UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
Facultad 9

Servicios Web para el Sistema de Reservación del Pase de la Universidad de las Ciencias Informáticas

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

AUTORES:

Ricardo Fonseca Pérez
Carlos Alberto Carvajal Vila

TUTOR:

Ing. Yoenis Pantoja Zaldívar

Ciudad de la Habana, Cuba
Julio del 2007

¿Por qué esta magnífica tecnología científica, que ahorra trabajo y nos hace la vida más fácil, nos aporta tan poca felicidad? La repuesta es esta, simplemente: porque aún no hemos aprendido a usarla.

"Albert Einstein"

DEDICATORIA

Dedicamos este trabajo a nuestros padres, familiares y amigos por su apoyo y preocupación en todo momento a lo largo de estos años de estudios, alegrías y tristezas. También para los buenos compañeros que quedaron en el camino, que sepan que aunque hoy no estén aquí, también ayudaron a que hayamos alcanzado nuestra meta.

AGRADECIMIENTOS

Llegue este profundo agradecimiento a nuestros padres por guiarnos en todo momento por el camino correcto, alentándonos a seguir adelante.

Agradecemos al tuto que supo guiarnos y apoyarnos en el desarrollo de este trabajo, brindándonos sus experiencias y sus consejos.

A todos nuestros compañeros de aula por aconsejarnos en los buenos y malos momento por lo que hemos pasado a lo largo de estos años de estudio.

A los amigos y hermanos que contribuyeron a que de una forma u otra la vida universitaria fuese una experiencia inolvidable.

Le agradecemos especialmente a nuestra comandante en jefe “Fidel Castro Ruz” por darnos la posibilidad de formarnos como futuros profesionales de esta Revolución Socialista y de esta forma aportar al desarrollo de la misma.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al departamento de informatización de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Carlos Alberto Carvajal Vila,

Ricardo Fonseca Pérez

_____

Yoenis Pantoja Záldivar

_____

DATOS DE CONTACTO

Síntesis del Tutor: Ing. Yoenis Pantoja Záldivar
Jefe de la Disciplina de Programación
Facultad 9 UCI

Síntesis del Asesor: Ana Luisa Vigó Mitjans
Licenciada en Educación Especialidad Lengua Inglesa
Profesora Auxiliar
Facultad 9 UCI

OPINIÓN DEL TUTOR

Trabajo de Diploma **Servicios Web para el Sistema de Reservación del Pase de la Universidad de las Ciencias Informáticas**

Como tutor del Trabajo de Diploma *"Servicios Web para el Sistema de Reservación del Pase de la Universidad de las Ciencias Informáticas"*, luego de haber culminado la realización del mismo, considero que los autores Ricardo Fonseca Pérez y Carlos Alberto Carvajal Vila, han desarrollado un conjunto de habilidades que les permitirán darle solución adecuadamente a cualquier tipo de necesidad de informatización que se presente en su vida profesional.

Su desempeño ha manifestado que han desarrollado un valioso nivel de asimilación de nuevas metodologías, llegando a alcanzar conocimiento y capacidad para la toma de decisiones correctas.

Durante la realización del presente trabajo los estudiantes han demostrado un alto grado de responsabilidad ante el cumplimiento en tiempo de las tareas que se les programaron. Han trabajado organizadamente dando muestras de poseer responsabilidad y compromiso en la realización de su tesis. La independencia, la optimización, la originalidad y la organización han sido cualidades dignas de destacar a lo largo de la realización del trabajo realizado. Los estudiantes manifestaron laboriosidad a lo largo del cumplimiento de las tareas programadas, logrando resultados satisfactorios a pesar de disponer de poco tiempo de desarrollo y de no constar con la merecida atención por parte del cliente.

Por otra parte, el elemento investigativo del documento, estuvo desde el inicio muy bien orientado y estructurado, basado en la bibliografía actualizada que se consultó. Cada contenido se ha expuesto con claridad y aporta grandes conocimientos al lector.

Por todo lo anteriormente planteado, considero que los diplomantes están aptos para ejercer como Ingenieros Informáticos; y propongo al tribunal que se le otorgue al Trabajo de Diploma la máxima calificación.

Ing. Yoenis Pantoja Zaldívar
Tutor

RESUMEN

El desarrollo de la informática y las comunicaciones en el mundo esta en constante cambio y búsquedas de soluciones optimas de acuerdo a las necesidades del hombre actual. Donde estas ayudan a mejorar el funcionamiento de las empresas, organismos y junto con ellos el mundo.

Actualmente la tendencia para elaborar aplicaciones informáticas está haciendo énfasis hacía el incremento de la interoperabilidad y la creación de aplicaciones multiplataforma. Es por ello que el proyecto se enmarca en el estudio y preparación de la aplicación web de reservación del pase para que la universidad pueda aplicar la arquitectura SOA. Conjuntamente definir y estudiar conceptos referentes a estos temas antes mencionados sin dejar de analizar, diseñar e implementar, Web Services que mejoren la cálida del servicio y la eficiencia, dirigidos precisamente a la gestión de reservación, reportes, la autenticación, la configuración de roles, además de permitir configurar y conocer información referente a la rutas, puntos de salida.
Permitiendo conocer el costo y beneficios que nos brinda el desarrollo de este proyecto para la universidad.

INDICE

TABLAS Y FIGURAS

Capitulo 3 Tablas y Figura

Capitulo 4 Tablas y Figuras

Capitulo 5 Tablas y Figura

INTRODUCCIÓN

En las últimas décadas, las aplicaciones informáticas de gestión de información están encaminadas cada vez más hacia una infraestructura de soporte integrado entre las operaciones de las empresas y sus clientes. El camino para llegar hasta este punto no ha sido fácil. Se ha aprendido de los errores y aciertos en la industria. El resultado de este proceso, ha sido la creación y mantenimiento de un número considerable de aplicaciones en el interior de las entidades, cada una responsable de sus propias tareas.

Los negocios cada vez exigen crear aplicaciones más complejas, con menos tiempo y presupuesto que antes. Crear estas aplicaciones, requiere en muchos casos de funcionalidades ya antes implementadas como parte de otros sistema.

La Universidad de las Ciencias Informáticas mantiene, en su estructura de aplicaciones, un diseño distribuido de sistemas de gestión de información a través de la Web interna. El crecimiento de dichos mecanismos hace imprescindible la puesta en marcha de otros que garanticen un óptimo aprovechamiento de los recursos y tecnologías ya planteadas.

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

El Sistema de Reservación del Pase de la Universidad de las Ciencias Informáticas (UCI) forma parte de esta red de aplicaciones Web que necesita de esta solución. La política a seguir está sustentada en futura puesta en marcha de una Arquitectura Orientada a Servicios (SOA) como mecanismo de integración entre los sistemas que coadyuvan en el funcionamiento y flujo de la información de la entidad.

La aplicación actual se distingue por una actividad estructurada de funcionalidades fijas, que como única utilidad tienen la solución de necesidades propias. Esto trae como consecuencia las siguientes disyuntivas:

- Funcionalidades replicadas por todos los sistemas de gestión de la entidad.
- Dificultad de migración de los sistemas internos. Al haber múltiples conexiones desde sistemas que dependen de estos para su funcionamiento.
- Al no haber una estrategia de integración de aplicaciones, se generan múltiples puntos de falla, que pueden detener la operación de todos los sistemas muy fácilmente.
- El inconveniente final es una pobre respuesta al cambio. Las aplicaciones siguen siendo concebidas desde un principio como islas independientes.

Luego del análisis, se define como **situación problémica** la no existencia de una arquitectura estándar que posibilite la integración de todas y cada una de las aplicaciones que se desarrollan para apoyar a la construcción de la *Ciudad Digital*, entre las que se encuentra el Sistema de Reservación del Pase. A raíz de esto surge la necesidad de realizar un replanteamiento de las aplicaciones existentes, enfocándolas hacía las bases de una Arquitectura Orientada a Servicios (SOA) estableciéndose como **problema** lo siguiente:

> ¿Cómo preparar el Sistema de Reservación del Pase en la UCI favoreciendo una posterior implementación de la arquitectura SOA en la universidad?

En correspondencia con lo formulado, el **objeto de estudio** estará enmarcado en la implementación de servicios Web *para sistemas de gestión de información*. Basándose en los patrones de la *Arquitectura Orientada a Servicios (SOA).*

A la hora de definir el **campo de acción** se decidió enfocar este trabajo hacia como crear los servicios web necesarios para el sistema de reservación del pase en la universidad de las ciencias informáticas.

El **objetivo general** de la investigación es llegar a implementar *los servicios web para el Sistema de Reservación del Pase de la Universidad de las Ciencias Informáticas*. Para dar cumplimento a lo formulado, se proponen las siguientes **tareas**:

- Identificación de todas las funcionalidades que brinda actualmente el Sistema de Reservación del Pase de la UCI.
- Desarrollo del análisis y el diseño de los servicios y funcionalidades identificados en el sistema actual.
- Selección de la metodología, arquitectura, estándares y herramientas para dar cumplimento al problema a resolver.
- Construcción de un mecanismo de Servicios Web para las funcionalidades identificadas en el Sistema de Reservación del Pase de la UCI.
- Realización de pruebas de adaptabilidad al sistema final.

Como **idea a defender** se formula: Desarrollo de un mecanismo de Servicios Web para el Sistema de Reservación del Pase de la UCI que logrará estandarizar la expansión y desarrollo de aplicaciones de gestión bajo los patrones de una Arquitectura Orientada a Servicios (SOA).

Con la puesta en marcha de este diseño funcional se obtendrá como **resultados** un proceso sistémico, que mejorará en amplio sentido, la forma de manejar la complejidad y la constante mejora de los servicios actuales y futuros de la aplicación de Reservación del Pase. Ya que el empleo de SOA y las ventajas que brinda, son la mejor alternativa disponible para construir sistemas altamente escalables, y se asegura con ello su interoperabilidad.

Como todo proceso investigativo se utilizaron métodos para el desarrollo de la investigación, se tuvieron en cuenta los métodos de investigación científica, que presentamos a continuación

Métodos Teóricos:

- Inductivo – deductivo: Se realiza un análisis de todo el proceso de la publicación de web service y el consumo de los mismo de forme detalla y precisa, cómo se debe de realizar este proceso. Sin dejar de mencionar todos los demás procesos que soporta la web para reservar el pase en la UCI.
- Analítico – sintético: analizar teóricamente el proceso de reservación del pase en la universidad y resumirlo.
- Modelación: crear modelos a través de la metodología RUP.

Métodos Empíricos:

- Observación: Se realizaron visitas al departamento de informatización de la UCI para observar los procesos, alcanzando el entendimiento de dichos procesos de la aplicación Web.
- Entrevista: Se realizan entrevistas al personal asignado por el departamento de informatización, que confecciono el sistema de reservación del pase, con el objetivo de precisar el problema a resolver.
- Estudiar y documentarse sobres el tema de los servicios web y su desarrollo en el mudo para poder aplicar esta tecnología en el proyecto.

El trabajo está estructurado en 5 capítulos:

El *primer capítulo* describe el objeto de estudio, se expone una valoración del estado del arte y se analizan las tendencias y tecnologías actuales referentes a las diferentes áreas del conocimiento del tema de la Arquitectura Orientada a Servicios en aplicaciones Web.

En el *segundo capítulo* se fundamenta el uso de las herramientas, lenguajes y metodologías a utilizar.

El *tercer capítulo* refleja la descripción de los principales procesos involucrados en el objeto de estudio. Se definen los conceptos en un Modelo del Dominio para capturar correctamente los requisitos y poder construir un sistema consistente, se enumeran los requisitos funcionales y no funcionales que debe tener el sistema que se propone. Además incluye las descripciones de los actores y casos de uso del sistema, así como los diagramas que representan a estos últimos.

El *cuarto 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 clases persistentes, del modelo de datos, de componentes y despliegue.

El *quinto y último capítulo* describe la estimación de costos del sistema propuesto y sus beneficios.

Capítulo 1
Fundamentación Teórica

1.1. Introducción

En este capítulo se hace una descripción general del objeto de estudio con el apoyo de las componentes de las diferentes áreas del conocimiento se expone una valoración del estado del arte y se analizan las tendencias y tecnologías actuales referentes al tema de la Arquitectura Orientada a Servicios en aplicaciones Web.

1.2. Conceptos asociados al dominio del problema

1.2.1. Arquitectura Orientada a Servicios (SOA)

La Arquitectura Orientada a Servicios (SOA) define los servicios de los cuales estará compuesto el sistema, sus interacciones, y con qué tecnologías serán implementados. Las interfaces que utiliza cada servicio para exponer su funcionalidad son gobernadas por contratos que definen claramente el conjunto de mensajes soportados, su contenido y las políticas aplicables.

Un servicio debe ser una aplicación completamente autónoma e independiente. A pesar de esto, no es una isla, porque expone una interfaz de llamado basada en mensajes, capaz de ser accedida a través de la red. Generalmente, los servicios incluyen tanto lógica de negocio como manejo de estado (datos) relevantes a la solución del problema para el cual fueron diseñados. La manipulación del estado es gobernada por las reglas de negocio.

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.

Esta arquitectura no se funda en la idea de cualquier servicio en general, comunicado de cualquier manera, sino que más específicamente va de la mano con la expansión de los Web Services.

Otros sistemas interactúan con el Web Service 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.

