motivo, y para lograr una mejor comunicación entre las aplicaciones, que actualmente no brindan el funcionamiento adecuado, así como lograr abstraer a las mismas de la estructura de la BD Ciudadano, se ha propuesto llevar a cabo una reconcepción, organización y migración de la fachada de los SW de la BD Ciudadano, constituyendo este el objetivo principal de este trabajo investigativo.

1.5 Análisis de los servicios implementados que actualmente conforman la fachada de los servicios Web de la Base de Datos Ciudadano.

Actualmente existe un gran número de servicios que no se utilizan. La causa fundamental de esto es que cuando se desarrollaron no se tuvo en cuenta las necesidades de los demás proyectos de la UCI. Estos servicios no cumplen con un estándar de nombre y los tipos de datos devueltos son específicos a la plataforma .NET lo que imposibilita su reutilización desde otros lenguajes, además se definieron con un mecanismo de autenticación propio de Windows, lo que hace imposible que sean usados desde Linux. Actualmente no se conocen detalladamente las aplicaciones que usan estos servicios. La comunicación entre aplicaciones se hace a nivel de BD lo que trae consigo que exista un alto nivel de dependencia entre las aplicaciones y se violen ciertos parámetros de seguridad y confidencialidad de la información. De todo esto se infiere que es inminente un rediseño de los SW para que exista una debida comunicación entre los sistemas de la Universidad.

1.6 Análisis de los servicios necesarios para las diferentes aplicaciones

Se hizo necesario realizar un profundo estudio de cuáles eran los servicios que necesitaban cada una de las aplicaciones desarrolladas en la Universidad, para ello se empleó la entrevista como método de la investigación (Anexo 1).

El objetivo era conocer en detalle qué servicios necesitaban cada una de las aplicaciones que sí respondieran a las funcionalidades de las mismas. Con la aplicación exitosa de este método de la investigación se conocieron entonces, dadas las necesidades reales de las aplicaciones, qué servicios proponían que sí respondieran a sus intereses (Anexo 2) y los problemas que presentan los servicios que actualmente se encuentran implementados que los hacían ineficientes para las mismas.

1.7 Reconcepción de la fachada de los servicios Web de la Base de Datos Ciudadano

Dada la información que se obtuvo de los servicios que propusieran cada una de las aplicaciones de forma independiente, que sí respondían a sus necesidades, se realizó un análisis de cuáles eran los más solicitados, qué información era la que más se requería de una persona, cuáles eran los servicios verdaderamente objetivos, de forma tal que se lograran unificar todas las peticiones de las diferentes aplicaciones en determinados servicios que de forma general respondieran a las necesidades de todos los sistemas. De esta forma se logró una reconcepción de la fachada de los SW, definiendo servicios que satisfacieran a todas las aplicaciones (Anexo 3).

1.8 Estudio de los nomencladores globales reglamentados en el país

Los nomencladores globales son conceptos que no pertenecen a un sistema en específico, sino que son de uso común para varios sistemas, por lo que sus nombres e identificadores deben estar reglamentados centralmente. Debido a que el objetivo de este trabajo investigativo es implementar una fachada de SW sobre la BD Ciudadano, servicios que serán solicitados por disímiles aplicaciones, se hizo necesario investigar los nombres e identificadores de determinados conceptos tales como: Provincia, Municipio, Raza, Color de Ojos, de forma tal que los identificadores definidos para los mismos fueran reconocidos por todas las aplicaciones que hacen uso de esta información. Para ello se llevó a cabo una investigación

en la Oficina Nacional de Estadísticas, de cuáles eran los identificadores y nombres reglamentados en el país para estos conceptos (Anexo 4).

1.9 Propuesta de sistema

Para darle cumplimiento a los objetivos propuestos, se plantea desarrollar una aplicación Web que implemente los servicios concebidos para la nueva fachada de servicios Web de la BD Ciudadano, bajo las especificaciones del patrón arquitectónico Capas, específicamente en tres capas, aunque se deje para futuras versiones si llegara a ser necesario la implementación de la capa de presentación, es decir, solo se implementará en esta aplicación la capa de lógica del negocio y la capa de acceso a datos. Además la aplicación brindará sus funcionalidades mediante servicios Web, por lo que se sigue además el estilo de arquitectura SOA (Arquitectura Orientada a Servicios).

1.10 Tecnologías de desarrollo

Siguiendo los valores éticos en los que se sustenta nuestra sociedad, para la cual trabajamos; realizando un análisis del desarrollo de las tecnologías, el impacto del software libre en los desarrollos nacionales y en particular en los países subdesarrollados y cumplimentando un acuerdo del Consejo de Estado, la dirección de Informatización está inmersa en la adopción de la filosofía del Software Libre para el desarrollo del macro proyecto de Informatización de la UCI.

Dada esta estrategia que se ha planteado el Departamento de Informatización se han escogido las tecnologías a utilizar de forma tal que cumplan estos requerimientos, además de que posean características que hagan de ellas las mejores opciones para el desarrollo de SW.

- **Lenguaje de programación**

PHP ofrece una solución sencilla y universal para las paginaciones dinámicas de la Web de fácil programación. Debido a su amplia distribución PHP esta perfectamente soportado por una gran comunidad de desarrolladores. Como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparen rápidamente.

El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

Una de sus características más potentes es su soporte para gran cantidad de BD. Entre ellas pueden mencionarse InterBase, MS SQL, MySQL, Oracle, PosgreSQL, entre otras (8).

PHP es multiplataforma, pudiendo ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X. PHP soporta la mayoría de servidores Web de hoy día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape, y muchos otros.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros.

PHP tiene unas características muy útiles para el procesamiento de texto, desde expresiones regulares POSIX extendidas o tipo Perl hasta procesadores de documentos XML. Para procesar y acceder a documentos XML, soportando los estándares SAX y DOM. Puede utilizar la extensión XSLT para transformar documentos XML (9).

PHP destaca su rapidez. Este generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.

Se resumen aquí sus principales ventajas:

- Rapidez de ejecución.
- Es un lenguaje específicamente diseñado para realizar aplicaciones Web, mientras que otros lenguajes son adaptaciones de lenguajes preexistentes, no pensados para la Web.
- Mantiene un bajo consumo de recursos de máquina.
- Gran seguridad, muy poca probabilidad de corromper los datos.

- Es un lenguaje libre, lo que implica menos costes y servidores más baratos que otras alternativas.
- Es multiplataforma.
- Excelente soporte de acceso a BD.
- Presenta una filosofía totalmente diferente y, con un espíritu más generoso, es progresivamente construido por colaboradores desinteresados que implementan nuevas funciones en nuevas versiones del lenguaje.
- Combina excelentemente con otras inmejorables herramientas, como son el servidor apache y la BD mysql (o msql, o postgres), todas ellas gratuitas.
- Rico en funciones predifinas (10).

- **Servidor Web**

Apache es uno de los servidores de código abierto más popularmente utilizados, este se encarga de resolver las peticiones de páginas de Internet de los clientes utilizando el protocolo de Internet HTTP. Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad. Apache ha demostrado ser substancialmente mas rápido que muchos otros servidores libres y compite de cerca con los mejores servidores comerciales. Funciona sobre muchas plataformas.

Se pueden destacar las siguientes mejoras de la versión 2.0 que es la propuesta en este trabajo.

- Nuevo sistema de configuración y compilación: El sistema de configuración y compilación ha sido escrito de nuevo desde cero para basarlo en autoconf y libtool. Esto hace que el sistema de configuración de Apache se parezca ahora más al de otros proyectos de código libre.
- Soporte Multiprotocolo: La nueva versión tiene la infraestructura necesaria para servir distintos protocolos.
- Soporte mejorado para las plataformas que no son tipo Unix: La versión 2.0 de Apache es más rápida y más estable en sistemas que no son tipo Unix, tales como BeOS, OS/2 y Windows, que la versión antigua. Con la introducción de módulos de multiprocesamiento (MPMs) específicos para

cada plataforma y del Apache Portable Runtime (APR), estas plataformas tienen ahora implementada su propia API nativa, evitando las capas de emulación POSIX que provocan problemas y un bajo rendimiento.

- Filtros: Los módulos de Apache pueden ahora escribirse para que se comporten como filtros que actúan sobre el flujo de contenidos tal y como salen del servidor o tal y como son recibidos por el servidor.
- Mensajes de error en diferentes idiomas: Los mensajes de error que se envían a los navegadores están ahora disponibles en diferentes idiomas, usando documentos SSI.
- Configuración simplificada: Muchas directivas que podían inducir a confusión han sido simplificadas.
- Actualización de la librería de expresiones regulares (11).

- **ADOdb**

Es una librería de abstracción de BD para PHP y Python. Esta permite a los programadores desarrollar aplicaciones web de una manera portable, rápida y fácil, en opinión de muchos usuarios, se trata de la librería de abstracción más rápida existente en la actualidad para PHP. ADOdb es código abierto, y tiene un mantenimiento muy exhaustivo, además de una amplia comunidad de usuarios que ofrece soporte continuo e innovación a sus funcionalidades. La principal ventaja reside en que la BD puede cambiar sin necesidad de reescribir cada llamada a la BD realizada por la aplicación, es decir permite implementar fácilmente una capa de abstracción de BD, de forma tal que se permita utilizar cualquier Sistema Gestor de Base de Datos sin cambiar el código de la aplicación.

- **NuSOAP**

NuSOAP es un kit de herramientas (ToolKit) para desarrollar SW bajo el lenguaje PHP. Esta compuesto por una serie de clases que hacen mucho más fácil el desarrollo de SW. Provee soporte para el desarrollo de clientes (aquellos que consumen los SW) y de servidores (aquellos que los proveen). NuSOAP está basado en SOAP 1.1, WSDL 1.1 y HTTP 1.0/1.1. No es el único soporte para SW en PHP, pero es uno de

los que están en una fase de desarrollo mucho más avanzada, es decir está en una fase madura de desarrollo, además no necesita módulos adicionales y es muy fácil su instalación y uso.

- **Herramienta de desarrollo**

Zend Studio es uno de los ambientes de desarrollo integrado o Integrated Development Environment (IDE) disponible para desarrolladores profesionales, agrupa todos los componentes necesarios para ciclos de desarrollo de aplicaciones PHP. A través de un comprensivo conjunto de herramientas de edición, depurado, análisis, optimización y BD, Zend Studio acelera los ciclos de desarrollo y simplifica los proyectos complejos.

- **Arquitectura en capas**

La arquitectura en capas es un estilo que se basa en la idea de dividir un problema en partes más pequeñas que puedan ser manejadas, desarrolladas de forma independientemente, donde cada parte tenga responsabilidades específicas que no dependan del funcionamiento de las otras, o que su dependencia sea mínima, es decir, cada capa compone un subsistema en el que se ubican clases con responsabilidades propias.

Una buena arquitectura de software debe facilitar los requerimientos de mantenimiento, reusabilidad, escalabilidad, y robustez del mismo. Al concertar la solución de un problema como una serie de capas, cada capa debe ocuparse de un subconjunto de responsabilidades fuertemente acopladas y tener poca cohesión con las demás. Los cambios internos en cualquier capa deben ocasionar la menor cantidad posible de cambios en las restantes (12).