Vale la pena destacar la forma en la cual este estilo de arquitectura orientada a servicios redefine los modelos de ORPC (Llamadas a Procedimientos de Objetos Remotos) propios de las arquitecturas orientadas a objetos y componentes, y al hacerlo establece un modelo en el que es casi razonable pensar que cualquier entidad computacional podría llegar a conversar o a integrarse con cualquier otra.

Lo que hace diferentes a los Web Services de otros mecanismos de RPC como Java RMI, CORBA o DCOM es que utiliza estándares de la Web para los formatos de datos y los protocolos de aplicación. Esto permite que las aplicaciones ínter operen con mayor libertad, dado que las organizaciones ya seguramente cuentan con una infraestructura activa de HTTP y pueden implementar tratamiento de XML y SOAP en casi cualquier lenguaje y plataforma. **(1)**

La Arquitectura Orientada a Servicios es un patrón en el que los recursos están interconectados en red y se conciben como servicios accesibles por terceros a través de una interfaz estándar. 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. **(2)**

1.2.2. SOAP

SOAP es el acrónimo de Simple Object Access Protocol, es decir protocolo simple de acceso a objetos. SOAP es un protocolo ligero de mensajes XML que se usa para codificar la información de los mensajes de petición y respuesta de los SW que se envían a través de una red. Los mensajes SOAP son independientes de los sistemas operativos y de los protocolos, y pueden ser transportados usando una variedad de protocolos Internet, incluyendo SMTP, y HTTP. Para poner un sencillo ejemplo si nos abstraemos y pensamos en el uso de servicios Web a la manera de la programación orientada a objetos, usar un Servicio Web seria como usar una clase, accediendo a sus métodos y atributos. Para usar una función especifica debemos instanciarla, es decir llamarla pasando por referencia valores o parámetros, luego esperamos un resultado, si es que existe, y en este caso ese es el rol que juega SOAP el de describir esta operación, llamar a la función, pasar los parámetros requeridos y describir cómo será la respuesta y cómo la recibiremos.

1.2.3. Algunas de las Ventajas de SOAP

- No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.

- No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

- No está atado a ninguna infraestructura de objeto distribuido La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.

- Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.

- Permite la interoperabilidad entre múltiples entornos: SOAP se desarrollo sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del back-end ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP. **(3)**

1.2.4. Servicios Web

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

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

- -Un formato que describa la interfaz del componente (sus métodos y atributos) basado en XML. Por lo general este formato es el WSDL (Web Service Description Language), Lenguaje de descripción de servicios Web.

- -Un protocolo de aplicación basado en mensajes y que permite que una aplicación interaccione (use, instancia, llame, ejecute) al servicio. Por lo general este protocolo es SOAP (Simple Object Access Protocol), Protocolo Simple de Acceso a Datos.

Un servicio Web puede ser usado internamente por una aplicación o ser publicado en Internet. Estos servicios permiten la ejecución de sus funcionalidades sin importar la plataforma, sistema operativo, o lenguaje en el cual estén implementados.

Gracias a los servicios Web podemos hacer que sistemas heterogéneos trabajen conjuntamente como una sola aplicación computacional. Estos servicios constituyen una potente herramienta para el desarrollo de aplicaciones distribuidas. **(4)**

1.2.5. XML

El Extensible Markup Language (XML), con todas las tecnologías relacionadas, representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes.

Así pues, el XML juega un papel importantísimo para los servicios Web en la actualidad, es la tecnología que permitirá compartir la información de una manera segura, fiable, fácil. Además, XML permite al programador y los soportes dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, porque algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello. **(5)**

1.2.6. Protocolos de transferencias

Hasta este momento hemos mencionado conceptos y términos que están reflejados en el contexto informático de acuerdo a la situación planteada en la introducción, pero no hemos tenido en cuenta los protocolos de trasferencias que son la vía principal de desarrollo de todas las aplicaciones informáticas por lo que a continuación expresamos varios de estos:

HTTP

Un protocolo de transporte que se encargue de transportar los mensajes por Internet. Por lo general este protocolo de transporte es HTTP (Hiper-Text Transport Protocol), Protocolo de Transferencia de Hipertexto, que es exactamente el mismo que usamos para navegar por la Web.

SMTP

(Simple Mail Transfer Protocol) o *Protocolo Simple de Transferencia de Correo Electrónico* es un conjunto de reglas que rigen el formato y la transferencia de datos en un envío de Correo Electrónico (e-mail). **(6)**

TCP/IP

TCP/IP son las siglas de Transmission Control Protocol/Internet Protocol, el lenguaje que rige todas las comunicaciones entre todos los ordenadores en Internet. TCP/IP es un conjunto de instrucciones que dictan cómo se han de enviar paquetes de información por distintas redes.

IP, Internet Protocol, es la especificación que determina hacia dónde son encaminados los paquetes, en función de su dirección de destino. TCP, o Transmission Control Protocol, se asegura de que los paquetes lleguen correctamente a su destino. Si TCP determina que un paquete no ha sido recibido, intentará volver a enviarlo hasta que sea recibido correctamente. **(7)**

1.2.7. Seguridad de los Web Services

Actualmente, los web services 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. Los desarrollos actuales no ayudan a la cooperación entre las empresas ya que no hay ningún estándar establecido sobre las técnicas de seguridad. Debido a la tecnología que es usada por los web services, y en concreto al uso de SOAP, las técnicas de seguridad convencionales que se han venido usando en Internet, ya no son suficientes. Con SOAP, cada mensaje simple que se intercambia realiza múltiples saltos y es ruteado a través de numerosos puntos antes de que alcance su destino final. Es por ello que los web services necesitan tecnologías que protejan los mensajes desde el principio hasta el final. Existen un conjunto de técnicas que se pueden usar para garantizar la seguridad a nivel de mensaje. Estas son:

- Encriptación XML: Evita que los datos se vean expuestos a lo largo de su recorrido.
- Firma Digital XML: Asocia los datos del mensaje al usuario que emite la firma, de modo que este usuario es el único que puede modificar dichos datos.
- XKMS y los Certificados: XKMS (XML Key Management Specification) define web services que se pueden usar para chequear la confianza de un certificado de usuario.
- SAML y la Autorización: SAML (Security Assertion Mark-up Language) hace posible que los web services intercambien información de autentificación y autorización entre ellos, de modo que un web service confíe en un usuario autentificado por otro web service.
- Validación de datos: Permite que los web services reciban datos dentro de los rangos esperados.

Además, también hay técnicas que permiten mantener la seguridad a otros niveles. La seguridad en UDDI permite autentificar todas las entidades que toman parte en la publicación de un web service: proveedor, agente y consumidor del servicio. De este modo, nadie podrá registrar servicios en el papel de un proveedor o hacer uso de ellos sin contar con los permisos adecuados. **(8)**

Objeto de Estudio

1.3. Descripción General

En la UCI se presta servicio de reservación de pase planificando el mismo de una manera organizada mediante la aplicación web, atendiendo a la disponibilidad de tiempo y de los recursos que la universidad tiene a su disposición para realizar esta operación. Donde los principales cliente de este servicio son los estudiantes que mediante la aplicación realizan su solicitud de acuerdo a sus necesidades. Además en otro sentido tenemos los responsables de dar cumplimiento a esta operación, obteniendo de la aplicación información relevante que le permitirá conocer el cómputo necesario que la universidad necesitará para brindar el servicio a los estudiantes.

1.3.1. Descripción del Dominio del problema

La UCI es un centro de estudios de nuevo tipo. Fue creada en septiembre del 2002 en la Ciudad de la Habana como uno de los mayores proyectos de la Revolución. Su principal reto es la formación de miles de jóvenes de todo el país como futuros profesionales en la rama de la informática.

Teniendo una matricula que supera los 9 000 estudiantes. En ella se garantiza una preparación académica de gran calidad, así como el entrenamiento profesional de los estudiantes, por medio de la participación de forma directa y sistemática en la producción. Se brinda una gran flexibilidad en los diseños curriculares, como nueva alternativa para la formación y capacitación del capital humano vinculado a la informática.

Cuenta con la mayor residencia estudiantil, teniendo de esta manera implementada unas de las redes informática más grande del país, que aprovecha al máximo de su capacidad para lograr mayor comunicación entre el personal y las actividades a realizar.

Por este medio la máxima dirección de la UCI logra informar a todos sus trabajadores y dirigir a los estudiantes en sus actividades. Teniendo en cuanta que otra vía seria compleja para lograr el correcto funcionamiento de la institución. La universidad cuenta con servicios informáticos que la ayudan en este proceso como:

- Los canales interno.
- Los servicio de la Intranet.
- Las aplicaciones docentes de las facultades.
- Los servicios de la Biblioteca Digital.
- El Servicio de reservación de pase de los estudiantes.
- Moodle

1.4. Situación en que se encuentran las aplicaciones en la UCI

Todas las aplicaciones de la uci se rigen por una base de datos central, donde esta es la que almacena toda la información referente al personal de la universidad ya sea un estudiante, o un trabajador de otra índole, de acuerdo a las especificaciones de la Web.

Estas aplicaciones que trabajan con este flujo de información cada una lo gestiona de forma autónoma, contribuyendo a que ocurra una sobre carga de operaciones con la base dato puesto que tenemos barias aplicaciones solicitando servicios o información de formas independiente, donde al final los resultado a devolver serian los mismo, además de estar reutilizando código constantemente, *problemas* :

- Demora de las aplicaciones al procesar las peticiones del cliente.
- Sobrecarga de los Servidores.
- Inconformidad del cliente para con el servicio o la entidad.
- No respuesta de la aplicación en mucho de los casos.

1.5. Flujo actual de los procesos

Los procesos que intervienen en la Reservación del Pase ocurren de la siguiente manera:

El estudiante utilizando la tecnología de las informáticas y las comunicaciones que existen en la universidad para establecer una conexión con el servidor donde esta montada la aplicación y la maquina cliente, desde ese instante comienza el flujo de información establecido entre el estudiante y la web, mediantes las funcionalidades específicas de la aplicación, que se exponen a continuación, con cierto grado de jerarquía.

Generales: funciones donde todos los usuario pueden acceder sin restricciones.

- Autenticarse

Usuario Estudiante

- Reservar Ida.
- Reservar Regreso.
- Cancelar Reservación
- Modificar Reservación

Usuario Administrador

- Impresión de Boletines
- Buscar Estudiantes
- Reporte Por Rutas
- Reporte Por Año
- Reporte Por Facultades
- Agregar Usuario
- Listar Usuarios
- Eliminar Usuario

- Listado de Rutas
- Agregar Ruta
- Eliminar Ruta
- Modificar Horario
- Insertar Punto de Salida
- Editar punto de Salida
- Eliminar Punto de Salida

Todas estas operaciones están montadas sobre la web, de ahí que surge el problema abordado en la situación problémica planteada en la introducción. En el instante preciso que la dirección de informatización de la universidad decida migrar para una plataforma libre, tendríamos que cambiar la aplicación íntegramente. Además de no poder reutilizar el código que brinda cada aplicación informática puesta en marcha. Para obtener un mejor funcionamiento con eficiencia y calidad.

Por lo que planteamos que todo proceso que se lleve a cabo en la universidad de forma automatizada se realice a través de servicios web que favorezcan la integración con las demás aplicaciones aprovechando las ventajas ofrecidas por los servicios. Pudiendo mantener las interfaces de las aplicaciones ya existentes. Además de esta manera estaríamos tributando hacia la tendencia del mundo actual, al igual que las grandes empresas que están buscando para elevar su calidad y eficiencia desarrollar sus sistemas a través de servicios web.

1.6. Análisis de otras soluciones existentes

En el mundo existen aplicaciones que prestan servicios de este tipo e incluyen otros, donde la gran mayoría están inclinados por los sitios web difundidos por todo internet y de esta manera buscan una red de usuario colosal para hacer sus operaciones. Ejemplo de esto tenemos la compañía *Rail Europe* manteniendo en funcionamiento una aplicación web oficial de los ferrocarriles de europeos. **(9)**

En *SANTA FE ARGENTINA* existe *JR SOFTWARE* Transporte de Pasajeros, es un sistema orientado a aquellas empresas que cuentan con una flota de colectivos, aviones, etc. Con terminales en distintos puntos del país o del mundo. Este sistema permite administrar: Reservas de asientos mostrando un plano de la unidad o vehículo con los lugares disponibles al momento de la venta, Compras y Gastos por unidad o vehículo, Caja Diaria, Turnos de los empleados en ventanilla, Unidades de transportes, Plan de cuentas, Configuración de viajes, etc. Entre otras prestaciones, cabe mencionar que con JR SOFTWARE para el Transporte de Pasajeros podrá llevar la cuenta corriente de sus proveedores. **(10)**

En nuestro país existen también aplicaciones de este tipo que contribuyen al desarrollo del país, entre las cuales se destaca la Aero-Caribbean poniendo a disposición de los usuario la tecnología web para consultar y aprovechar los viajes y ofertas que brinda la Empresa. **(11)**

Teniendo en cuenta lo expuesto y analizando sus especificaciones llegamos a la conclusión de que ninguna de las soluciones mencionadas puede resolver nuestro problema ya que ellas están echas con un fin especifico y seria muy complejo adecuar nuestra situación a estas soluciones existentes.

1.7. Conclusiones

En este capítulo se detallaron las condiciones y problemas que rodean el objeto de estudio a través de los conceptos y definiciones planteadas, se determinaron las condiciones específicas que tributan al problema y sobre la base del cual se obtuvieron los objetivos generales. Además de que se mostraron algunas aplicaciones similares a la nuestra que hoy se están aplicando para gestionar distintos tipos de reservaciones.

Capítulo 2
Tendencias y Tecnologías Actuales a Desarrollar

2.1. Introducción

En este capitulo se fundamentará el uso de las herramientas, lenguaje y tecnologías que se utilizaran para el posterior desarrollo de este trabajo. Pero para esto primero se ha de conocer el concepto de ingeniería del software y a que se están refiriendo cuando oye hablar de la misma.

Abordará un poco acerca del “Rational Rosse” ya que esta es la herramienta case líder en el mundo del modelado de sistemas de software. La misma se apoya en RUP como proceso unificado de desarrollo de software y utiliza el lenguaje unificado de modelación (UML) para la confección de sus diagramas a la hora de modelar un sistema.

Se explicará que es RUP, UML y en que se basa el proceso de desarrollo de software. Además de exponer las características, ventajas y desventajas del PHP como lenguaje de programación que será utilizado a la hora de implementar los servicios web.

Al finalizar este capitulo también se expondrán dos razones que justifican la decisión que llevó a utilizar php a la hora de implementar los servicios web y no otro lenguaje de programación como java o C#.

2.2. Que es la Ingeniería del Software

La ingeniería del software es una tecnología que nos indica como debemos construir técnicamente un software y como hacerlo de una manera económica, fiable y que funcione correctamente.

La Ingeniería de Software es una tecnología multicapa en la que, según Pressman, se pueden identificar:

Los métodos que son los que indican cómo construir técnicamente el software, el proceso que no es más que el fundamento de la Ingeniería de Software ya que es la unión que mantiene juntas las capas de la tecnología.

Las herramientas que brindan el soporte ya sea automático o semiautomático para el proceso y los métodos.

2.3. Herramienta case "Rational Rosse"

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico.

Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. Rational Rose utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones.

Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para qué éstos se hagan mínimos.
Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Rational Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que

contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades.

Es una herramienta generadora de código ya que se puede generar código en distintos lenguajes de programación a partir de un diseño realizado previamente en UML. Ahorrándole tiempo y esfuerzo a los programadores.

Además Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño. **(12) (13)**

2.4. Proceso de desarrollo del software

El Proceso de Desarrollo de Software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

Por lo tanto, las piedras angulares del proceso de desarrollo del software son: el proyecto, las personas y el producto; siendo las características del cliente, el entorno de desarrollo y las condiciones del negocio, elementos que influyen en el proceso.

El conocimiento y la experiencia que crea y sostiene la evolución del producto lo tienen las personas. Ellas también financian, se benefician, lo prueban y planifican.

Sin un personal competente y experimentado es imposible crear productos competitivos que satisfagan las necesidades de los clientes. Y en esto es precisamente donde nos beneficia el proceso de desarrollo del software.

Además, el modo en que se organiza y gestiona un producto (factibilidad del proyecto, definición del equipo que lo desarrolló, identificación y análisis de riesgos que implican su realización, planificación, nivel de entendimiento de lo que se está haciendo y sensación de que se avanza en el cumplimiento) afectan a las personas involucradas en su realización.

Un proyecto es un elemento organizativo a través del cual se gestiona el desarrollo del software. Un proyecto de desarrollo obtiene un versión de un producto que contiene modelos, código fuente, documentación y un ejecutable.

Este producto irá evolucionando durante el proceso de desarrollo desde un proyecto inicial o innovador (prototipo inicial) hasta convertirse en un release del proyecto. **(14)**

2.5. El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta

El lenguaje unificado de modelación (UML) ofrece la oportunidad de modelar, construir y documentar los elementos que conforman un sistema o software orientado a objeto esto se puede lograr a través de sus distintos diagramas.

Uno de los objetivos principales de la creación de UML era la necesidad que existía de posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado.

Para lograr el cumplimiento de ese primer objetivo era necesario definir una notación y una semántica común para todos los desarrolladores de software mundiales. Y precisamente esto fue lo que se logró con la aparición de UML.

Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación. UML a través de sus diagramas permite llevar a cabo la representación gráfica de un conjunto de elementos, que facilitan visualizar un sistema desde diferentes perspectivas.

El UML no solo se puede utilizar para modelar sistemas de software de forma estandarizada. También se pueden modelar sistemas de hardware e incluso organizaciones del mundo real.

2.6. El Proceso Unificado de Desarrollo de Software (RUP) como base en el desarrollo de la solución

El RUP es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML), forman la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas de software orientados a objetos.

Sus principales características son:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software:
    - Desarrollo iterativo.
    - Administración de requisitos.
    - Uso de arquitectura basada en componentes.
    - Control de cambios.
    - Modelado visual del software.
    - Verificación de la calidad del software.

RUP Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. El RUP incluye Artefactos y Roles, para su organización.

Los artefactos son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc. El Rol es el papel que desempeña una persona en un determinado momento, teniendo en cuenta que una persona puede desempeñar distintos roles a lo largo del proceso.

RUP divide el proceso de desarrollo en ciclos, obteniéndose un producto final, al final de cada ciclo. Cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante. **(15)**

2.7. Otros Proceso de Desarrollo del Software, XP y FDD

2.7.1. Extreme Programming (XP)

Mientras que el RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

XP intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Este representante debería estar en condiciones de contestar rápida y correctamente a cualquier pregunta del equipo de desarrollo de forma que no se retrase la tomar de decisiones, de ahí lo de competente.

Extreme Programming define UserStories como base del software a desarrollar. Estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del software, que no solo se limitan a la GUI si no también pueden describir el modelo, dominio, etc. A partir de las UserStories y de la arquitectura perseguida se crea un plan de releases (dejaré el término en inglés, pues las habituales traducciones al castellano, liberación o entrega del software, no son de mi agrado, supongo que son cosas de vivir en el extranjero) entre el equipo de desarrollo y el cliente.

Para cada release se discutirán los objetivos de la misma con el representante del cliente y se definirán las iteraciones (de pocas semanas de duración) necesarias para cumplir con los objetivos de la release. El resultado de cada iteración es un programa que se transmite al cliente para que lo juzgue. En base a su opinión se definen las siguientes iteraciones del proyecto y si el cliente no esta contento se adaptará el plan de releases e iteraciones hasta que el cliente de su aprobación y el software este a su gusto.
Junto a los UserStories están los escenarios de pruebas que describen el escenario contra el que se comprueba la realización de las UserStories. UserStories y casos de pruebas son la base sobre la que se asienta el trabajo del desarrollador.

Como primer paso de cada iteración se escribirán las pruebas, de tal forma que puedan ser ejecutadas automáticamente, de manera que pueda comprobarse la corrección del software antes

de cada release. Esto es de vital importancia en XP debido a su apuesta por las iteraciones cortas que generan software que el cliente puede ver y por la refactorización para mejorar el código constantemente, que hacen más que deseable una cantidad considerables de test lo más automatizables posible. Así pues, la funcionalidad concreta del software solo se escribe cuando las pruebas para su corrección estén preparadas.

En XP se programará solo la funcionalidad que es requerida para la release actual. Es decir, una gran flexibilidad y capacidad de configuración solo será implementada cuando sea necesaria para cumplir los requerimientos de la release. Se sigue un diseño evolutivo con la siguiente premisa: conseguir la funcionalidad deseada de la forma más sencilla posible. De ahí una variación educada del famoso *KISS* (*Keep It Simple Stupid*), *Keep things as simple as possible* (mantén las cosas tan sencillas como sea posible). Este diseño evolutivo hace que no se le de apenas importancia al análisis como fase independiente, puesto que se trabaja exclusivamente en función de las necesidades del momento. **(16)**

2.7.2. FDD

FDD es un proceso diseñado por Peter Coad, Erich Lefebvre y Jeff De Luca y se podría considerar a medio camino entre RUP y XP, aunque al seguir siendo un proceso ligero (en mi opinión, si tenemos que distinguir entre pesado/ligero) es más similar a este último.

La Metodología esta pensado para proyectos con tiempo de desarrollo relativamente cortos (menos de un año). Se basa en un proceso iterativo con iteraciones cortas (~2 semanas) que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar.

Los proyectos que siguen FDD se dividen en 5 fases:

1. Desarrollo de un modelo general
2. Construcción de la lista de funcionalidades
3. Plan de releases en base a las funcionalidades a implementar
4. Diseñar en base a las funcionalidades
5. Implementar en base a las funcionalidades

Las primeras tres fases ocupan gran parte del tiempo en las primeras iteraciones, siendo las dos últimas las que absorben la mayor parte del tiempo según va avanzando el proyecto, limitándose las primeras a un proceso de refinamiento.

El trabajo (tanto de modelado como de desarrollo) se realiza en grupo, aunque siempre habrá un responsable último (arquitecto jefe o jefe de programadores en función de la fase en que nos encontremos), con mayor experiencia, que tendrá la última palabra en caso de no llegar a un acuerdo. Al hacerlo en grupo se consigue que todos formen parte del proyecto y que los menos inexpertos aprendan de las discusiones de los mas experimentados, y al tener un responsable último, se asignan las responsabilidades que todas las empresas exigen.

En el proceso de implementar la funcionalidad también se contemplan como partes del mismo (en otros métodos se describen como actividades independientes) la preparación y ejecución de pruebas, así como revisiones del código (para distribuir el conocimiento y aumentar la calidad) e integración de las partes que componen el software.
FDD también define métricas para seguir el proceso de desarrollo de la aplicación, útiles para el cliente y la dirección de la empresa, y que pueden ayudar, además de para conocer el estado actual del desarrollo, a realizar mejores estimaciones en proyectos futuros. **(17)**

2.8. El lenguaje de programación PHP

En un lenguaje interpretado de alto nivel embebido en páginas HTML pero sus instrucciones son ejecutadas del lado del servidor. 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 reparan rápidamente.

El código se pone al día continuamente con mejoras y extensiones del lenguaje para ampliar las capacidades de PHP. Una de sus características más potentes es su soporte para un gran número de gestores de bases de datos.

PHP es la opción natural para los programadores que tienen Apache como servidor web, pero funciona igualmente bien con cualquier otra plataforma de Unix o de Windows.

Ventajas de PHP:

- Muy sencillo de aprender.
- Similar en gran parte de sus sintaxis a C y a PERL
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores.
- Se puede incrustar código PHP con etiquetas HTML y como ya se había hablado antes este código se ejecutará del lado del servidor.
- Soporte de acceso para un gran numero de bases de datos. Entre los que se encuentran InterBase, MySQL, Oracle, Informix, PosgreSQL, entre otros.
- La comprobación de que los parámetros son validos se hace en el servidor y no en el cliente (como se hace con javascript) de forma que se puede evitar tener que chequear que no se reciban solicitudes adulteradas.

  Además PHP viene equipado con un conjunto de funciones de seguridad que provienen de la interacción de órdenes dentro de una solicitud de datos.

- Permite transmitir vía HTTP, todo lo que se puede construir con el.

Desventajas de PHP:

- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y php.
- La orientación a objetos es aún muy deficiente para aplicaciones grandes.

2.9. Por que utilizar PHP y no otras opciones

- PHP no soporta directamente punteros, como el C, de forma que no existen los problemas de depuración provocados por estos.
- Se pueden hacer grandes cosas con pocas líneas de código. Lo que hace que merezca la pena aprenderlo.
- El código php es mucho más legible que el de PERL, todo el que haya programado PERL podrá corroborar esta afirmación.
- Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, gestión de un e-commerce, XML, creación de PDF ...)
- Al poderse encapsular dentro de código HTML se puede recoger el trabajo del diseñador gráfico e incrustar el código php posteriormente.
- Esta siendo utilizado con éxito en varios millones de sitios web.
- Hay multitud de aplicaciones php para resolver problemas concretos (weblogs, tiendas virtuales, periódico) listas para usar.
- Es multiplataforma, funciona en todas las plataformas que soporten apache.
- Es software libre. Se puede obtener en la web y su código esta disponible bajo la licencia GPL.

2.10. Por que hemos decidido usar PHP como lenguaje de programación a la hora de desarrollar nuestro trabajo

A parte de todas las ventajas expuestas con anterioridad que brinda PHP. Hay dos razones fundamentales por la que se escogió a la hora de implementar los servicios web.

Una de ellas es que la aplicación web que precede a este trabajo, y cuyas funcionalidades se quieren modelar a través de servicios web. Fue construida utilizando PHP como lenguaje de programación.

Por tanto al construir los servicios utilizando el mismo lenguaje se ahorraría mucho tiempo y esfuerzo. Reutilizando parte del código de la aplicación anterior y funciones que ya están implementadas.

Y la otra razón es porque PHP es software libre y como se conoce Cuba es un país bloqueado. Al que no se le permite el uso de tecnologías como el Visual Studio.Net para desarrollar aplicación utilizando el lenguaje de programación C#.

2.11. Patrón GRASP

GRASP es el acrónimo de General Responsibility Assignment Software Patterns. Una de las cosas más complicadas en Orientación a Objeto consiste en elegir las clases adecuadas y decidir como estas clases deben interactuar. Incluso cuando utilizamos metodologías rápidas como programación extrema (extreme programming) y centramos el proceso en el desarrollo continuo, es inevitable elegir cuidadosamente las responsabilidades de cada clase en la primera codificación y, fundamentalmente, en la refactorización (continual) de nuestro programa.