Las ventajas de la arquitectura en capas son bastantes obvias, por ejemplo proporciona amplia reutilización, ya que al hacer abstracciones de las distintas funcionalidades o responsabilidades del sistema agrupándolas en capas, se obtendrán clases que serán independientes del resto, con lo que se podrán reutilizar fácilmente.

Asimismo se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas (13).

Resumiendo los beneficios que ofrece este estilo de arquitectura se tiene:

- Aislamiento de la lógica de la aplicación en componentes separados reutilizables en otras aplicaciones.
- Distribución de capas en diferentes máquinas o procesos, lo que puede mejorar el rendimiento, aumentar la coordinación y la compartición de información entre cliente y servidor.
- Dedicación de recursos a cada una de las capas y posibilidad de desarrollarlas en paralelo (14).

- **Arquitectura Orientada a Servicios**

La Arquitectura Orientada a Servicios (SOA) define la utilización de servicios para dar soporte a los requerimientos de software del usuario. 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.

En este modelo existen tres actores principales: el proveedor del servicio, el registro del servicio y el solicitante del servicio. Un componente en esta arquitectura podría considerarse como un servicio que puede ser publicado, descubierto o invocado de forma dinámica. La característica más importante de este modelo es el bajo grado de acoplamiento entre los componentes junto a una mayor flexibilidad a cambios futuros ya que una modificación en el diseño interno puede ser factible sin necesidad de modificar el servicio (15).

La comunicación hacia y desde el servicio, es realizada utilizando mensajes y no llamadas a métodos. Estos mensajes deben contener o referenciar toda la información necesaria para entenderlo. La idea es que haya el mínimo posible de llamadas entre el cliente y el servicio. Los sistemas interactúan con el SW

de una manera prescripta por su descripción utilizando mensajes SOAP, típicamente transportados usando HTTP con una serialización en XML en conjunción con otros estándares relacionados con la Web.

SOA se basa en la independencia de plataformas de hardware, de sistemas operativos y de lenguajes de programación ya que al contrario de las arquitecturas orientado a objetos, la arquitectura SOA esta formada por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación (WSDL). La definición de la interfaz encapsula las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo. Con esta arquitectura, se pretende que los componentes software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar (15).

- **Metodología de desarrollo**

Muchas empresas dedicadas al desarrollo de software se ven afectadas en la productividad, dado que los software desarrollados no son culminados en las fechas planificadas, los costos no son los esperados, además de que no se satisfacen los requisitos planteados por los usuarios, dado que no utilizan alguna herramienta que les permita seguir de cerca estos aspectos y que por tanto les sirva como una guía para el desarrollo de las aplicaciones.

Para enfrentar esta situación las empresas requieren desarrollar o adquirir una disciplina en el desarrollo del software y controlar que los ingenieros usen de forma consistente los nuevos métodos. Cualquier camino que siga una empresa de software para obtener buena calidad implica mejorar el proceso de desarrollo de software, que no es más que seguir una metodología de desarrollo con procesos bien definidos que les permita obtener resultados óptimos.

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

Algunas Metodologías:

Extreme Programing (XP): Es una de las metodologías de desarrollo de software más exitosa en la actualidad, utilizadas para proyectos de corto plazo, y corto equipo. 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.

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 (16).

Microsoft Solution Framework (MSF): Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF tiene las siguientes características:

- Adaptable
- Escalable
- Flexible

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. (16)

Justificación de la metodología utilizada

Después de haber realizado un análisis de las metodologías más importantes para el desarrollo de software, se decide utilizar la metodología RUP (Proceso Unificado de Desarrollo), dado que después de muchos años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través de UML, y trabajo de muchas metodologías utilizadas por los clientes se ha logrado estandarizar RUP como proceso de desarrollo de software.

RUP es una metodología para el desarrollo de software que esta dirigida a la programación orientación a objetos, sus principales características: guiado por los casos de uso, centrado en la arquitectura e iterativo e incremental hacen de esta una de las metodologías más eficaces y productivas en el mundo de la producción de software.

Esta metodología permite realizar los requisitos del sistema mediante los casos de uso, que van guiando todo el proceso de desarrollo, centrándose en las primeras etapas en los casos de uso arquitectónicamente más significativos y de forma iterativa e incremental se van implementando e integrando los mismos hasta lograr una aplicación ejecutable.

A pesar de ser una metodología desarrollada directamente para el trabajo con clases y objetos brinda amplias posibilidades con el manejo eficiente del tiempo de diseño e implementación de aplicaciones Web.

Hay que destacar que el RUP capacita a las organizaciones de muchas maneras, la más significativa es que proporciona la forma en la que el equipo de proyecto puede trabajar de forma conjunta con los clientes y demás implicados, lo que favorece una mayor organización y entendimiento de lo que realmente el cliente necesita y una excelente proyección del trabajo.

- **Lenguaje de modelado.**

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

documentar un sistema de software. UML ofrece un estándar para describir el modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de BD y componentes de software reutilizables.

UML se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir en el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para soportar una metodología de desarrollo de software pero no especifica en sí mismo qué metodología o proceso usar.

UML es ahora un estándar para el diseño de software orientado a objetos. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama (17).

Los principales beneficios de UML son:

- Mejores tiempos totales de desarrollo (de 50 % o más).
- Modelar sistemas utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos (18).

1.11 Conclusiones

Dado el análisis del objeto de estudio, se observó la revolución que ha causado el desarrollo de los SW cuando se habla de interoperabilidad entre aplicaciones, dado las facilidades que el uso de los mismos proporcionan al basarse en estándares, y las tecnologías que han dado pie al auge de los mismos. Asimismo se hizo un estudio de cuáles lenguajes de programación, herramientas, metodologías de desarrollo de software, son los más factibles a utilizar para el desarrollo de SW siguiendo la política de Software Libre.

Capítulo 2

Descripción de la solución propuesta

2.1 Introducción

En este capítulo se describen los principales procesos involucrados en el objeto de estudio. Se hace una descripción de los conceptos más importantes en el dominio del problema que son expresados o se incluyen en los requerimientos y que por tanto tienen un gran peso en la construcción del sistema. Se realiza una descripción general de los requisitos funcionales y no funcionales del sistema a desarrollar. Además incluye las descripciones de los actores y casos de uso del sistema, así como los diagramas que representan a estos últimos.

2.2 Definición de las entidades y los conceptos principales

- **Aplicación:** Sistema externo que va a utilizar los servicios Web brindados por nuestra aplicación**.**
- **Servicio:** Aplicación que es ofrecida para ser utilizada por solicitantes que completan los requisitos especificados por el proveedor de servicios.
- **Proveedor de Servicios:** Sistema que tiene publicado diferentes servicios ha ser usados por distintas aplicaciones.
- **Ciudadano:** Base de Datos que contiene toda la información común de las personas de la UCI.

2.3 Representación del modelo del dominio

Un modelo conceptual (Modelo del Dominio) explica los conceptos significativos en el dominio del problema, constituyendo el artefacto más importante durante el análisis orientado a objetos, ya que descompone el problema en conceptos u objetos individuales.

Este permite esclarecer la nomenclatura del dominio, además cuáles son los términos más importantes y cómo se relacionan entre sí (19).

Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en que trabaja el sistema.

Este modelo conceptual se ilustra en UML mediante un diagrama de clases que permite modelar las asociaciones o interacciones entre los diferentes conceptos esclareciéndoles a los clientes, usuarios, desarrolladores, como estos se relacionan entre sí.

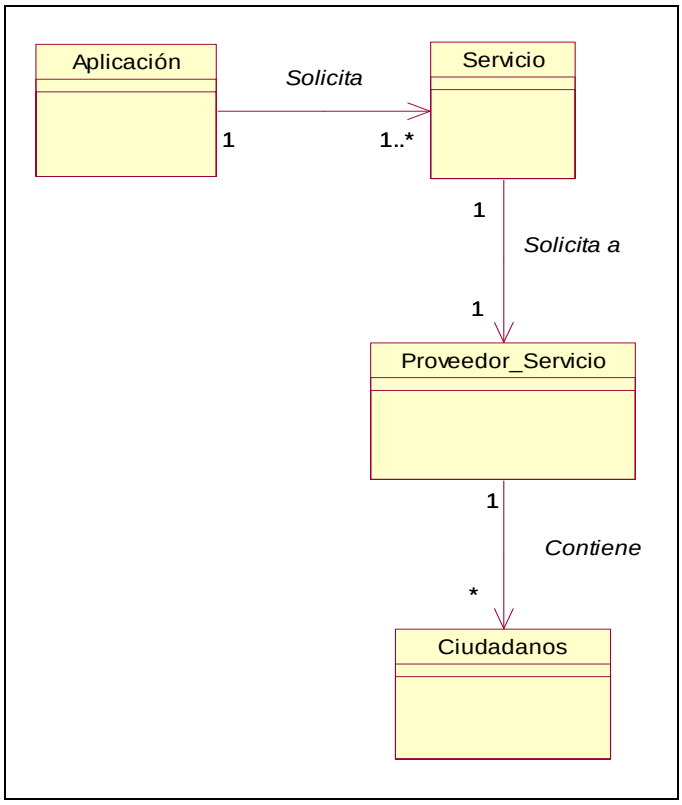

Figura 3: Diagrama del modelo del dominio

2.4 Requerimientos funcionales

Para dar cumplimiento a los objetivos planteados se han identificado los siguientes requerimientos funcionales.

El sistema debe ser capaz de:

R1: Buscar información de personas.

1.1 Buscar datos especificados de una persona.

1.2 Buscar la foto de una persona.

1.3 Buscar el correo de una persona.

1.4 Buscar el login (usuario) de una persona.

R2: Devolver listados de personas.

2.1 Devolver listado de estudiantes

2.2 Devolver listado de profesores.

2.3 Devolver listado de dirigentes.

2.4 Devolver listado de terciarizados

2.5 Devolver listado de familiares.

2.6 Devolver listado de trabajadores.

2.7 Devolver listado de eventuales.

2.8 Devolver listado de personas que han causado altas desde una fecha determinada.

2.9 Devolver listado de personas que han causado bajas desde una fecha determinada.

R3: Devolver nomencladores globales.

3.1 Devolver nomencladores por provincia.

3.2 Devolver nomencladores por municipio.

3.3 Devolver nomencladores por raza.

3.4 Devolver nomencladores por color de ojos.

2.5 Requerimientos no funcionales

- **Usabilidad.**

  La información tendrá que estar disponible en todo momento, las únicas limitaciones serán las acordadas en las políticas de seguridad de la misma.

  El sistema debe permitir un gran número de solicitudes de servicios de otras aplicaciones al mismo tiempo.

- **Rendimiento.**

  Se debe garantizar que el sistema sea lo más rápido y eficiente posible para poder lograr un tiempo de respuesta adecuado a las necesidades de las aplicaciones.

- **Confiabilidad.**

  Tendrá que garantizar la consistencia de los datos en todo momento.

- **Para una correcta ejecución de la aplicación se necesita (De servidores)**

  Sistema Gestor de Base de Datos (SGBD): SQL-Server 2000

  Servidor Web: Apache

- **Hardware (De servidores)**

  Velocidad del Procesador: 3.6 GHz

  Mínimo de capacidad Memoria RAM: 1G

  Mínimo de capacidad de Disco duro: 80G

2.6 Actores del sistema

ACTORES DEL SISTEMA	DESCRIPCIÓN
Sistema Externo	Son los sistemas clientes a quienes se les brinda los diferentes servicios.