2.11.1. Bajo Acoplamiento

- ✓ Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿cuanto software podemos extraer de un modo independiente y reutilizarlo en otro proyecto?
- ✓ Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML.
- ✓ Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia.
- ✓ Como ejemplo (de mal diseño), en muchos diseños Web se puede ver como se crea un servlet base con capacidades de paginación y se hereda de él para construir los demás. La paginación es un servicio que se podría usar en aplicaciones no Web, por lo que es más adecuado mantener estas capacidades en clases externas.
- ✓ Otro ejemplo clásico se produce cuando se pasan los objetos relacionados con la capa de presentación a la capa de negocio (HttpRequest, HttpResponse).

2.11.2. Alta Cohesión

Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.
Ejemplos de una baja cohesión son clases que hacen demasiadas cosas.
En todas las metodologías se considera la refactorización. Uno de los elemento a refactorizar son las clases saturadas de métodos.
Ejemplos de buen diseño se producen cuando se crean los denominados "paquetes de servicio" o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia).

2.11.3. Experto

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema:

- ✓ Lógica de negocio
- ✓ Persistencia a la base de datos
- ✓ Interfaz de usuario

No tiene sentido considerar que una clase se debe escribir a si misma en base de datos o formatearse para presentarse en una página HTML por el hecho de poseer los datos. Estos son elementos estructuralmente distintos y deben considerarse desde una perspectiva distinta.

2.11.4. Creador

El patrón *Creador* guía la asignación de responsabilidades relacionadas con la creación de objetos (una tarea muy común). La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Teniendo como ventaja bajo acoplamiento logrando mayor mantenibilidad y reutilización.
Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando

1. B contiene a A
2. B es una agregación (o composición) de A
3. B almacena a A
4. B tiene los datos de inicialización de A (datos que requiere su constructor)
5. B usa a A.

El hecho de crear objetos tiene casuísticas particulares:

- Pool de Objetos
- Caches
- Instancias únicas

Estos casos son candidatos para la utilización de otros patrones más concretos (de diseño).
A la hora de crear objetos en distintos lenguajes hay que tener en cuenta sus peculiaridades. En Java por ejemplo:

- No confundir referencias y el valor referenciado a la hora de retornar objetos desde métodos.
- Conocer la peculiaridad de las String (cadenas constantes)
- Identificar problemáticas del recolector de basura y el uso de recursos del sistema
- Conocer el ámbito y naturaleza de distintos tipos de componentes
    - En servlets y JSPs pueden descargarse y estar activos en más de una máquina virtual.
    - Los servlets y JSPs son multi-thread a priory.
    - Los tags son reciclados en JSP (ojo con valores anteriores de atributos de clase).
    - Los EJB se desactivan.
    - etc...
- Y muchas cosas más... que la experiencia nos enseña **(18)**

2.12. Arquitecturas

2.12.1. Arquitectura en Capas

Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia mencionados como categorías mayores del catálogo, o, por el contrario, como una de las posibles encarnaciones de algún estilo más envolvente. En Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Instrumentan así una vieja idea de organización estratigráfica que se remonta a las concepciones formuladas por el patriarca Edsger Dijkstra en la década de 1960, largamente explotada en los años subsiguientes.

En algunos ejemplares, las capas internas están ocultas a todas las demás, menos para las capas externas adyacentes, y excepto para funciones puntuales de exportación; en estos sistemas, los componentes implementan máquinas virtuales en alguna de las capas de la jerarquía.

En otros sistemas, las capas pueden ser sólo parcialmente opacas. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas. El uso de arquitecturas en capas, explícitas o implícitas, es frecuentísimo; existen cerca de cien patrones que son variantes del patrón básico de capas. Patrones de uso común relativos al estilo.

Ventajas del estilo en capas

- ✓ Primero que nada, el estilo soporta un diseño basado en niveles de abstracción crecientes.
- ✓ Permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- ✓ En segundo lugar, el estilo admite muy naturalmente optimizaciones y refinamientos.
- ✓ En tercer lugar, proporciona amplia reutilización. Al igual que los tipos de datos abstractos.
- ✓ 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

Desventajas del estilo en capas

- ✓ No admiten un buen mapeo en una estructura jerárquica. Incluso cuando un sistema se puede establecer lógicamente en capas.
- ✓ Las consideraciones de performance pueden requerir acoplamientos específicos entre capas de alto y bajo nivel.
- ✓ A veces es también extremadamente difícil encontrar el nivel de abstracción correcto.
- ✓ Los cambios en las capas de bajo nivel tienden a filtrarse hacia las de alto nivel, en especial si se utiliza una modalidad relajada. **(19)**

2.12.2. Arquitectura Cliente Servidor

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como servidores, estos últimos responden a la demanda del cliente que la produjo.
Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es la Internet.
Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

¿Que es un Cliente?
Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente

¿Que es un Servidor?
Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

Características del Modelo Cliente/Servidor
En el modelo CLIENTE/SERVIDOR podemos encontrar las siguientes características:
1. El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
2. Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
3. Un servidor da servicio a múltiples clientes en forma concurrente.
4. Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

5. La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

6. Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes**.** Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.
También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.
Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

7. Además se constituye como el nexo de unión mas adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.

8. Designa un modelo de construcción de sistemas informáticos de carácter distribuido.

1. Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los
2. recursos de este host central y otros sistemas de la organización ponen a su servicio.

En conclusión, Cliente/Servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en TokenRing, Ethernet, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

2.13. Conclusiones

En este capítulo se ha abordado acerca de las herramientas y tecnologías más usadas a la hora de diseñar, desarrollar y documentar un software. Para esto se explicó en que se basa la ingeniería de software, y el proceso de desarrollo de software, así como la herramienta que se utiliza a la hora de realizar el modelado de un sistema cualquiera.

Se menciono a RUP como proceso de desarrollo que junto al UML como lenguaje unificado de modelado, conforman la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas de software. Conjuntamente con la exposición de las características de otras metodología que existen como es el caso de XP y FDD.

Y además se expusieron las características, ventajas y desventajas del lenguaje de programación PHP. Y las razones por las que se escogió el mismo para elaborar los servicios web necesarios.

Capítulo 3
Presentación de la solución propuesta

3.1. Introducción

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

3.2. Entorno donde trabajará el sistema

3.2.1. Descripción de los procesos del negocio

La mayoría de las aplicaciones de la UCI necesitan manipular los datos relativos a la identificación de las personas, para poder cada una, cumplir con sus objetivos específicos. Además de contribuir a la organización de la estancia de los estudiantes y profesores en la universidad.
Una de estas aplicaciones es la reservación del pase para los fines de semana, este servicio verifica la confidencialidad de los datos de todos los estudiantes que la solicitan. Datos como usuario, contraseña, dirección asía donde se dirige en la habana, parentesco con el estudiante, cual es la ruta a la que debe de dirigirse para efectuar la salida. Así mismo corrige y verifica la veracidad de los datos con debida consistencia en que están almacenados en la UCI. Ella posterior a esto reportar toda la ruta posible a cubrir por la universidad, reporta todos los estudiantes que reservaron, e incluye otros reportes.
Para el desarrollo de la aplicación en la actualidad se requiere que se realice un análisis que contribuya al establecimiento de *conjunto de servicios Web* a través de los cuales el producto siga brindando sus funcionalidades pero de una forma más eficiente y funcional.

3.2.2. Modelo del dominio

Teniendo en cuenta la descripción que brindamos en el *epígrafe 3.2.1* se puede inferir que la aplicación que sirve como soporte para efectuar el pase en la institución, no es un proceso de negocio puesto que no están bien definidos los flujos de los eventos, así como las funciones que se desarrollan en el ambiente o entorno que definimos, en el cual esta enmarcado el problema.
Esto sugiere utilizar un modelo de dominio que nos permite la representación visual de los conceptos u objetos significativos del entorno o área del problema.
Para ello le presentamos una tabla con los conceptos y descripciones que ayudarán a una mejor comprensión del problema.

Concepto	Descripción
Universidad	Institución donde se desarrolla el dominio de nuestra trabajo. La cual tiene una aplicación para gestionar el pase semanal en la misma.
Departamentos	Los distintos departamentos de la universidad entre los cuales están el departamento de trasporte y el departamento de informatización entre otros…
Departamento de Trasporte	Departamento que se encarga de gestionar todo el trasporte de la universidad, por lo que también se relaciona con la aplicación de reservación del pase haciendo uso de sus reportes.
Departamento de Informatización	Departamento que se encarga de la informatización de la universidad, por tanto entre sus funciones está administrar la aplicación utilizada para la reservación del pase en la misma.
Aplicación para la Reservación del Pase	Es la aplicación que permite que los estudiantes puedan gestionar los pases del fin de semana, de forma rápida. Y que se lleve a cabo la salida de pase de la universidad de forma organizada.
Reportes	Información que se puede obtener de la aplicación acerca de las reservaciones realizadas por los estudiantes. De los cuales existen 3 tipos…
Por Rutas	Es la información acerca de las reservación del pase echas para una ruta determinada.
Por Año	Es la información acerca de las reservación del pase echas por los estudiantes de un año determinado.
Por Facultad	Es la información acerca de las reservación del pase echas por los estudiantes de una facultad determinada.
Facultades	Es como está estructurada docentemente la universidad; en la actualizad existen 10 facultades.
Estudiantes	Representa el conjunto de estudiantes de la facultad, los que utilizan la aplicación para gestionar sus pases.
Decanato	Oficina donde se encuentra los directivos de la facultad: decano, vice decanos y la secretaría del decano.

Decano	Es el principal directivo de la facultad.
Secretaría	La secretaría del decano que es la encargada de imprimir los reportes de las reservaciones y repartir los boletines del pase a los estudiantes.

Tabla 3.1 Descripción de los Concepto del dominio del problema

3.2.3. Diagrama de clases del Modelo de Dominio

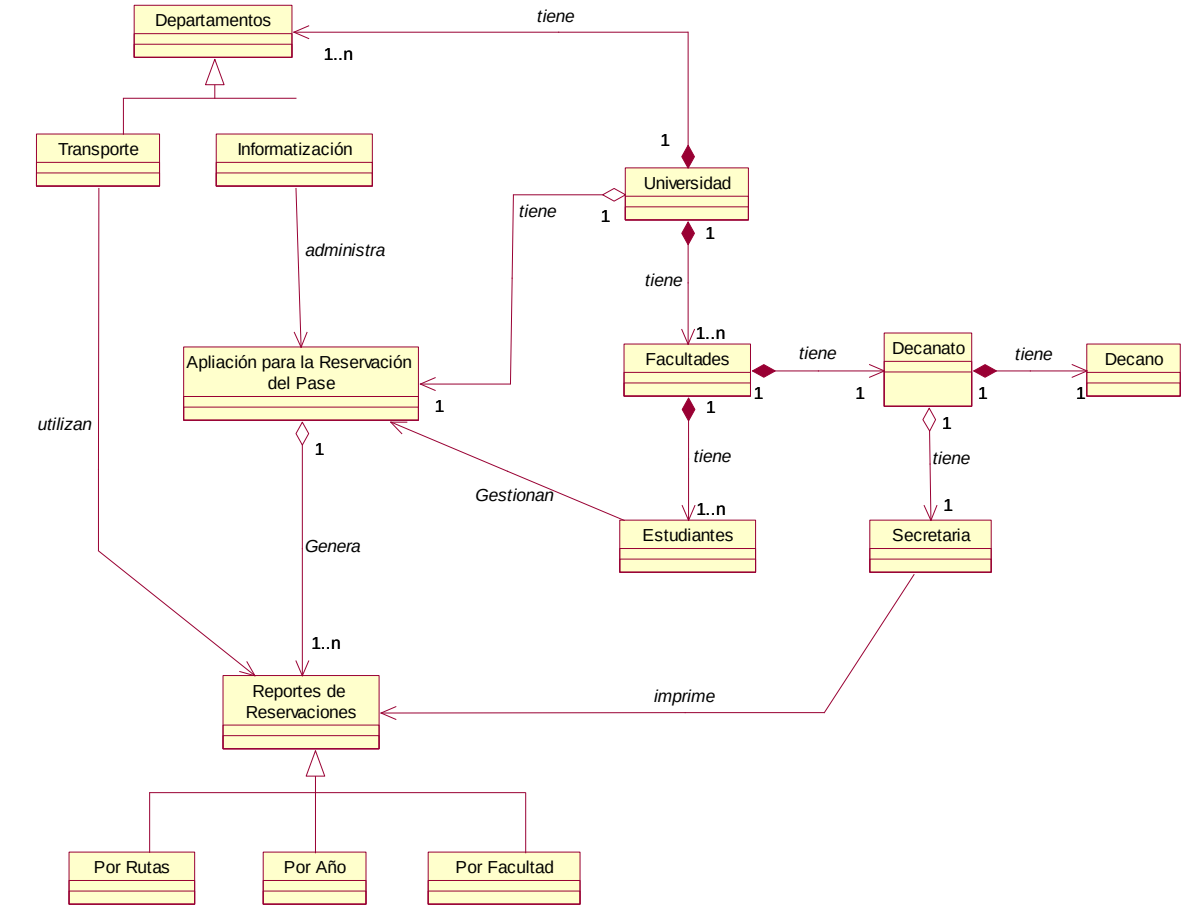


Figura 3.1 Modelo de Dominio

3.3. Requerimientos Funcionales

De acuerdo con las especificaciones del cliente y un estudio realizado a la aplicación que se esta empleando para realizar la reservación del pase en la universidad. Se han definido los siguientes requisitos funcionales de acuerdo a los servicios que se desean implementar:

R1: Autenticar el usuario que solicite cualquier servicio con su usuario y contraseña.
R2: Brindar los servicios destinados a la reservación del pase.
1.1. Reservar la salida.
1.2. Reservar el regreso.
1.3. Modificar una reservación realizada.
1.4. Cancelar una reservación realizada.
R3: Realizar una búsqueda de estudiantes.
3.1. Por facultad.
3.2. Por grupo.
3.3. Por tipo de pase.
3.4. Por solapín.
3.5. Por nombre.
3.6. Por primer apellido.
3.7. Por segundo apellido.
R4: Solicitar los reportes utilizados por los directivos de la universidad para la organización del pase.
4.1. Reporte por ruta.
4.2. Reporte por año.
4.3. Reporte por facultad.
R5: Solicitar los servicios para la administración de los usuarios con privilegios.
5.1. Agregar un nuevo usuario.
5.2. Ver un listado de los usuarios existentes.
5.3. Eliminar un usuario.

R6: Solicitar los servicios referentes a las rutas del pase.

6.1. Agregar una nueva ruta.

6.2. Ver un listado de las rutas existentes.

6.3. Eliminar una ruta.

R7: Solicitar los servicios para la administración de los puntos de salida.

7.1. Insertar un nuevo punto de salida.

7.2. Editar un punto de salida existente.

7.3. Eliminar un punto de salida.

3.4. Requerimientos No Funcionales

Rendimiento

El sistema debe ser lo más eficiente posible para poder lograr un tiempo de respuesta adecuado.

Seguridad

Debe mantenerse la integridad de la información, es decir, la corrección e integridad de los datos.

Permitir que un Servicios Web sean utilizados solo por las personas autorizadas a para solicitar dicho servicio.

Confiabilidad.

Debe mantenerse la consistencia de los datos en correspondencia con la realidad.

Para una correcta ejecución de la aplicación se necesita:

Sistema operativo servidor: Windows 2000 o superior.
Navegador: Internet Explorer 4.0 o superior.
Servidor Web: Internet Information Services 5.1 o superior.

Hardware.

Velocidad del Procesador: Intel Pentium, 2.4 GHz
Mínimo de Memoria RAM: se recomiendan 128 MB
Mínimo de Disco duro: 15 MB.

3.5. Descripción del Sistema Propuesto

3.5.1. Descripción de los actores

Actores del Sistema	Descripción
Aplicación Web.	Es el sistema cliente a quien se le brindan los diferentes servicios para la Reservación del Pase.

3.5.2. Diagrama de Casos de Uso del Sistema

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

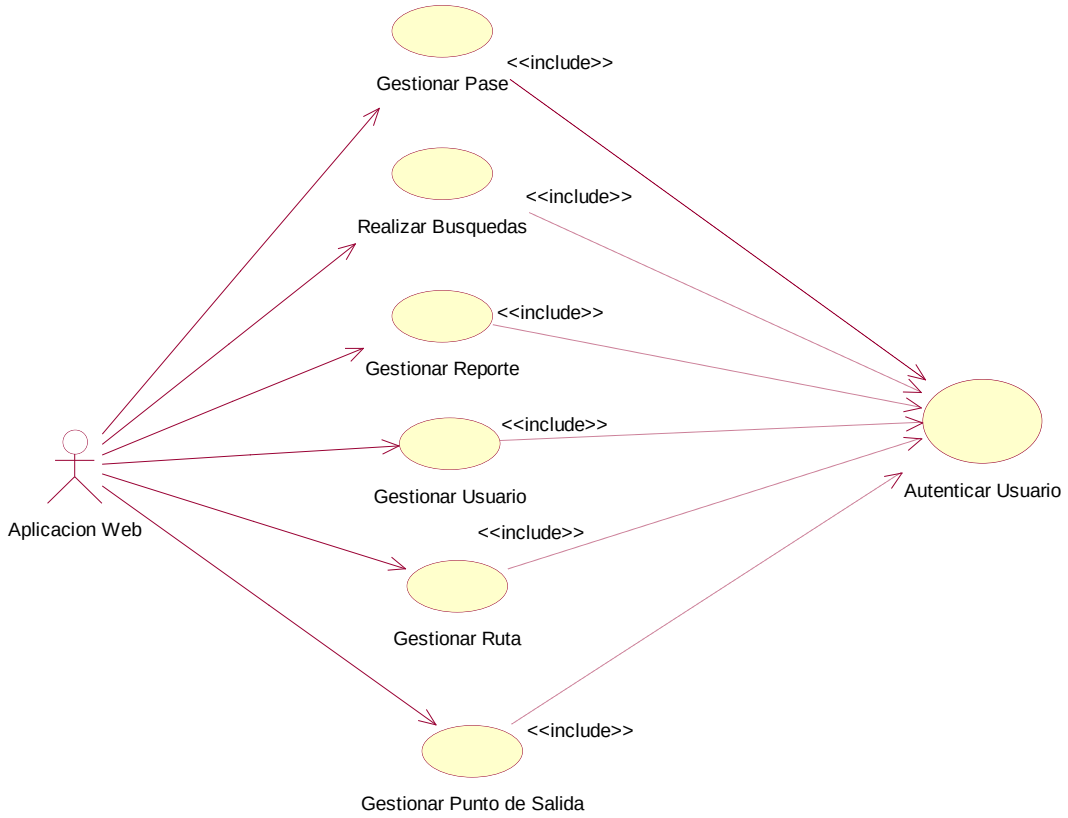


Figura 3.2 Diagrama de Casos de Uso del Sistema

3.5.3. Descripción de los Casos de Uso Del Sistema

Nombre del caso de uso	**Autenticar Usuario**
Actores	Aplicación Web (inicia)
Propósito	Verifica que cada usuario pueda solicitar solo los servicios que le han sido asignados.
Resumen El caso de uso se inicia cuando se solicita cualquier servicio, entre los datos necesarios para la ejecución del mismo debe viajar también el usuario y la contraseña del cliente, el sistema verifica que el usuario exista y que tenga permiso para solicitar dicho servicio.	
Referencia	R1
Precondición	Se realice la solicitud de algún servicio.
Poscondición	Se autentifique el usuario.
Caso de uso asociado:	
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1. La aplicación solicita un servicio	1.1 El web service solicita los datos para efectuar el servicio
2. La aplicación envía los datos	2.1 El web service verifica la valides de los datos para autenticar al cliente
	2.2 Se autentifica el usuario
Cursos alternativos	
	2.1 Si los datos enviados no son validos se muestra un mensaje de error

Tabla 3.2 Descripción del Casos Uso Autenticar Usuario

<table>
<tr><th>Nombre del caso de uso</th><th>Gestionar Pase</th></tr>
<tr><td>Actores</td><td>Aplicación Web (inicia)</td></tr>
<tr><td>Propósito</td><td>Ofrecer todos los servicios destinados a la gestión del pase.</td></tr>
<tr><td colspan="2">Resumen
El caso de uso se inicia cuando se solicita uno de los servicios destinados a la gestión del pase como: reservar ida, reservar regreso, modificar una reservación y cancelar una reservación. Se envía los datos necesarios para el procesamiento de los servicios a través de la aplicación, los servicios solicitados son ejecutados y se brinda una respuesta en correspondencia.</td></tr>
<tr><td>Referencia</td><td>R2</td></tr>
<tr><td>Precondición</td><td>El estudiante se encuentra previamente autenticado.</td></tr>
<tr><td>Poscondición</td><td>Obtener los datos devueltos por los servicios solicitados.</td></tr>
<tr><td>Caso de uso asociado:</td><td></td></tr>
<tr><td colspan="2">Curso normal de eventos</td></tr>
<tr><td>Acción del actor:</td><td>Respuesta del sistema:</td></tr>
<tr><td>1 La aplicación de desea Gestionar Pase del estudiante:
a. Si selecciona la opción Reservar, véase sección Reservar.
b. Si selecciona la opción Modificar, véase sección Modificar módulo.
c. Si selecciona la opción Cancelar, véase sección Eliminar módulo.</td><td></td></tr>
<tr><td colspan="2">Cursos alternativos</td></tr>
<tr><td>Acción del actor:</td><td>Respuesta del sistema:</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Prioridad: Crítico</td></tr>
</table>

Caso de uso:	**Gestionar Pase (Escenario: Reservar)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1. El sistema web realiza la petición del servicio pasando los datos necesarios	1.1 El servicio recoge los datos enviados por el sistema y verifica la integridad de los mismos.
	1.2 El web service realiza la petición de la web y envía respuesta.
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si se verifica que los datos sean defectuoso se muestra un mensaje de error
Requerimientos especiales:	

Caso de uso:	**Gestionar Pase (Escenario: Modificar)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.El sistema web realiza la petición del servicio pasando los datos necesarios	1.1 El servicio recoge los datos enviados por el sistema y verifica la integridad de los mismos.
	1.2 El sistema guarda los cambios de la Reservación.
Cursos alternativos	

Acción del actor:	**Respuesta del sistema:**
	1.1 Si se verifica que los datos sean defectuoso se muestra un mensaje de error

Caso de uso:	**Gestionar Pase(Sección: Cancelar Reservación)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.El sistema web realiza la petición del servicio pasando los datos necesarios	1.1 El servicio pide confirma la petición realizada.
2 El sistema confirma la petición	2.1 Se elimina la reservación y se envía mensaje.
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si la confirmación no es aceptada el web service mantiene los datos guardados

Tabla 3.3 Descripción del Casos Uso Gestionar Pase

Nombre del caso de uso	**Realizar Búsquedas**
Actores	Aplicación Web (inicia)
Propósito	Ofrecer todos los servicios destinados para buscar usuarios.
Resumen El caso de uso se inicia cuando se solicita uno de los servicios destinados a la búsqueda de usuarios que ya hayan reservado con anterioridad. Estas búsquedas pudieran realizarse por: facultas, grupo, tipo de pase, solapen, nombre, 1er apellido, 2do apellido o por cualquier combinación de los elementos anteriores. Introducidos los datos necesarios el sistema realiza la búsqueda según lo especificado por el cliente y devuelve un listado con los resultados.	
Referencia	R3
Precondición	El estudiante se encuentra previamente autenticado.
Poscondición	Obtener los resultados devueltos por el servicio solicitado.
Caso de uso asociado:	
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1. El sistema realiza una petición, envía los datos al servicio	1.1 El Servicio Web valida los datos y realiza la búsqueda
2 El sistema recoge el resultado y	2.1 El Servicio muestra el resultado de la búsqueda.

visualiza información.	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si los datos no son confiables el sistema muestra un mensaje de error.

Tabla 3.4 Descripción del Casos Uso Realizar Búsqueda

Nombre del caso de uso	**Gestionar Reportes**
Actores	Aplicación Web (inicia)
Propósito	Ofrecer todos los servicios destinados para la gestión de reportes.
Resumen El caso de uso se inicia cuando se solicita uno de los servicios destinados para la gestión de reportes. Estos servicios son obtener reporte por ruta, por año y por facultad. Después de escogido el tipo de reporte deseado. El sistema ejecuta el servicio y devuelve el resultado de dicho reporte.	
Referencia	R4
Precondición	El estudiante se encuentra previamente autenticado.
Poscondición	Obtener los resultados según el reporte que se haya solicitado.
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.La aplicación realiza petición búsqueda envía los datos	1.1 El web service comprueba que los datos sean correctos.
2. El sistema visualiza la respuesta.	2.1 El servicio realiza la búsqueda y envía una respuesta
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si se comprueba que los datos no son correctos
	1.2 El servicio Envía un mensaje error.

Tabla 3.5 Descripción del Casos Uso Gestionar Reporte

<table>
<tr><th>Nombre del caso de uso</th><th>Gestionar Usuarios</th></tr>
<tr><td>Actores</td><td>Aplicación Web (inicia)</td></tr>
<tr><td>Propósito</td><td>Ofrecer todos los servicios destinados para la gestión de usuarios.</td></tr>
<tr><td colspan="2">Resumen
El caso de uso se inicia cuando se solicita uno de los servicios destinados para la gestión de usuarios. Estos servicios son agregar un nuevo usuario, listar usuarios existentes y eliminar usuario. Después de escogido el tipo de servicio deseado y de introducido los datos necesarios para la ejecución del servicio requerido. El sistema ejecuta dicho servicio y devuelve el resultado esperados según el servicio solicitado.</td></tr>
<tr><td>Referencia</td><td>R5</td></tr>
<tr><td>Precondición</td><td>El administrador esta autenticado</td></tr>
<tr><td>Pos condición</td><td>Obtener los resultados según el servicio que se haya solicitado.</td></tr>
<tr><td colspan="2">Curso normal de eventos</td></tr>
<tr><td>Acción del actor:</td><td>Respuesta del sistema:</td></tr>
<tr><td>1 La aplicación desea Gestionar Usuario:
a. Si selecciona la opción Agregar, véase sección Agregar Usuario.
b. Si selecciona la opción Listar, véase sección Listar Usuario.
c. Si selecciona la opción Eliminar, véase sección Eliminar Usuario.</td><td></td></tr>
<tr><td colspan="2">Cursos alternativos</td></tr>
<tr><td>Acción del actor:</td><td>Respuesta del sistema:</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Prioridad: Crítico</td></tr>
</table>

Caso de uso:	**Gestionar Usuario(Sección: Agregar Usuario)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.La web solicita el servicio de agregar usuario, pasándole los datos	1.1 El servicio verifica que el usuario no este
	1.2 Se procesa el pedido y se confirma con una respuesta al cliente.
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si se confirma que el usuario no esta registrado
	1.2 El servicio web informa a la aplicación con un mensaje el error.
Caso de uso:	**Gestionar Usuario(Sección: Listar Usuario)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.La aplicación web referencia el servicio, pasándole los datos	1.1.El servicio recoge los datos y valida la solicitud
	1.2 El servicio web realiza el pedido solicitado
	1.3 Envía la respuesta de dicho pedido
2. La aplicación visualiza la respuesta.	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si al recoger los datos la solicitud no es validad
	1.2 El web service envía mensaje de error
2 .El sistema visualiza el mensaje	

Caso de uso:	**Gestionar Usuario(Sección: Eliminar Usuario)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.La aplicación web solicita el pedido, envía los datos	1.1 El web service verifica que el usuario exista
	1.2 Se validan los datos, se realiza el servicio
	1.3 Se envía una respuesta de confirmación
2.El sistema muestra al usuario la confirmación	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si el web service verifica que el usuario no existe
	1.2 Se envía una respuesta de confirmación que el usuario no esta en la BD
2.El sistema muestra al usuario la confirmación	

Tabla 3.5 Descripción del Casos Uso Gestionar Usuario

Nombre del caso de uso	**Gestionar Ruta**
Actores	Aplicación Web (inicia)
Propósito	Ofrecer todos los servicios destinados para la gestión de las rutas.
Resumen El caso de uso se inicia cuando se solicita uno de los servicios destinados para la gestión de las rutas del pase. Estos servicios son agregar una nueva ruta, listar rutas existentes y eliminar ruta. Después de escogido el tipo de servicio deseado y de introducido los datos necesarios para la ejecución del servicio requerido. El sistema ejecuta dicho servicio y devuelve el resultado esperados según el servicio solicitado.	
Referencia	R6
Precondición	Los administradores se encuentran previamente autenticados.