Tabla 1: Definición de Actores del Sistema a Automatizar

2.7 Casos de uso del sistema a automatizar

Un caso de uso es un fragmento de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores (20).

Mediante UML se modela el diagrama de casos de uso del sistema que representa gráficamente a los procesos y su interacción con los actores, identificando previamente cuáles serían los actores que van a interactuar con el sistema, y los casos de uso que van a representar las diferentes funcionalidades, modelando de esta forma los requerimientos funcionales que el sistema debe cumplir.

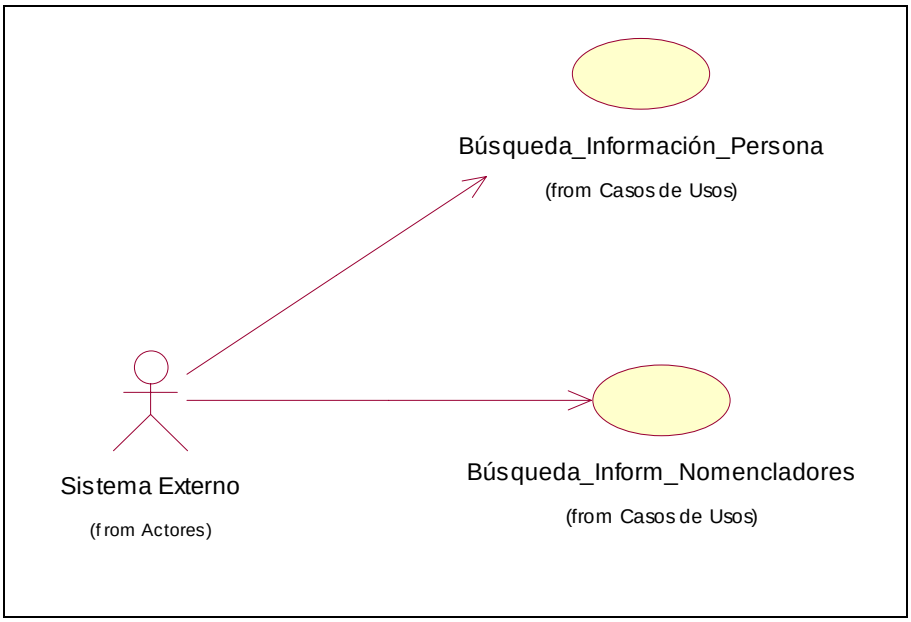

Figura 4: Diagrama de CU del sistema

2.8 Descripción de los casos de uso

Caso de uso:	**Búsqueda_Inform_Nomencladores**
Actores:	Sistema_Externo
Propósito:	Obtener información sobre los nomencladores globales.
Resumen:	El caso de uso inicia cuando el sistema externo solicita un listado de los nomencladores de un concepto en específico, o conocer dado un identificador el nombre del mismo. El sistema obtiene la información que es devuelta al sistema externo finalizando así el caso de uso.
Referencias:	R3.1,R3.2,R3.3,R3.4
Precondiciones:	

Poscondiciones:	
Caso de uso asociado:	
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1. El sistema externo solicita. a) Si solicita ver los identificadores y nombres de los nomencladores del concepto provincia ver sección Nomencladores _ Provinciales. b) Si solicita ver el nombre de un nomenclador específico dentro del concepto provincia ver sección Obtener_Nombre_Dado_Identificador_Provincia. c) Si solicita ver los identificadores y nombres de los nomencladores del concepto municipio ver sección Nomencladores _ Municipales. d) Si solicita ver el nombre de un nomenclador específico dentro del concepto municipio ver sección Obtener_Nombre_Dado_Identificador_Municipio. e) Si solicita ver los identificadores y nombres de los nomencladores del concepto raza ver sección Nomencladores _ Raza. f) Si solicita ver el nombre de un nomenclador específico dentro del concepto raza ver sección Obtener_Nombre_Dado_Identificador_Raza. g) Si solicita ver los identificadores y nombres de los	

nomencladores del concepto color de ojos ver sección Nomencladores _ ColorOjos. h) Si solicita ver el nombre de un nomenclador específico dentro del concepto color de ojos ver sección Obtener_Nombre_Dado_Identificador_ColorOjos.	
Sección Nomencladores _ Provinciales.	
1. El sistema externo solicita el listado de los identificadores y nombres de los nomencladores del concepto provincia.	2. El sistema confecciona el listado.
	3. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
Sección Obtener_Nombre_Dado_Identificador_Provincia.	
1. El sistema externo solicita conocer el nombre de un nomenclador dentro del concepto provincia especificando el identificador del mismo.	2. El sistema verifica que el identificador sea válido.
	3. El sistema obtiene el nombre de la provincia correspondiente al identificador especificado.
	4. El sistema devuelve la respuesta al sistema

	externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador entrado no es válido.
Sección Nomencladores _ Municipales	
1. El sistema externo solicita el listado de los identificadores y nombres de los nomencladores del concepto municipio.	2. El sistema obtiene el listado.
	3. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
Sección Obtener_Nombre_Dado_Identificador_Municipio	
1. El sistema externo solicita conocer el nombre de un nomenclador dentro del concepto municipio especificando el identificador del mismo.	2. El sistema verifica que el identificador sea válido
	3. El sistema obtiene el nombre del municipio

	correspondiente al identificador especificado.
	4. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador entrada no es válido.
Sección Nomencladores _ Raza	
1. El sistema externo solicita el listado de los identificadores y nombres de los nomencladores del concepto raza.	2. El sistema obtiene el listado.
	3. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
Sección Obtener_Nombre_Dado_Identificador_Raza	
1. El sistema externo solicita conocer el nombre de un nomenclador dentro del concepto raza	2. El sistema verifica que el identificador sea válido

especificando el identificador del mismo.	
	3. El sistema obtiene la raza correspondiente al identificador especificado.
	4. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador entrada no es válido.
Sección Nomencladores _ ColorOjos	
1. El sistema externo solicita el listado de los identificadores y nombres de los nomencladores del concepto color de ojos.	2. El sistema obtiene el listado.
	3. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
Sección Obtener_Nombre_Dado_Identificador_ColorOjos	

1. El sistema externo solicita conocer el nombre de un nomenclador dentro del concepto color de ojos especificando el identificador del mismo.	2. El sistema verifica que el identificador sea válido
	3. El sistema obtiene el color de ojos correspondiente al identificador especificado.
	4. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador entrada no es válido.
Requerimientos especiales.	

Tabla 2: Descripción del CU Búsqueda_Inform_Nomencladores

Caso de uso:	**Búsqueda_Información_Ciudadano**
Actores:	Sistema Externo
Propósito:	Obtener información de las personas.
Resumen:	El caso de uso inicia cuando el Sistema Externo especifica ciertos criterios de búsquedas y dado estos el sistema obtiene los resultados pedidos que son devueltos al sistema externo, finalizando así el caso de uso.
Referencias:	R1.1, R1.2, R1.3, R1.4, R2.1, R2.2, R2.3, R2.4, R2.5, R2.6, R2.7, R2.8, R2.9.
Precondiciones:	
Poscondiciones:	
Caso de uso asociado:	

Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1. El sistema externo solicita. a) Si solicita obtener datos de una persona ver sección Buscar_Datos. b) Si solicita obtener la foto de una persona ver sección Obtener_Foto. c) Si solicita obtener login de una persona ver sección Obtener_Login.	

d) Si solicita obtener correo de una persona ver sección Obtener_Correo. e) Si solicita obtener datos de una persona especificando su usuario ver sección Obtener Datos_Dado_Usuario. f) Si solicita obtener datos de una persona dado un criterio de búsqueda ver sección Buscar_Datos_Dado_Filtro. g) Si solicita obtener un listado de identificadores de personas especificando el tipo de persona ver sección Obtener_IdPerosna_Dado_TipoPersona.j h) Si solicita listar las personas que han causado alta desde una fecha especifica ver sección Listar _ Altas. j) Si solicita listar las personas que han causado baja desde una fecha especifica ver sección Listar _ Bajas.	
Sección Buscar_Datos	
1. El sistema externo especifica el identificador de una persona (IdPersona), además los campos de la misma que desea obtener dentro de los especificados: Nombre, 1er Apellido, 2do Apellido, Tipo de Persona, Sexo, Raza, Provincia, Municipio, Color de Ojos, Foto, Login, Teléfono, Apartamento.	2. El sistema verifica que el identificador sea válido.

	3. El sistema verifica que los campos pedidos existan.
	5. El sistema obtiene la información solicitada.
	6. El sistema devuelve la respuesta al sistema externo.
Cursos alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador no es válido.
	3. El sistema informa al sistema externo que de todos los campos solicitados no se tiene información.
Sección Obtener_Foto	
Acción del actor:	**Respuesta del sistema:**
1. El sistema externo solicita obtener la foto de una persona especificando el identificador de la misma (IdPersona).	2. El sistema verifica que el identificador sea válido.
	3. El sistema obtiene la información solicitada.
	4. El sistema devuelve la respuesta al sistema externo.

Cursos alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador no es válido.
Sección Obtener_Login	
1. El sistema externo solicita obtener el login (usuario) de una persona especificando el identificador de la misma (IdPersona).	2. El sistema verifica que el identificador sea válido.
	3. El sistema obtiene la información solicitada.
	4. El sistema devuelve la respuesta al sistema externo.
Cursos alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador no es válido.
Sección Obtener_Correo	
1. El sistema externo solicita obtener el correo de una persona especificando el identificador de la misma (IdPersona).	2. El sistema verifica que el identificador sea válido.

	3. El sistema obtiene el nombre de usuario.
	4. El sistema determina el tipo de persona.
	5. El sistema conforma la dirección de correo.
	6. El sistema devuelve la respuesta al sistema externo.
Cursos alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el identificador no es válido.
	3. El sistema informa al sistema externo que el nombre de usuario no existe.
Sección Obtener Datos_Dado_Usuario	
1. El sistema externo solicita dado el usuario de una persona(login) obtener cierta información de la misma especificando los campos que desea dentro de los especificados a continuación: Nombre, 1er Apellido, 2do Apellido, Tipo de Persona, Sexo, Raza, Provincia, Municipio, Color de Ojos, Foto, Login, Teléfono, Apartamento, IdPersona.	2. El sistema verifica que el usuario sea válido.
	3. El sistema verifica que los campos pedidos existan.

	5. El sistema obtiene la información solicitada.
	6. El sistema devuelve la respuesta al sistema externo.
Cursos alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el usuario no es válido.
	3. El sistema informa al sistema externo que los campos solicitados no existen.
Sección Buscar_Datos_Dado_Filtro	
1. El sistema externo especifica un criterio de búsqueda especificando algunos de los siguientes campos: Nombre, 1er Apellido, 2do Apellido, Tipo de Persona, Sexo, Raza, Provincia, Municipio, Color de Ojos, Foto, Login, Teléfono, Apartamento, IdPersona y solicita obtener los identificadores de personas (IdPersona) que cumplan con el mismo.	2. El sistema verifica que los valores dados en los campos del criterio de selección sean válidos.
	3. El sistema obtiene el listado de las personas que cumplan con el criterio de selección especificado.

	4. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que no todos los valores dados son válidos.
Sección Obtener_IdPerosna_Dado_TipoPersona	
1. El sistema externo solicita obtener especificando un tipo de persona: Estudiante, Profesor, Trabajador, Terciarizado, Familiar, Eventual, Dirigente, los identificadores de las personas (IdPersona) que pertenezcan a este tipo.	2. El sistema verifica que el tipo de persona sea valido.
	3. El sistema obtiene la información solicitada.
	4. El sistema devuelve la respuesta al sistema externo.
Curso alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que el tipo de persona especificado no es válido.

Sección Listar_Altas	
1. El sistema externo solicita obtener un listado con las personas (IdPersona) que han causado alta desde una fecha determinada especificando la misma.	2. El sistema verifica que la fecha sea válida.
	3. El sistema obtiene la respuesta solicitada.
	4. El sistema devuelve la respuesta al sistema externo.
Curso Alterno	
Acción del actor:	**Respuesta del sistema:**
	2. El sistema informa al sistema externo que la fecha no es válida.
Sección Listar_Bajas	
1. El sistema externo solicita obtener un listado con las personas (IdPersona) que han causado baja desde una fecha determinada especificando la misma.	2. El sistema verifica que la fecha sea válida.
	3. El sistema obtiene la respuesta solicitada.

	4. El sistema devuelve la respuesta al sistema externa.
Curso Alterno	
	2. El sistema informa a la aplicación que la fecha no es válida.
Requerimientos especiales:	

Tabla 3: Descripción del CU Búsqueda_Información_Ciudadano

2.9 Conclusiones

En este capítulo se modelaron los principales conceptos que le dan forma al sistema mediante el Modelo de Dominio, representando como se relacionan los mismos para la construcción del sistema. Se describió de forma general los requerimientos funcionales y no funcionales del sistema, así como los actores que intervienen. Mediante los diagramas de casos de uso y la descripción de los mismos se especificaron en detalle los casos de usos a automatizar.

Capítulo 3

Diseño y construcción de la solución propuesta

3.1 Introducción

En este capítulo se plantea la descripción del diseño y construcción de la solución propuesta mediante la representación de diagramas de clases del diseño, de componentes y despliegue. Además se especifica un caso de prueba propuesto para comprobar la aplicación.

3.2 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en el cumplimiento de los requisitos funcionales y no funcionales, es decir, se modela el sistema y encuentra su forma para dar soporte a todos los requerimientos. Para ello mediante los diagramas de clases del diseño se relacionan las clases necesarias para el desarrollo de la aplicación.

3.2.1 Diagramas de clases del diseño

Para darle solución a esta aplicación se identificaron dos casos de uso, Búsqueda_Inform_Nomencladores y Búsqueda_Información_Personas, se relacionan a continuación los diagramas de clases correspondientes a estos casos de usos.

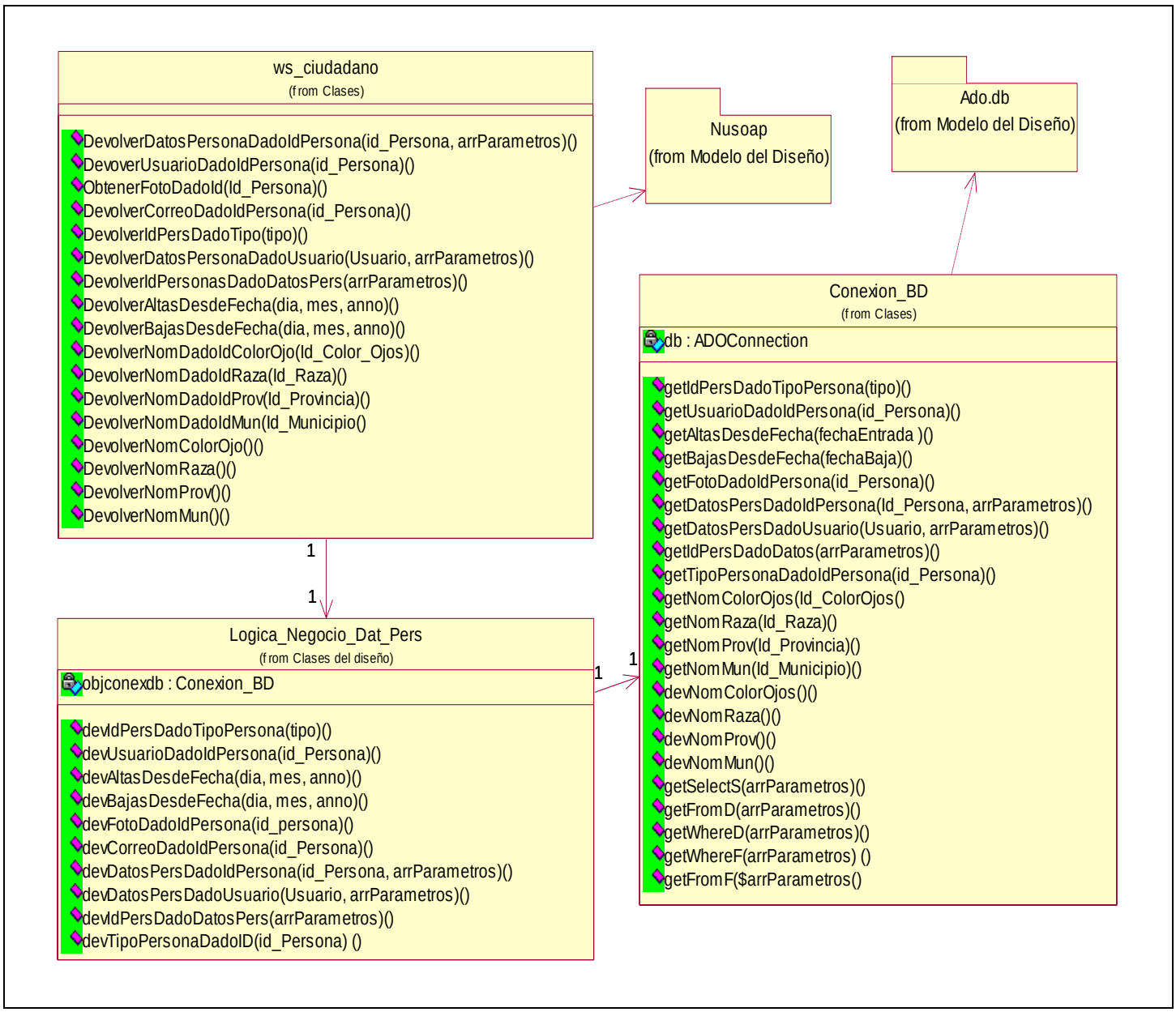

Figura 5: Diagrama de clases del diseño del CU Búsqueda_Informormación_Personas

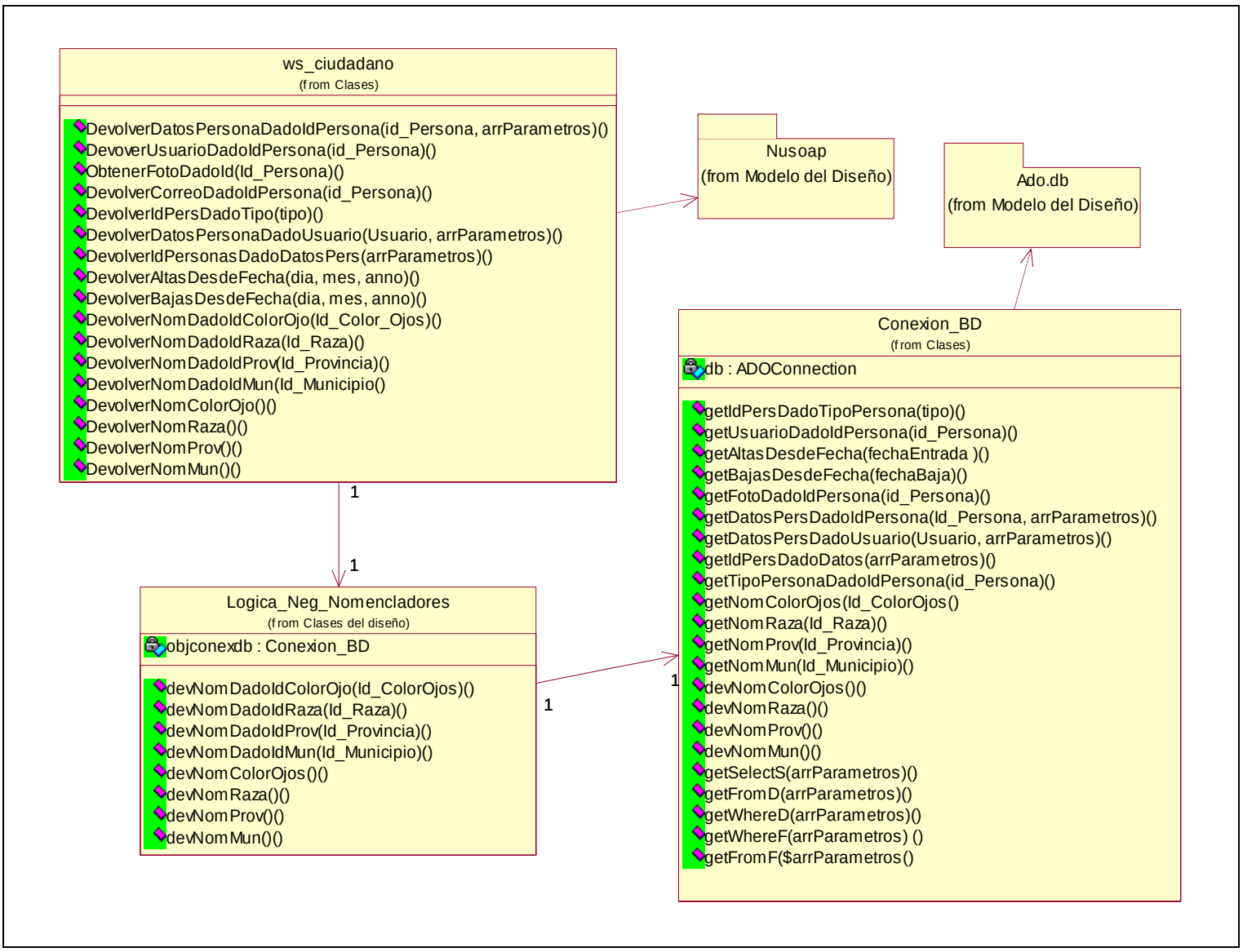

Figura 6: Diagrama de clases del diseño del CU Búsqueda_Inform_Nomencladores

3.3 Principios del diseño

3.3.1 Descripción de los servicios Web

Como un manual de operaciones, los servicios Web y con ello la aplicación desarrollada en este trabajo, cuentan con un lenguaje de descripción estándar, WSDL, que permite especificar la información necesaria para que una aplicación cliente sepa como interactuar o acceder a los servicios. En este se especifica la ubicación del servicio Web, los parámetros de entradas y de salidas de los servicios, el tipo de dato respectivamente, así como una breve descripción de la funcionalidad del servicio.