Pos condición	Obtener los resultados según el servicio que se haya solicitado.
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1 La aplicación desea Gestionar Ruta: a. Si selecciona la opción Agregar, véase sección *Agregar Ruta.* b. Si selecciona la opción Listar, véase sección *Eliminar Ruta*. c. Si selecciona la opción Eliminar, véase sección *Listar Ruta*.	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
Prioridad: Crítico	
Caso de uso:	**Gestionar Ruta(Sección: Agregar Ruta)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.La aplicación web solicita el servicio de Agregar ruta pasando los datos necesarios	1.1 El servicio web verifica que los datos sean correctos
	1.2 El servicio web realiza la operación solicitada por la aplicación web
	1.3 El servicio envía el resultado del servicio solicitado
2.La aplicación web visualiza el resultado	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si el servicio verifica que los datos no son correctos

	1.2 El web service retorna un resultado de error
2.La aplicación web visualiza el resultado	
Caso de uso:	**Gestionar Ruta(Sección: Eliminar Ruta)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.El sistema escoge el servicio y pasa los datos	1.1 El web service verifica que el usuario este registrado en el sistema
	1.2 El web service pasa a realizar la función
	1.4 El servicio web emite una respuesta confirmando la operación
2.La aplicación web visualiza el resultado	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si no esta registrado
	1.2 El servicio envía un mensaje de error
2.El sistema informa al usuario el error ocurrido	
Caso de uso:	**Gestionar Usuario(Sección: Listar Ruta)**
1. La web pide el servicio pasando los datos	1.1 El servicio analiza los datos
	1.2 El servicio procesa la información
	1.3 El Web service obtiene el resultado de la operación y lo envía
2. La pagina web muestra el resultado del pedido.	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si los datos analizados no son correctos
	1.2 El web service emite un mensaje de error
2. La aplicación muestra el mensaje error al usuario.	

Tabla 3.7 Descripción del Casos Uso Gestionar Ruta

<table>
<tr><th>Nombre del caso de uso</th><th>Gestionar Punto de Salida</th></tr>
<tr><td>Actores</td><td>Aplicación Web (inicia)</td></tr>
<tr><td>Propósito</td><td>Ofrecer todos los servicios destinados para la gestión de los puntos de salida.</td></tr>
<tr><td colspan="2">Resumen
El caso de uso se inicia cuando se solicita uno de los servicios destinados para la gestión de los puntos de salida en la uci. Estos servicios son insertar un nuevo punto de salida, editar un punto de salida existente y eliminar un punto de salida. Después de escogido el tipo de servicio deseado y de introducido los datos necesarios para la ejecución del servicio requerido. El sistema ejecuta dicho servicio y devuelve el resultado esperados tras la ejecución del mismo.</td></tr>
<tr><td>Referencia</td><td>R7</td></tr>
<tr><td>Precondición</td><td>El administrador se encuentra previamente autenticado.</td></tr>
<tr><td>Poscondición</td><td>Obtener los resultados según el servicio que se haya solicitado.</td></tr>
<tr><td colspan="2">Curso normal de eventos</td></tr>
<tr><td>Acción del actor:</td><td>Respuesta del sistema:</td></tr>
<tr><td>1 La aplicación desea Gestionar Punto Salida:
a. Si selecciona la opción Agregar, véase sección Agregar Ruta.
b. Si selecciona la opción Listar, véase sección Eliminar Ruta.
c. Si selecciona la opción Eliminar, véase sección Listar Ruta.</td><td></td></tr>
<tr><td colspan="2">Cursos alternativos</td></tr>
<tr><td>Acción del actor:</td><td>Respuesta del sistema:</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Prioridad: Crítico</td></tr>
</table>

Caso de uso:	**Gestionar Punto Salida (Sección: Agregar Punto Salida)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.La Aplicación web pide el servicio de agregar ruta pasándole los datos	1.1 El servicio web recibe los datos y verifica que estén correctos
	1.2 El servicio procede a ejecutar la función solicitada
	1.3 El web service envía respuesta de la función solicitada
2. La web recoge la respuesta y visualiza el resultado del servicio.	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si el servicio verifica que los datos no son correctos
	1.2 El web service retorna un resultado de error
2.La aplicación web visualiza el resultado	
Caso de uso:	**Gestionar Punto Salida (Sección: Eliminar Punto Salida)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1.El sistema escoge el servicio y pasa los datos	1.1 El web service verifica que el usuario este registrado en el sistema
	1.2 El web service pasa a realizar la función
	1.4 El servicio web emite una respuesta confirmando la operación
2.La aplicación web visualiza el resultado	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si no esta registrado
	1.2 El servicio envía un mensaje de error

2.El sistema informa al usuario el error	
Caso de uso:	**Gestionar Usuario(Sección: Listar Ruta)**
Curso normal de eventos	
Acción del actor:	**Respuesta del sistema:**
1. La web pide el servicio listar ruta pasando los datos	1.1 El servicio analiza los datos
	1.2 El servicio procesa la información
	1.3 El Web service obtiene el resultado de la operación y lo envía
2. La pagina web muestra el resultado del pedido.	
Cursos alternativos	
Acción del actor:	**Respuesta del sistema:**
	1.1 Si los datos analizados no son correctos
	1.2 El web service emite un mensaje de error
2. La pagina web muestra el resultado del pedido.	

Tabla 3.8 Descripción del Casos Uso Gestionar Ruta

3.6. Conclusiones

Después del estudio de la propuesta de solución a desarrollar, la captura de requisitos y la descripción de los proceso del negoción. Se modelo a través de un diagrama de casos de uso las funciones del sistema que se corresponden con el conjunto de web services a desarrollar, además de realizar la descripción detallada de cada una de los casos de uso modelados. En este momento ya estamos en condiciones de continuar el desarrollo de la propuesta presentada dando paso a la construcción de la solución.

Capítulo 4
Construcción de la solución propuesta

4.1. Introducción

En este capítulo los componentes de la aplicación se tratan como clases y se representan las relaciones entre ellas. Se tratan algunos principios de diseño, estándares de codificación a seguir y por último se presentan los diagramas de despliegue e implementación, donde se definen los componentes que forman parte de la aplicación y los nodos que forman la topología de hardware, que soportará al sistema.

4.2. Diagramas de Clases

Para una mejor representación grafica se tomo la decisión de representar las clases del diseño por caso de uso, puesto que estas representarían de mejor forma las funcionalidades del sistema.

Entre las clases que se utilizaron se encuentran la clase de acceso a dato y clases de lógica del negocio, para de esta manera realizar un diseño basado en la arquitectura en capas.