3.3.2 Patrones de diseño

Para el diseño del software se hace uso de los patrones GRASP, para la asignación de responsabilidades.

- Experto: Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.
- Creador: Ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que:
    - Tiene la información necesaria para realizar la creación del objeto, o
    - Usa directamente las instancias creadas del objeto, o
    - Almacena o maneja varias instancias de la clase
- Alta cohesión: Especifica que la información que almacena una clase debe de ser coherente y estar en la mayor medida de lo posible relacionada con la clase.
- Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (21).

3.3.3 Estándar de codificación

La utilización de un estándar de codificación resulta muy conveniente a la hora de escribir el código, dado que trae consigo significativas ventajas.

- Reducir errores.
- Escribir un código comprensible y fácil de leer.
- Garantizar una buena comunicación entre los programadores del equipo.
- Facilitar el mantenimiento del software.

Por tanto para el desarrollo de este trabajo se utiliza el estándar propuesto por el grupo de desarrollo de la librería PEAR para el lenguaje de programación PHP, que es un estándar definido y aceptado internacionalmente, además es el estándar especificado por el Departamento de Informatización quien es nuestro principal cliente.

3.4 Modelo de Implementación

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, se describe como los elementos del diseño, como las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, etc**.**

3.4.1 Diagramas de componentes

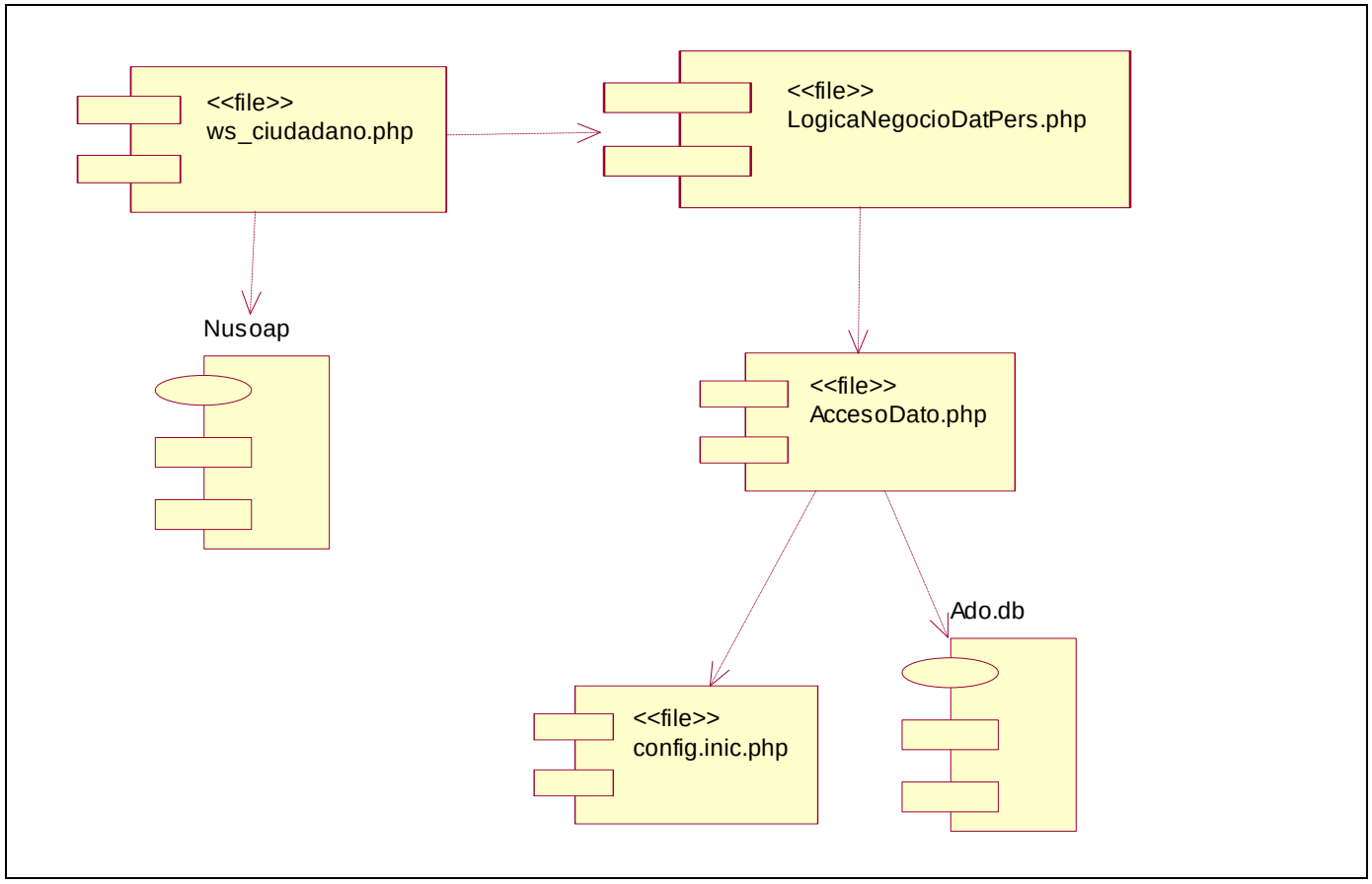

Figura 7: Diagrama de componentes del CU Búsqueda_Información_Personas

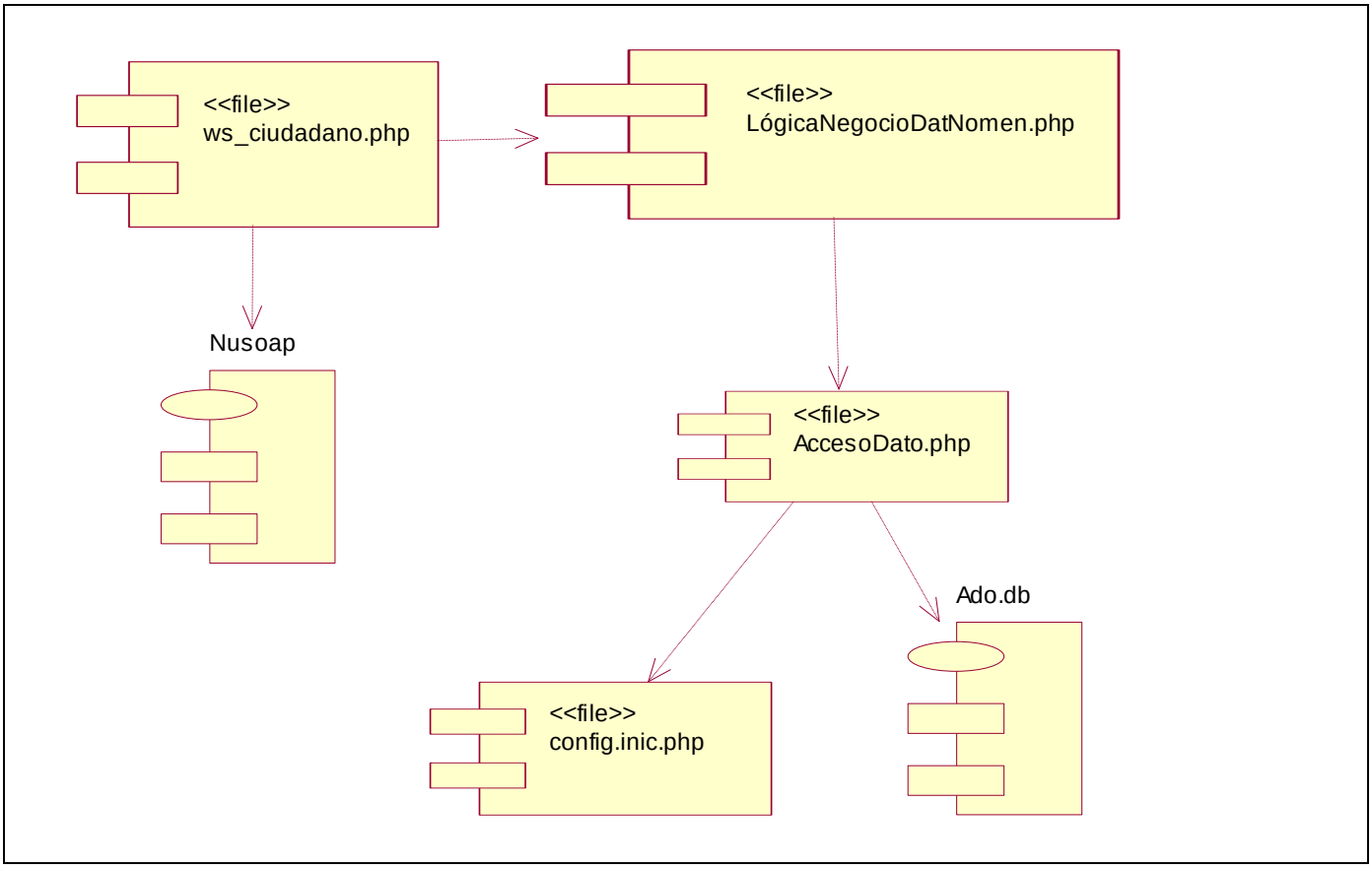

Figura 8: Diagrama de componentes del CU Búsqueda_Inform_Nomencladores

3.5 Modelo de despliegue

La fachada de servicios Web se ha desarrollado sobre la base de la lógica de funcionamiento del patrón arquitectónico Capas, definiendo su arquitectura en tres capas, aunque no se implemente la capa de presentación, siguiendo además un estilo arquitectónico orientado a servicios.

La figura 9 muestra el diagrama de despliegue, donde el nodo PC_Cliente representa las diferentes aplicaciones que harán uso de los servicios Web, el nodo Servidor _ Web representa el servidor donde serán publicados los servicios Web y donde por tanto se ubican además las capas de lógica del negocio y acceso a datos y el nodo Servidor_BD, representa el servidor donde se encuentra la Base de Datos Ciudadano.

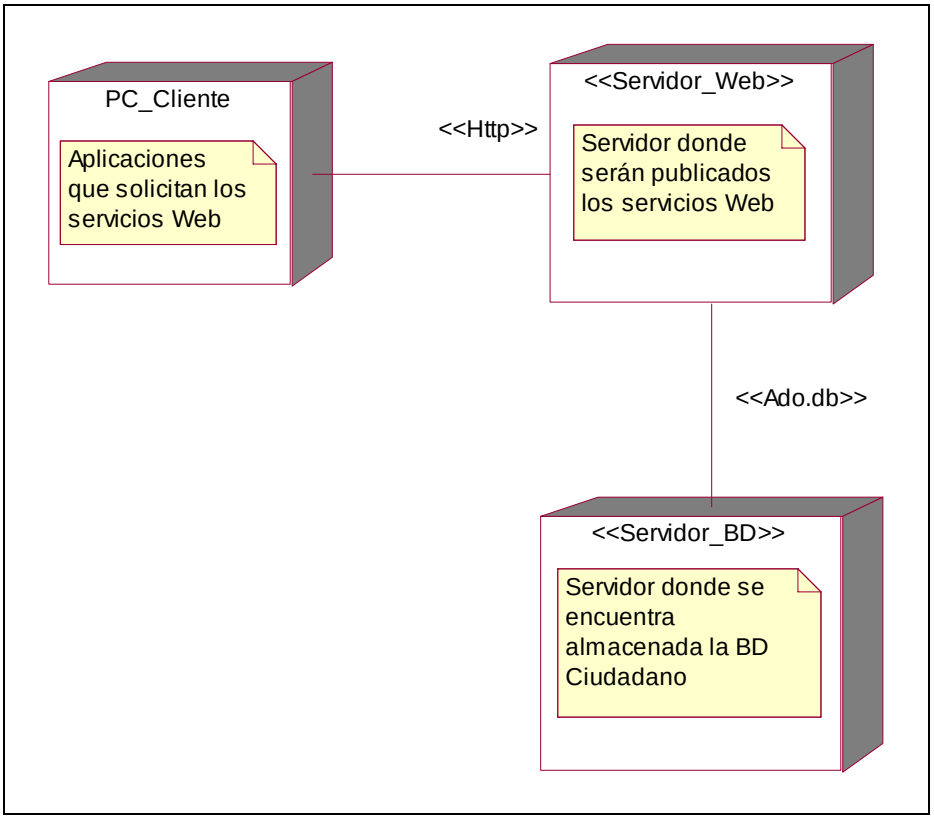

Figura 9: Diagrama de despliegue

3.6 Prueba del sistema propuesto

El objetivo de las pruebas del software es descubrir errores, es decir aquellas fallas que impiden que el programa cumpla con sus requisitos y que por tanto afectan la calidad del software. En este trabajo se utiliza la prueba de caja blanca con la cual se comprueban los caminos lógicos del software, proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones o bucles a nivel de código fuente. Dentro de este tipo de prueba se hace uso del método del camino básico, el cual permite obtener una medida de la complejidad lógica de la aplicación, además garantiza que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

3.6.1 Prueba de la caja blanca

```
    public function getDatosPersDadoIdPersona ($Id_Persona, $arrParametros)
{
if (!$this->db->IsConnected()) {                                                    1
    $this->db->Connect (SERVER_DB, USUARIO_BD, PASSW_BD, NOMBRE_BD);                2
$retorno = array ();                                                                3
$strFrom = $this->getFromD ($arrParametros);
$strWhere = $this->getWhereD ($arrParametros);
$strSelect = implode (“, '', $arrParametros);
$strCadena = “Select". $strSelect. $strFrom. $strWhere." (Tbl_Persona.Id_Persona = "'"".$Id_Persona."'"
)";
$recordSet = $this->db->GetAll ($strCadena);
if ($recordSet) {                                                                   4
      $h= 0;                                                                        5
      for ($f = 0; $f < count ($recordSet); $f++) {                                       6
       while (list ($key, $value)=each($recordSet[$f])) {                           7
            $retorno [$h] = $key."=". $value;                                       8
            $h++;
            }
      }
        $this->db->Close ();                                                        9
        return $retorno;
}
else {
      $this->db->Close ();                                                          10
    return '';
}
}                                                                                   11
```

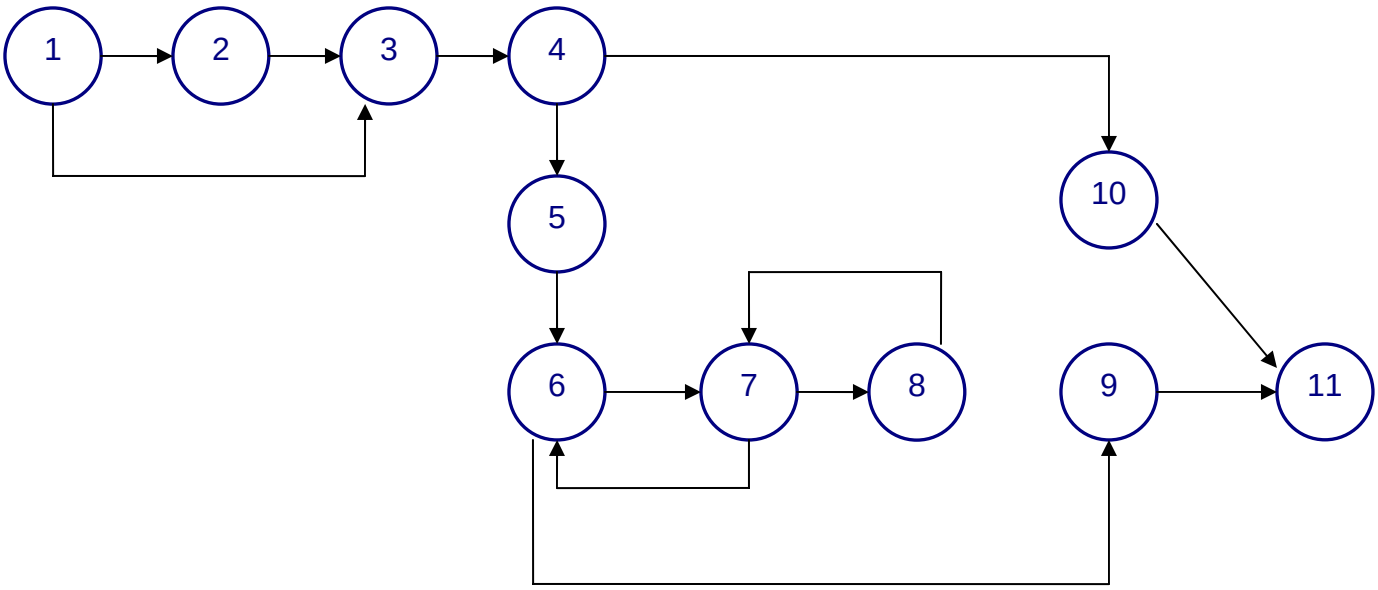

Figura 10: Prueba de caja blanca. Cálculo de la complejidad ciclomática

Complejidad ciclomática [V (G)] = Cantidad de Aristas [A] – Cantidad de nodos [N] + 2

V (G) = A – N + 2

V (G) = 14 - 11 + 2

V (G) = 5

3.8 Conclusiones

En este capítulo se realizó el modelo del diseño, describiendo la realización de los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación. Se describió mediante los diagramas de despliegue y componentes, los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará a aplicación. Con la especificación de todos estos elementos se cuenta con una base sólida para pasar a la construcción de la aplicación.

Capítulo 4

Estudio de factibilidad

4.1 Introducción

En el proceso de desarrollo de software un elemento muy importante es el estudio de factibilidad, pues como resultado de este análisis se obtienen las estimaciones de: esfuerzo, tiempo de desarrollo en meses, costo del producto, la cantidad de personas, aspectos que son muy importantes a la hora de hacer una planificación del proyecto, de determinar cuan factible es la continuación del mismo, haciendo una comparación entre el costo y los beneficios que el mismo reporta. Además posibilita fijar con los clientes una fecha de terminación del producto, dirigiendo todo el esfuerzo a lograr la construcción del producto en el tiempo establecido. En este capítulo se describe la estimación de costos del sistema propuesto y sus beneficios, basado en las técnicas de Análisis de Puntos de Casos de Uso.

4.2 Estimación de Esfuerzos Basado en Casos de Uso

El modelo de casos de uso puede ser usado para estimar el esfuerzo en el desarrollo de un proyecto de software. Existen elementos específicos que representan la diferencia entre un caso de uso y otro:

- La generalización entre los actores.
- Los casos de uso incluidos (include) y extendidos (extended).
- El nivel de detalle en las descripciones.

Si se quiere aplicar la estimación basada en casos de uso, se ha de tener un modelo donde se han identificado detalladamente cada caso de uso, específicamente, el número de transacciones y/o escenarios, siendo esta un evento que ocurre entre un actor y el sistema.

4.3 Planificación basada en casos de uso. Análisis de Puntos de Casos de Uso

Paso 1. Cálculo de los Puntos de casos de uso Desajustados.

UUCP=UAW+UUCW

Donde:

- 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.

Tipo de actor	Descripción	Factor de peso	Actores	Total
Simple	Sistema con sistema a través de interfaz de programación.	1	0	0
Medio	Sistema con sistema mediante protocolo de interfaz basada en texto.	2	1	2
Complejo	Persona que interactúa con el sistema mediante interfaz gráfica.	3	0	0

Tabla 4: Factor de Peso de los Actores sin ajustar

$$UAW = \sum cant\ \ actores * peso$$

UAW=2

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones y/o escenarios.	5	0	0
Medio	El caso de uso tiene de 4 a 7 transacciones y/o escenarios.	10	0	0
Complejo	El caso de uso tiene más de 8 transacciones y/o escenarios.	15	2	30

Tabla 5: Factor de Peso de CU sin ajustar

$$UUCW = \sum cant\,CU * Peso$$

UUCW=30

UUCP=2+30

UUCP=32

Paso 2. Cálculo de los Puntos de casos de uso ajustados.

UCP=UUCP*TCF*EF

Donde:

- UCP: Puntos de casos de uso ajustados.
- UUCP: Puntos de casos de uso sin ajustar.
- TCP: Factor de complejidad técnica.
- EF: Factor de ambiente.

El factor de complejidad técnica (TCF) se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada factor se cuantifica en un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Factor	Descripción	Peso	Valor asignado	Total
T1	Sistema distribuido	2	0	0
T2	Tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	2	2
T4	Funcionamiento interno complejo	1	3	3
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0,5	4	2
T7	Facilidad de uso	0,5	5	2.5
T8	Portabilidad	2	4	8
T9	Facilidad de cambio	1	4	4
T10	Concurrencia	1	5	5
T11	Incluye objetivos especiales de seguridad	1	2	2
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de entrenamiento de usuarios	1	2	2

Tabla 6: Factor de Complejidad Técnica

$$TCF = 0.6 + 0.01 * \sum (peso * valor\ asignado)$$

TCF = 0.6 + 0.01 * 38.5

TCF = 0.6 + 0.345

TCF=0.985

El factor de ambiente (EF) está relacionado con las habilidades y entrenamiento del grupo de desarrollo que realiza el sistema. Cada factor se cuantifica con un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Factor	Descripción	Peso	Valor asignado	Total
E1	Familiaridad con el modelo de proyecto utilizado	1,5	1	1.5
E2	Experiencia en la aplicación	0,5	3	1.5
E3	Experiencia en la programación orientación a objetos.	1	4	4
E4	Capacidad del analista líder.	0,5	2	1
E5	Motivación.	1	4	4
E6	Estabilidad de requerimientos	2	3	6
E7	Personal de media jornada.	-1	5	-5
E8	Dificultad del lenguaje de programación	-1	2	-2

Tabla 7: Factor de Ambiente

$$EF = 1.4 - 0.03 * \sum (peso * valor\ asignado)$$

EF = 1.4 – 0.03 * 11

EF = 1.4 – 0.33

EF=1.07

UCP = UUCP * TCF * EF

UCP = 32 *0.985 * 1.07

UCP=33.7264

Paso 3. Estimación de esfuerzo a través de los puntos de casos de uso.

E=UCP*CF

Donde:

- E: Esfuerzo estimado en horas hombres.
- UCP: Punto de casos de usos ajustados.
- CF: Factor de conversión.