4.2.1. Caso de Uso: Autenticación

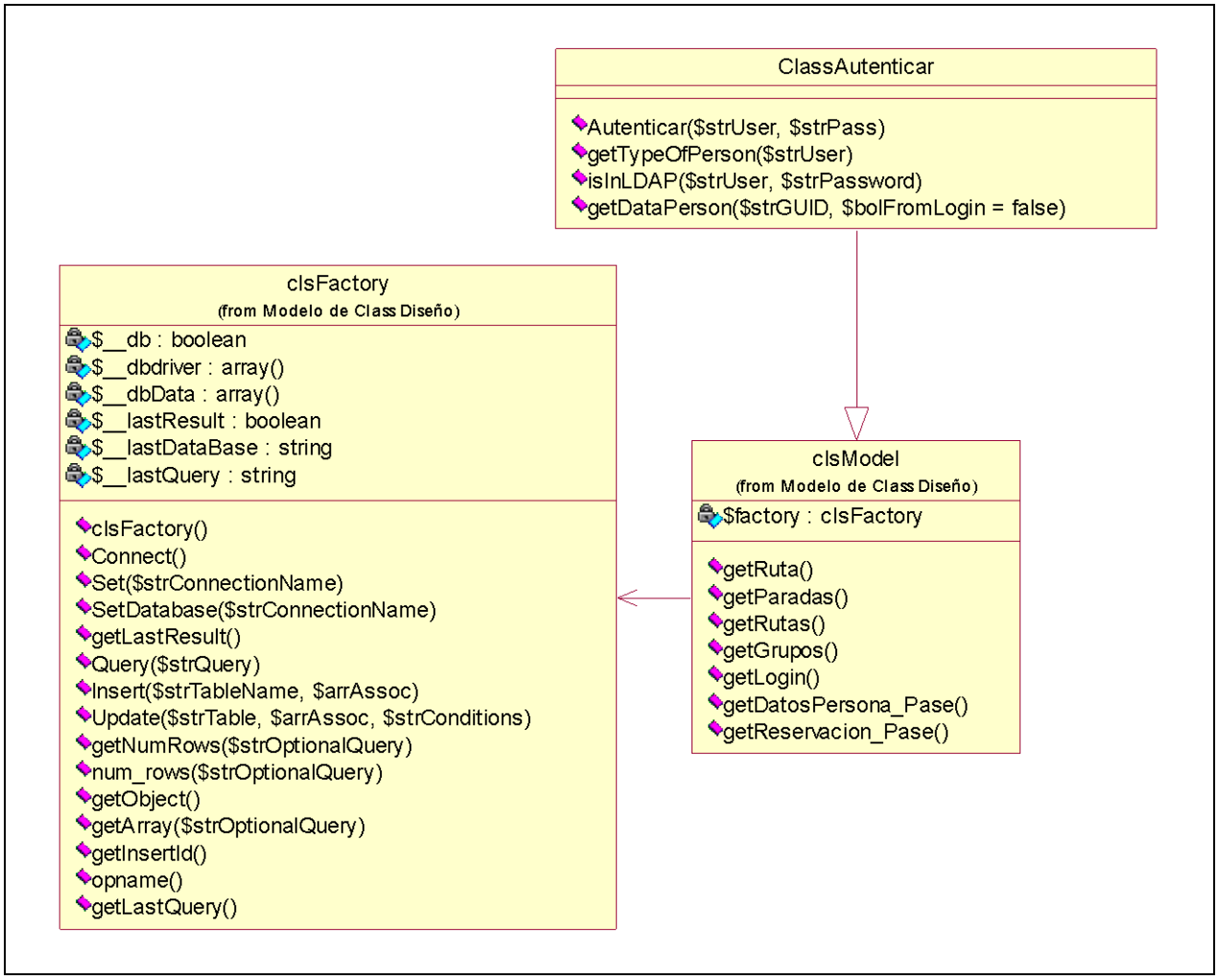


Figura 4.1 Diagrama de Clase de diseño Autenticar

4.2.2. Caso de Uso: Gestionar Pase

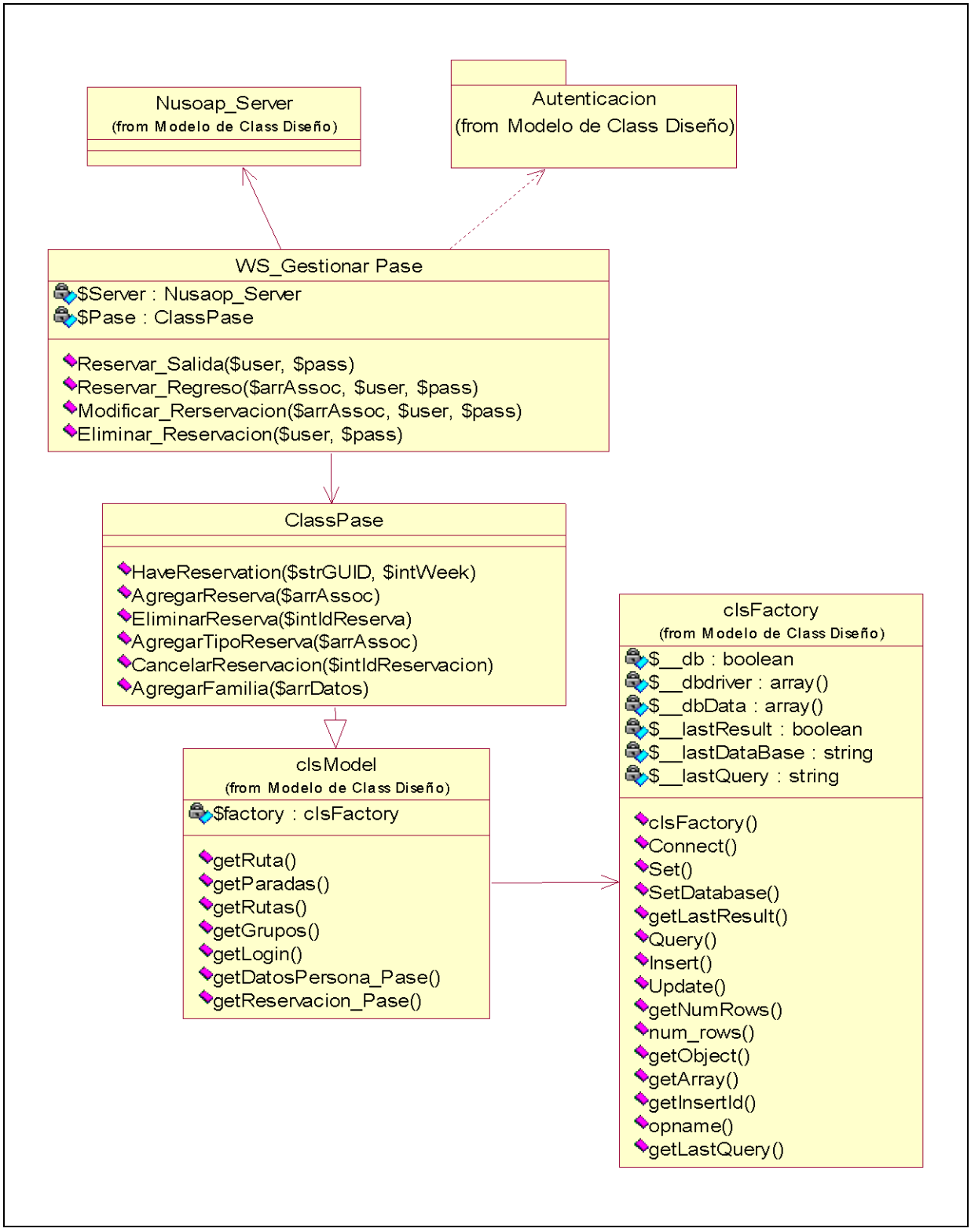


Figura 4.2 Diagrama de Clase de diseño Pase

4.2.3. Caso de Uso: Realizar Búsquedas

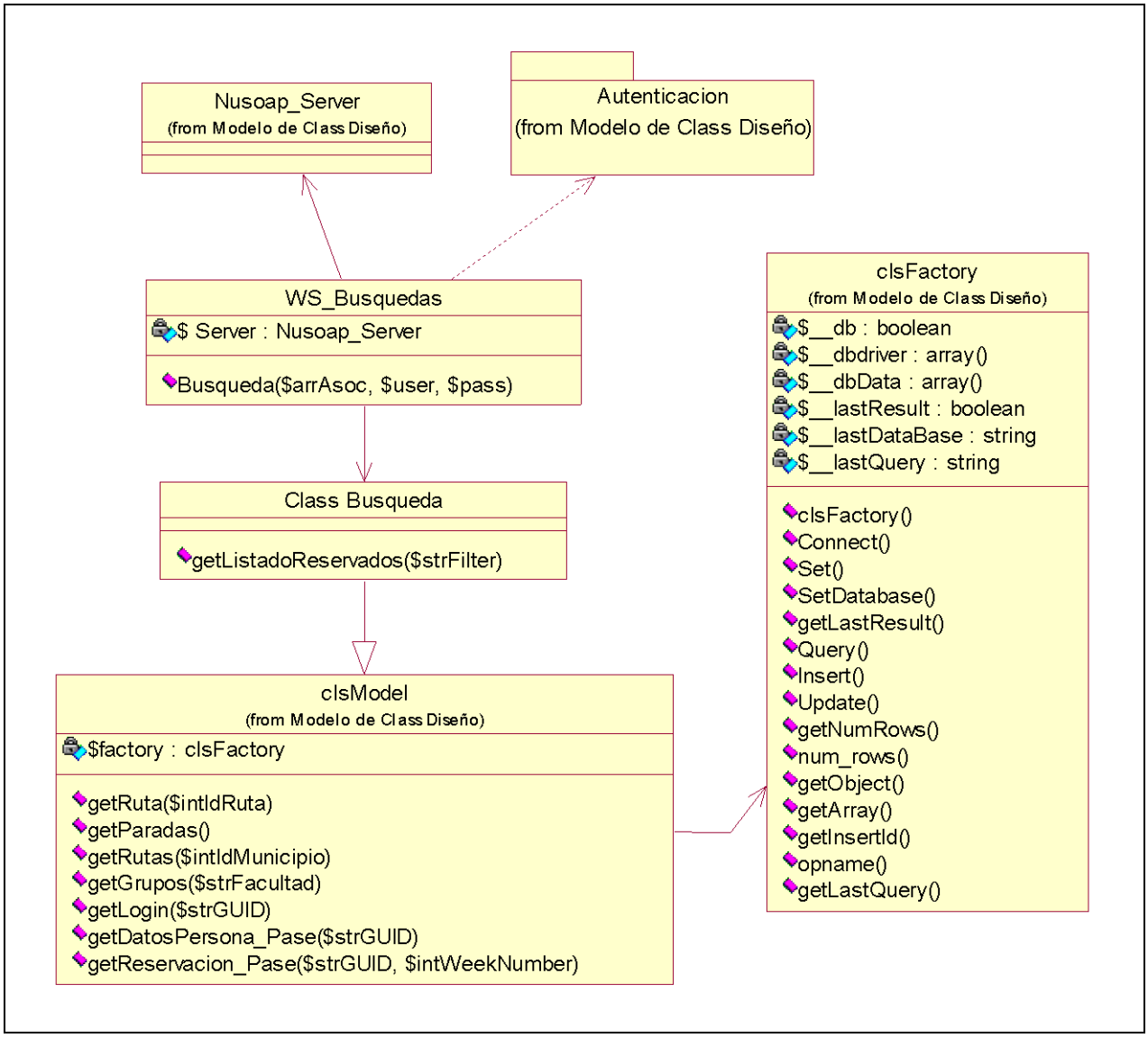


Figura 4.3 Diagrama de Clase de diseño Búsquedas

4.2.4. Caso de Uso: Gestionar Reporte

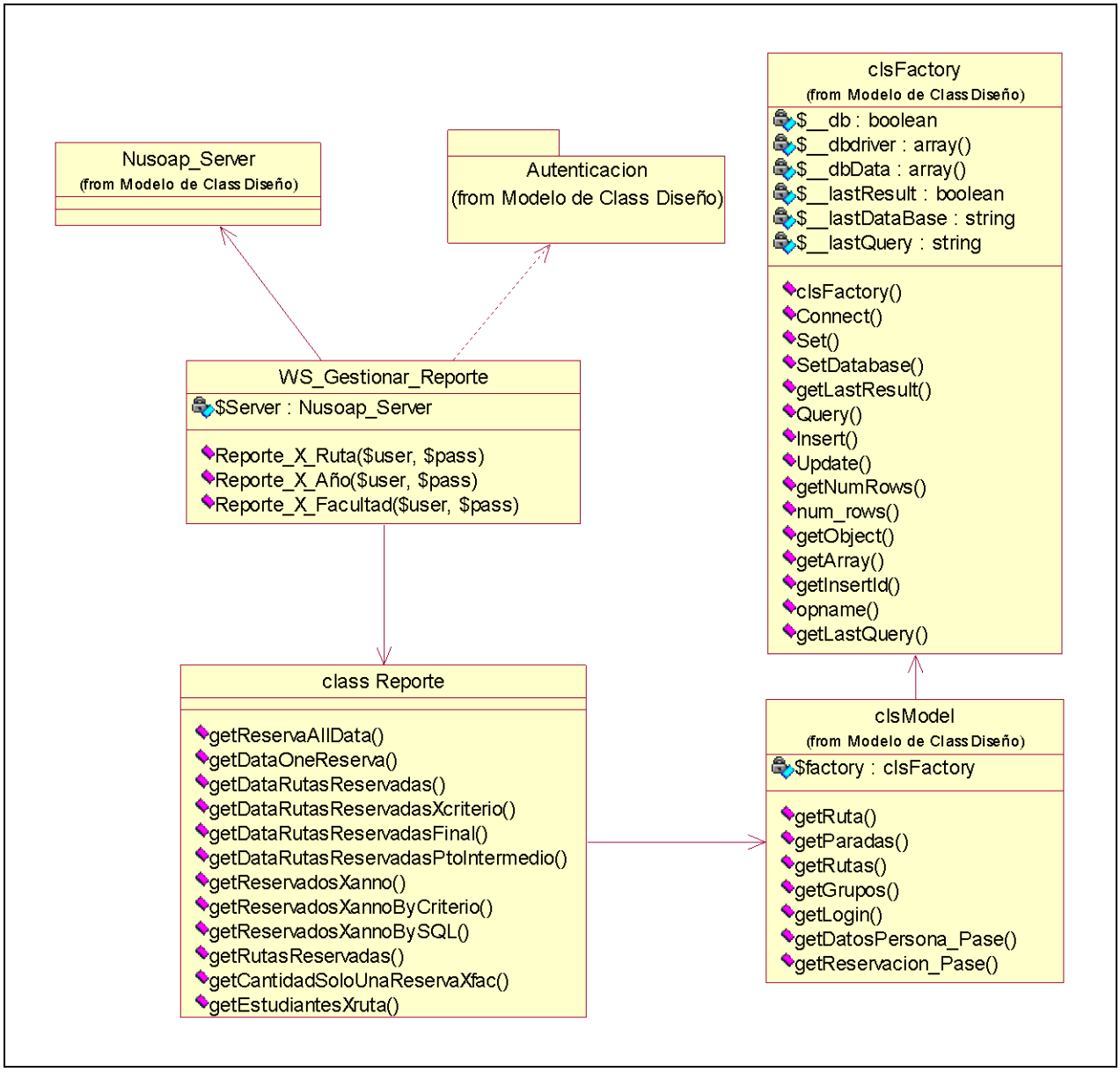


Figura 4.4 Diagrama de Clase de diseño Reporte

4.2.5. Caso de Uso: Gestionar Usuario

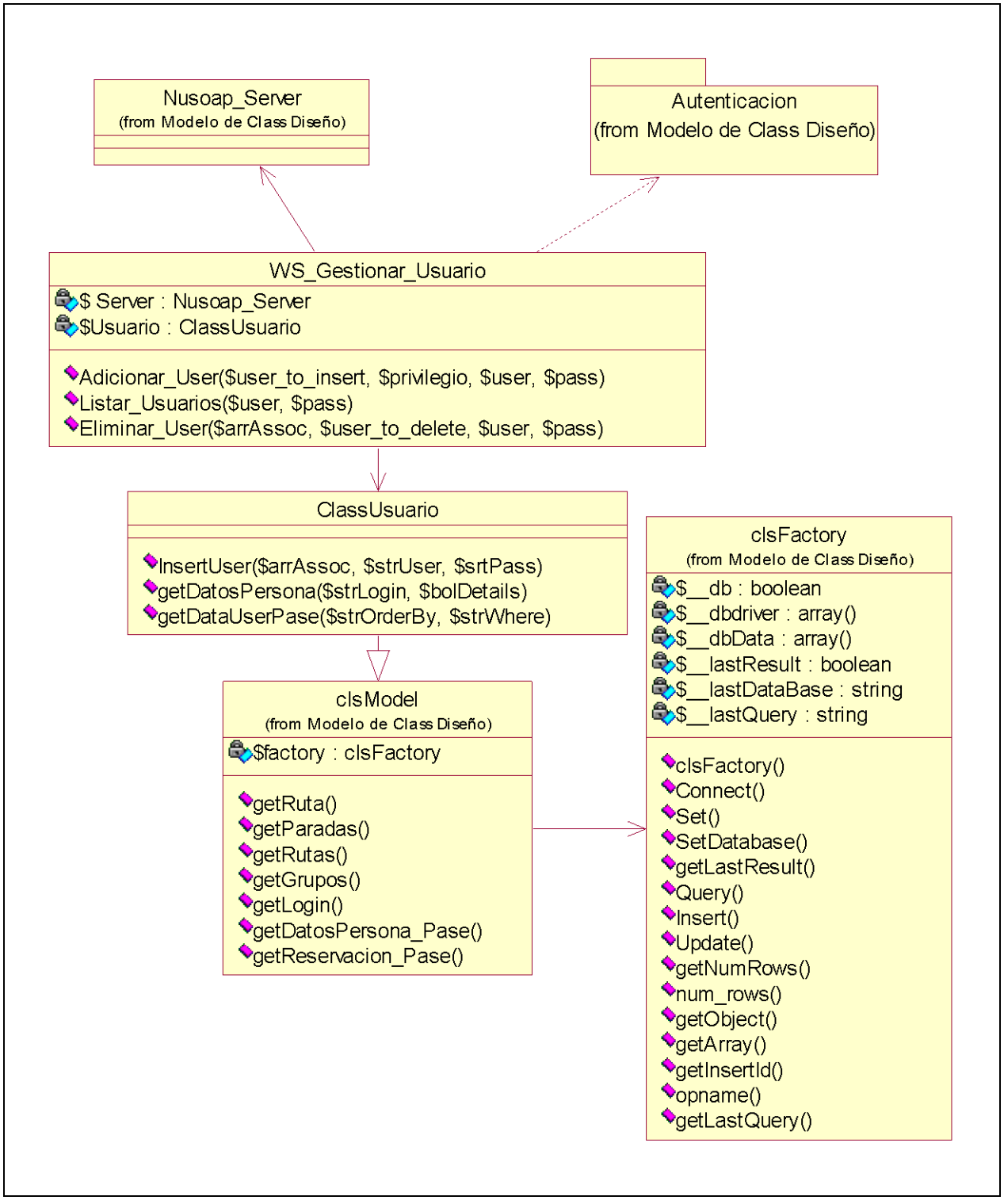


Figura 4.5 Diagrama de Clase de diseño Usuario

4.2.6. Caso de Uso: Gestionar Ruta

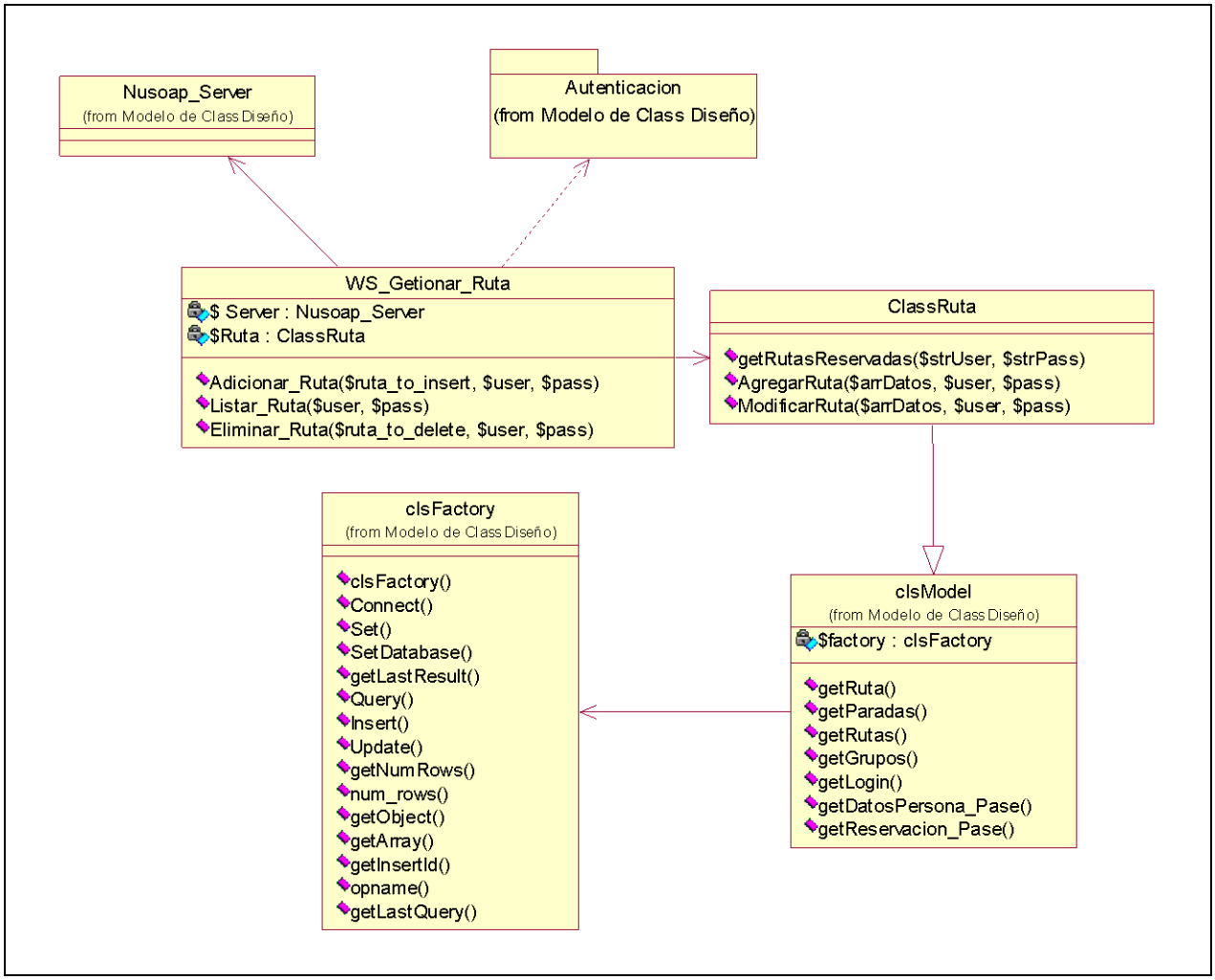


Figura 4.6 Diagrama de Clase de diseño Ruta

4.2.7. Caso de Uso: Gestionar Punto de Salida

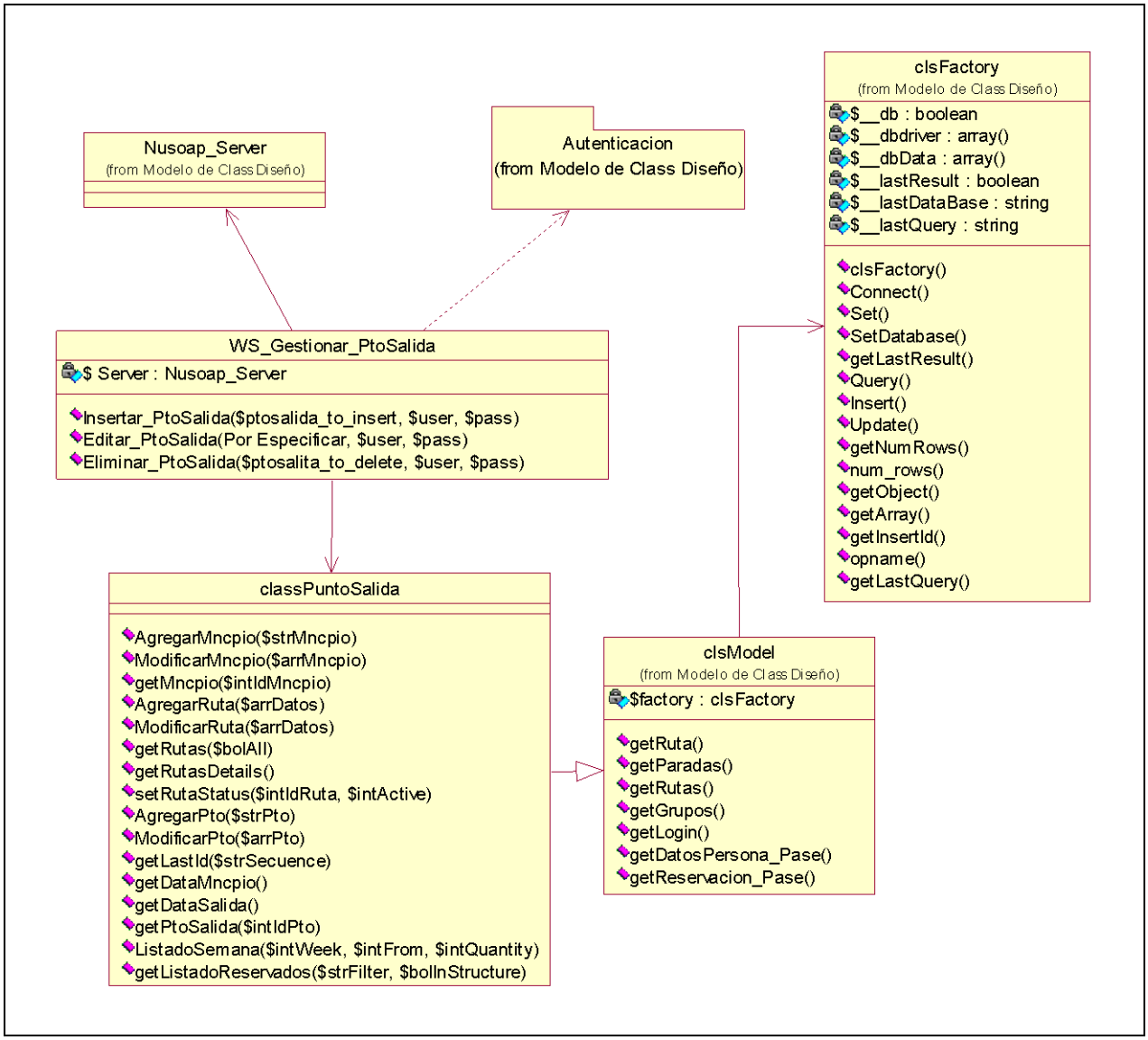


Figura 4.7 Diagrama de Clase de diseño Salida

4.3. Modelo de Despliegue

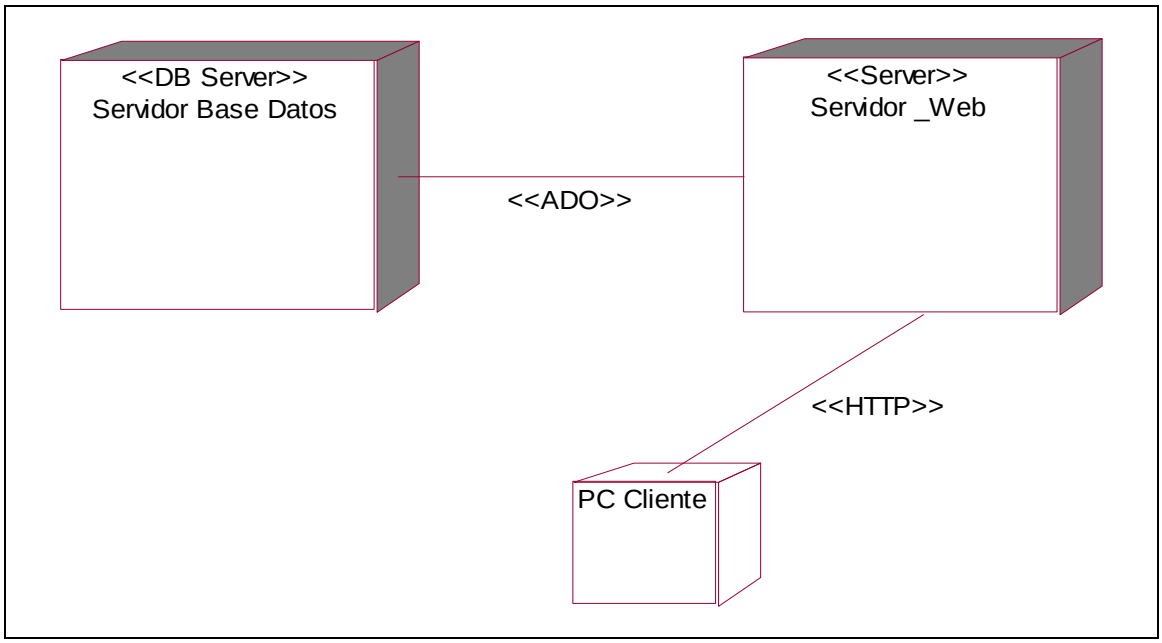


Figura 4.8 Diagrama de Despliegue

Componentes	Descripción
DB Server	Servidor donde se encuentra ubicada la base de dato de donde nuestros servicios se nutren de información.