Para obtener el factor de conversión (CF) se cuentan cuántos valores de los que afectan el factor ambiente (E1...E6) están por debajo de la media (3), y los que están por arriba de la media para los restantes (E7, E8). Si el total es 2 o menos se utiliza el factor de conversión 20 Horas-Hombre / Punto de Casos de uso. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de Casos de uso. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

En este caso se puede decir que:

CF = 28 Horas-Hombre / Punto de Casos de uso.

E=33.7264*28

E=944.3392 Horas-Hombre

Paso 4. Calcular esfuerzo de todo el proyecto.

Actividad	Porcentaje %	Horas-Hombres
Análisis	10	236.0848
Diseño	20	472.1696
Implementación	40	944.3392
Pruebas	15	354.1272
Sobrecarga (otras actividades)	15	354.1272
Total	100	2360.848

Tabla 8: Relación Actividad / Porcentaje

Si E_T = 2360.848 horas-hombre y cada mes los desarrolladores trabajan como promedio 192 horas, eso daría un:

E_T = 12.2960833 mes-hombre.

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

Costo del Proyecto.

Se asume como salario promedio mensual $150.00

CHM = 2 * Salario Promedio

CHM = 300.00 $/mes

Costo = Salario Promedio* E_T

Costo = 150.00 * 12.2960833

Costo= $ 1844.4125

4.4 Beneficios tangibles e intangibles

La reconcepción, organización y migración de la fachada de los servicios Web de la Base de Datos Ciudadano reportará importantes beneficios en el proceso de informatización de nuestra Universidad, cuyo principal objetivo es convertirla en el prototipo para la Informatización de la Sociedad Cubana, debido a que con ello se desarrollaran servicios Web que brinden funcionalidades objetivas, que respondan a las necesidades de los diferentes sistemas desarrollados en nuestra Universidad, el tiempo de respuesta de los mismos será un tiempo adecuado para lograr el buen funcionamiento de las aplicaciones que hagan uso de los mismos.

También se logrará abstraer a los diferentes sistemas de la estructura e implementación de la Base de Datos Ciudadano, aspecto de gran importancia para las aplicaciones ya que se independizan del medio de almacenamiento de la información que necesitan. Asimismo reportará significativas ventajas en relación a la seguridad de la información almacenada en la Base de Datos, ya que los servicios Web constituyen una barrera entre las aplicaciones y la Base de Datos minimizando los privilegios de las mismas de modificar la información.

Todo esto contribuirá al buen funcionamiento de los diferentes procesos llevados a cabo en nuestra Universidad y cuya responsabilidad recae en el Departamento de informatización. En relación a los beneficios tangibles, esta es una aplicación desarrollada para beneficios internos, pues son servicios brindados sobre una Base de Datos específica de nuestra Universidad.

4.5 Análisis de costos y beneficios

El desarrollo de un proyecto de software está determinado entre otros aspectos por la relación entre los costos y beneficios, es necesario al menos obtener un equilibrio entre estos dos parámetros, de forma tal, que sea factible el seguimiento del mismo, no sería lógico desarrollar una aplicación cuyos beneficios sean insignificantes en comparación con los costos de la misma.

El desarrollo de la fachada de servicios Web de la Base de Datos Ciudadano, la cual reporta significativos beneficios para las diferentes aplicaciones desarrolladas en nuestra Universidad, para la integración y buen funcionamiento de las mismas y con ello materializando la misión del Departamento de Informatización de convertir nuestra Universidad en una Ciudad Digital tiene un costo de: **$ 1844.4125.**

Por esta razón, haciendo una comparación entre el costo y los beneficios que el desarrollo de esta aplicación reporta se concluye que resulta factible el desarrollo de la misma.

4.6 Conclusiones

En este capítulo se hizo un estudio de la factibilidad del sistema propuesto, mediante la estimación por Puntos de Casos de Uso, concluyendo que dados los costos estimados del sistema, en comparación con los beneficios que el desarrollo del mismo reportaría, resulta prudente seguir el desarrollo e implantación del sistema.

CONCLUSIONES

El desarrollo de este trabajo investigativo y con ello la implementación de la fachada de los servicios Web de la Base de Datos Ciudadano dio cumplimiento a los objetivos planteados, pudiendo arribar a las siguientes conclusiones.

- El uso de los métodos de la investigación científica, particularmente la entrevista, permitió conocer las necesidades de los diferentes sistemas desarrollados en la Universidad, y de esta forma reconcebir los servicios Web de forma tal que satisfagan a los distintos sistemas.
- Se investigó cuáles eran los nombres e identificadores de los nomencladores globales, que son aquellos conceptos que no pertenecen a un sistema en específico, sino que son de uso común para varios sistemas, y que por tanto sus nombres e identificadores deben estar reglamentados centralmente.
- Se logró una reconcepción de la fachada de los servicios Web de la Base de Datos Ciudadano, implementando servicios Web objetivos, rápidos, con mayor seguridad, permitiendo abstraer los diferentes sistemas de la implementación y estructura de la Base de Batos Ciudadano, y logrando un mejor funcionamiento de los mismos.
- La aplicación desarrollada cumplió con los requerimientos pedidos por el Departamento de Informatización, teniendo de esta forma una buena aceptación por parte del cliente.

RECOMENDACIONES

- Que los servicios Web implementados sean publicados en una UDDI, de forma tal que puedan ser usados libremente por cualquier aplicación cliente que los necesite.

- Que cuando se especifiquen los estándares de seguridad para los servicios Web, que sean aplicados a estos servicios para garantizar una mayor seguridad.

REFERENCIAS BIBLIOGRÁFICAS

1. SAMPER ZAPATER, J. J. *Ontologías para servicios Web semánticos de información de tráfico: descripción y herramientas de explotación.* Disponible en: http://www.tesisenxarxa.net/TESIS_UV/AVAILABLE/TDX-0628106-085805//SAMPER.pdf. [Consultado el: 4 de marzo de 2007].

2. YURISBEL VEGA ORTIZ, Y. P. Z. *Sistema Genérico de Seguridad, adaptación para el Sistema de Gestión del INSMET. Trabajo de diploma para optar por el título de Ingeniería Informática.* Universidad de las Ciencias Informáticas, 2006.

3. WIKIPEDIA. *Servicio Web*. Disponible en: http://es.wikipedia.org/wiki/Servicio_Web. [Consultado el: 15 de febrero de 2007].

4. BANKHACKER.COM. *Taller: Computación Distribuida: Servicios Web de Traducción.* Disponible en: http://web-services.bankhacker.com/. [Consultado el: 15 de marzo de 2007].

5. MOSCATELLI, S. *S.O.A.P*. Disponible en: http://www.fing.edu.uy/inco/grupos/lins/docsgen/soap/soap.doc. [Consultado el: 31 de marzo de 2007].

6. MARAÑÓN, A. *Seguridad en servicios Web XML* Disponible en: http://www.instisec.com/publico/verarticulo.asp?id=70. [Consultado el: 16 de febrero de 2007].

7. GONZÁLEZ, B. *Una Visión General II.* Disponible en: http://www.desarrolloweb.com/articulos/1538.php. [Consultado el: 7 de mayo de 2007].

8. HENST, C. V. D. *¿Qué es PHP?* Disponible en: http://www.maestrosdelweb.com/editorial/phpintro. [Consultado el: 30 de marzo de 2007].

9. MEHDI ACHOUR, F. B., ANTONY DOVGAL, NUNO LOPES, PHILIP OLSON, GEORG RICHTER, DAMIEN SEGUY, JAKUB VRANA. *Manual de PHP*. Disponible en: http://www.idiota.es/php/. [Consultado el: 12 de abril de 2007].

10. GONZÁLEZ, R. B. *Un poco sobre PHP.* Disponible en: http://php.prod.uci.cu/?q=node/50. [Consultado el: 1 de junio de 2007].

11. FOUNDATION, T. A. S. *Visión general de las nuevas funcionalidades de Apache 2.0.* Disponible en: http://httpd.apache.org/docs/2.0/new_features_2_0.html. [Consultado el: 31 de mayo de 2007].

12. POINT, P. P. *Arquitectura en Capas* Universidad Pública de Navarra: Disponible en: http://www.mhproject.org/media/blogs/mhpenlaces/Interno/Presentaciones/ATS-Interactiva/Arquitectura%20Tres%20Capas.ppt#263,9,Diapositiva9. [Consultado el: 20 de abril de 2007].

13. MSDN. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#1. [Consultado el: 21 de abril de 2007].

14. UNIVERSIDAD CASTILLA-LA-MANCHA, E. D. I. D. C. R. *Prácticas Ingeniería del Software "Arquitectura Multicapa".* Disponible en: http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr5_Multicapa.pdf. [Consultado el: 21 de abril de 2007].

15. WIKIPEDIA. *Arquitectura orientada a servicios.* Disponible en: http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios_%28inform%C3%A1tica%29. [Consultado el: 3 de mayo de 2007].

16. SÁNCHEZ, M. M. *Metodologías De Desarrollo De Software.* Disponible en: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html. [Consultado el: 20 de mayo de 2007].

17. PERE, M. *Introducción a UML.* Disponible en: http://www.programacion.net/tutorial/uml/1. [Consultado el: 10 de abril de 2007].

18. ROMÁN ZAMITIZ, C. A. *El lenguaje unificado de modelado (uml).* Disponible en: http://www.fi-b.unam.mx/pp/profesores/carlos/aydoo/uml.html. [Consultado el: 10 de abril de 2007].

19. LARMAN, C. *UML y Patrones I, Introducción al análisis y diseño orientado a objetos.*

20. TELEFORMACION UCI, P. P. L. P. D. N. D. I. D. S., CONFERENCIAS_IS1_05_06 CONFERENCIA # 3. *Análisis y diseño.* Disponible en: http://teleformacion.uci.cu/mod/resource/view.php?id=10817&subdir=/Conferencias_IS1_05-06. [Consultado el: 4 de abril de 2007].

21. WIKIPEDIA. *Grasp.* Disponible en: http://es.wikipedia.org/wiki/Grasp. [Consultado el: 11 de junio de 2007].

BIBLIOGRAFÍA

1. Bañares, J. "*Conceptos y Arquitectura de Servicios Web*". Disponible en: [http://iaaa.cps.unizar.es/docencia/SW.html] [Consultado 13 de junio 2007]

2. Bañares, J. "*Conceptos y estándares de arquitecturas orientadas a servicios Web*". Disponible en: [http://iaaa.cps.unizar.es/docencia/SWDoct.html] [Consultado 13 de junio 2007]

3. Esendex.com. *"¿Por qué SOAP? ¿Por qué no utilizar simplemente URLs con cadenas de consulta?".* Disponible en: [http://www.esendex.com/es/Productos-SDK-SMS-FAQ-Por-qu%C3%A9-SOAP-Por-qu%C3%A9-no-utilizar-simplemente-URLs-con-cadenas-de-consulta.aspx] [Consultado 31 de marzo, 2007]

4. Fernando. *"La "metodología RUP".* Disponible en: [http://daguilar.evolutionperu.com/?p=4] [Consultado 31 de marzo, 2007]

5. Hernando, S. "*Vulnerabilidades en ADOdb para PostgreSQL*". Disponible en: [http://archives.postgresql.org/pgsql-es-ayuda/2006-01/msg00648.php] [Consultado 12 de junio 2007]

6. Lim, J. "*ADOdb Database Abstraction Library for PHP (and Python)".* Disponible en: [http://adodb.sourceforge.net/#download] [Consultado 12 de junio 2007]

7. MSDN. *"Información general acerca de Visual Studio .NET 2003 Enterprise Developer."* Disponible en: [http://www.microsoft.com/spanish/msdn/vstudio/productinfo/vstudio03/overview/edoverview.asp] [Consultado 16 de febrero, 2007]

8. Place, E. *"PHP5: Diseño en 3 capas y problemas con subdirectorios "* Disponible en:[http://phpsenior.blogspot.com/2006/04/php5-diseo-en-3-capas-y-problemas-con.html] [Consultado 30

de marzo, 2007]

9. TheServerSide. *"Mejoras en la Versión 2.1 de la Especificación EJB."* Disponible [http://www.programacion.net/tutorial/ejb_21/2/#pag2_intro] [Consultado 31 de marzo, 2007]

10. W3C. *"Guía Breve de Servicios Web."* Disponible en: [http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb] Consultado: 15 de febrero, 2007,

11. Webmaster. *"Tutorial sobre web services: SOAP."* Disponible en: [http://www.malditainternet.com/node/506] [Consultado 7 de mayo, 2007]

12. Webmaster. *"Servicios Web con PHP."* Disponible en:[http://www.malditainternet.com/node/1169] [Consultado 7 de mayo, 2007]

13. Wolter, R. *"Fundamentos de los servicios Web XML."* Disponible en:[http://www.microsoft.com/spanish/msdn/articulos/archivo/151102/voices/fundamentos_xml.asp] [Consultado 16 de febrero, 2007]

ANEXOS

Anexo 1. Entrevista realizada a líderes de Proyectos

Estimado líder de proyecto, somos estudiantes de 5to año de la Universidad de las Ciencias Informática y se está realizando una investigación que facilite la información necesaria para lograr una adecuada reconcepción, organización y migración de la fachada de servicios Web de la Base de Datos Ciudadano. Su colaboración será una ayuda muy importante para nuestra investigación.

Datos personales.

Nombre del líder del proyecto: _____.

Años de experiencia: _____.

1. ¿Considera que la fachada actual de servicios Web de la Base de Datos Ciudadano brinda las funcionalidades requeridas para el desarrollo de su aplicación?
2. ¿Qué servicios utiliza de los que se encuentran actualmente implementados?
3. ¿Se encuentra usted satisfecho con el funcionamiento de estos servicios?
4. ¿Qué servicios usted propone que respondan a las necesidades de su aplicación?
5. ¿Tiene alguna observación que pueda contribuir al perfeccionamiento de la fachada de servicios Web? Especifique:

Anexo 2. Listado de los servicios propuestos por algunas aplicaciones

SERVICIOS			
	Nombre	**Parámetros de entrada**	**Parámetros de salida**
Akademos	ObtenerTodosEstudiantes()		IdPersona
	DevolverAltasDesde()	Fecha	IdPersona
	DevolverBajasDesde()	Fecha	IdPersona
	BuscarPersonas()	Filtro	IdPersona
	ExisteCiudadano()	IdPersona	Verdadero o Falso
	ActulizarFoto()	IdPersona, Foto	
	ObtenerDatosPersona()	IdPersona	Datos especificados
	DevolverLogin()	IdPersona	Login(usuario)
	DevolverDatosPersona()	Login	Datos especificados
	ObtenerFoto()	IdPersona,IdFoto	Foto
	ObtenerCorreo()	IdPersona	Correo
Asset	DevolverTodosLosTrabajadores()		IdPersona
	DevolverAltasDesde()	Fecha	IdPersona
	DevolverBajasDesde()	Fecha	IdPersona
	BuscarPersonas()	Filtro	IdPersona
Eventuates y Terciarizados	DevolverTodosLosTrabajadores()		IdPersona
	DevolverAltasDesde()	Fecha	IdPersona
	DevolverBajasDesde()	Fecha	IdPersona
	BuscarPersonas()	Filtro	IdPersona

Identificación	ObtenerFoto()	IdPersona	URL Foto
	DevolverDado()	CI	Nombre, sexo,Tipo Persona,Fecha EntradaUci, Usuario, ID Uci, IdFoto, IdPersona
	DevolverDado()	Nombre, Sexo, tipo Persona,Fecha EntradaUci	CI, Usuario, ID Uci, IdFoto, IdPersona
	DevolverDado()	Usuario	Nombre, sexo,Tipo Persona,Fecha EntradaUci, CI, ID Uci, IdFoto, IdPersona
	DevolverDado()	IdUci	Nombre, sexo,Tipo Persona,Fecha EntradaUci, CI, Usuario, IdFoto, IdPersona
	DevolverDado()	IdFoto	Nombre, sexo,Tipo Persona,Fecha EntradaUci, CI, Usuario, IdUci, IdPersona

Anexo 3. Listado de servicios Web concebidos para la fachada de la Base de Datos Ciudadano

Servicios		
Nombre	**Parámetros de entrada**	**Parámetros de salida**
ObtenerDatosPersonaDadoID	ID persona	Datos especificados
ObtenerLoginDadoID	ID persona	Login
ObtenerFotoDadoId	ID persona	Foto
ObtenerCorreoDadoId	ID persona	Correo
DevolverIdDadoTipoPersona	TipoPersona	ID persona
BuscarDatosDadoUsuario	Usuario(Login)	Datos de Persona
BuscarPersonas	Filtro	ID Persona
DevolverAltasDesde	Fecha	IdPersona
DevolverBajasDesde	Fecha	IdPersona
NomencladorIdProvincia		IdProvincia,NombreProvincia
NomencladorProvinciaDadoId	IdProvincia	NombreProvincia
NomencladorIdMunicipio		IdMunicipio,NombreMunicipio
NomencladorMunicipioDadoId	IdMunicipio	NombreMunicipio
NomencladorIdRaza		IdRaza,Raza
NomencladorRazaDadoId	IdRaza	Raza
NomencladorIdColorOjos		IdColorOjos,ColorOjos
NomencladorColorOjosDadoId	IdColorOjos	ColorOjos

Anexo 4. Listado de algunos nomencladores

Nomencladores Globales		
Concepto	**Nombre**	**Identificador**
Provincias	Pinar del Rio	01
	La Habana	02
	Ciudad de La Habana	03
	Matanzas	04
	Villa Clara	05
	Cienfuegos	06
	Sancti Spiritus	07
	Ciego de Avila	08
	Camaguey	09
	Las Tunas	10
	Holguín	11
	Granma	12
	Santiago de Cuba	13
	Gunatánamo	14
	Isla de la Juventud	15
	Gibara	1101
	Rafael Freyre	1102
	Banes	1103

Municipios(Holguín)	Antilla	1104
	Báguanos	1105
	Holguín	1106
	Calixto García	1107
	Cacocum	1108
	Urbano Noris	1109
	Cueto	1110
	Mayarí	1111
	Frank País	1112
	Sagua de Tánamo	1113
	Moa	1114

GLOSARIO DE TÉRMINOS

- AKADEMOS: Base de Datos que almacena la información de los estudiantes.
- ASSET: Base de datos que almacena la información de los trabajadores.
- CAPITAL TERCIARIZADO**:** Personas que trabajan en la Universidad, pero que no pertenecen a la misma, es decir, son atendidos directamente por las empresas a la que pertenecen Ej. oficiales Sepcom, los médicos, los dependientes de gastronomía.
- CIUDADANO: Base de datos que almacena la información general de todas las personas que pertenecen a la Universidad.
- CORBA: (Common Object Request Broker Architecture) Es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos
- DCOM: (Distributed Component Object Model) Modelo de Objeto Componente Distribuido. Es un juego de conceptos e interfaces de programa de Microsoft en el cual los objetos de programa del cliente pueden solicitar servicios de objetos de programa servidores en otros ordenadores dentro de una red.
- EXTRANETS: Las Extranets son el puente entre la red pública Internet y las redes privadas corporativas o Intranets, o sea un canal que conecta múltiples y diversas organizaciones online, donde las que comparten información puedan comunicarse con fines comerciales. Es decir, es una herramienta que permite la colaboración entre empresas, comunicación entre empresa y proveedores, de empresa a empresa, de empresas a consumidores.
- FACHADA DE SERVICIOS WEB: Conjunto de servicios Web brindados de un determinado sistema.
- FIREWALL: Dispositivo que funciona como cortafuegos entre redes, permitiendo o denegando las transmisiones de una red a la otra.
- HTTP: Protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol). Es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas Web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido

- IBM: (International Business Machines), conocida coloquialmente como el Gigante Azul, es una empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática.
- INTEROPERABILIDAD: Comunicación.
- MICROSOFT: Acrónimo de Microcomputer Software, es una empresa de Estados Unidos, fundada por Bill Gates y Paul Allen, los cuales siguen siendo sus principales accionistas. Dueña y productora de los sistemas operativos: Microsoft DOS y Microsoft Windows, que se utilizan en la mayoría de las computadoras del planeta.
- NOMENCLADORES GLOBALES: Conceptos que no pertenecen a un sistema en específico, sino que son de uso común para varios sistemas, por lo que sus nombres e identificadores deben estar reglamentados centralmente. Ej. Los nombres de provincia, municipios, etc.
- OASIS: (Organization for the Advancement of Structured Information Standards) es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares.
- OMG: (Object Management Group) Grupo de Gestión de Objeto, es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos.
- RMI: (Remote Method Invocation) son mecanismos para invocar o ejecutar procedimientos remotos en computadoras o servidores distribuidos.
- SISTEMA EXTERNO: Aplicaciones implementadas en la Universidad que harán uso de los servicios Web de la fachada de la base de datos Ciudadano.
- SERVICIOS WEB: Aplicación que realiza un cometido y que puede formar parte de otros servicios para formar un servicio más completo. La comunicación hacia y desde los servicios Web se realiza con XML.
- SGML: Standard Generalizad Markup Language" o "Lenguaje de Marcación Generalizado". Consiste en un sistema para la organización y etiquetado de documentos, es decir, sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial. Este lenguaje fue normalizado por la Organización Internacional de Estándares (ISO) en 1986.

- SMTP: Simple Mail Transfer Protocol (SMTP), o protocolo simple de transferencia de correo electrónico. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos.
- SOAP: (Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM 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. SOAP es uno de los protocolos utilizados en los servicios Web.
- SUN: Sun Microsystems es una empresa informática del Silicon Valley, fabricante de semiconductores y software.
- TCP: Protocolo de Control de Transmisión (TCP, Transmission Control Protocol) es uno de los protocolos fundamentales en Internet. Muchos programas dentro de una red de datos compuesta por ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales enviarse un flujo de datos.
- UDDI: (Universal Description, Discovery and Integration) protocolo para publicar la información de los servicios Web. Permite a las aplicaciones comprobar qué servicios Web están disponibles.
- W3C: Comités responsables de la arquitectura y reglamentación de los servicios Web
- WSDL: (Web Services Description Language) Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- XML: (Extensible Markup Language) Lenguaje de marcas extensible, es un lenguaje de definición de documentos que permite una forma estándar de representar datos para su fácil intercambio.