PC Cliente	Ordenador desde donde los usuarios se conectan a las aplicaciones web
Server	Servidor Web que contiene las aplicaciones web y los servicios web.
HTTP	Protocolo utilizado para la conexión entre las PC Cliente y el servidor de aplicaciones y servicios web.
ADO	Librería utilizada para establecer la conexión entre el servidor de aplicaciones y servicios web y el servidor de base de dato.

Tabla 4.1 Descripción de los componentes del Diagrama de Despliegue

4.4. Modelo de Implementación

4.4.1. Diagrama de Componente por Caso de Uso

En este acápite se muestran los diagramas de implementación para cada uno de los casos de uso con que cuenta el sistema. Con los cuales se ha trabajando en el flujo de trabajo de análisis y diseño. Lo cual hemos decidido representarlos de esta forma por que esta seria la mejor manera la más entendible para el proceso, puesto que este, el desarrollo de los servicios Web que tenemos que representar no se relaciona en su totalidad.

4.4.2. Diagrama de Componente Caso de Uso Autenticar

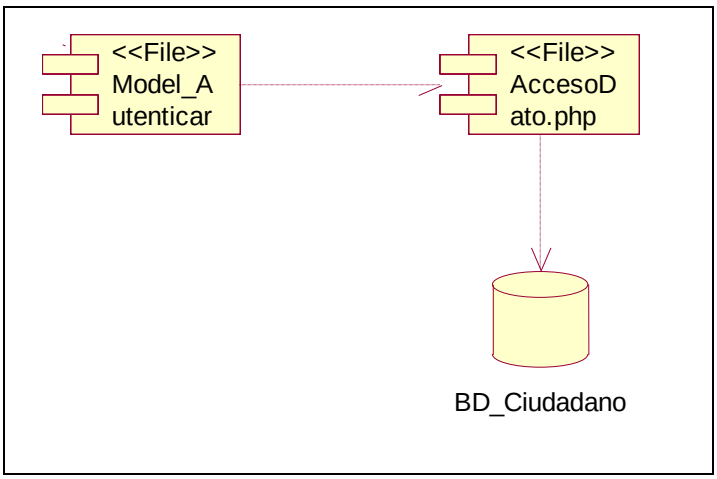


Figura 4.9 Diagrama Componente Autenticar

4.4.3. Diagrama de Componente Caso de Uso Gestionar Punto Salida

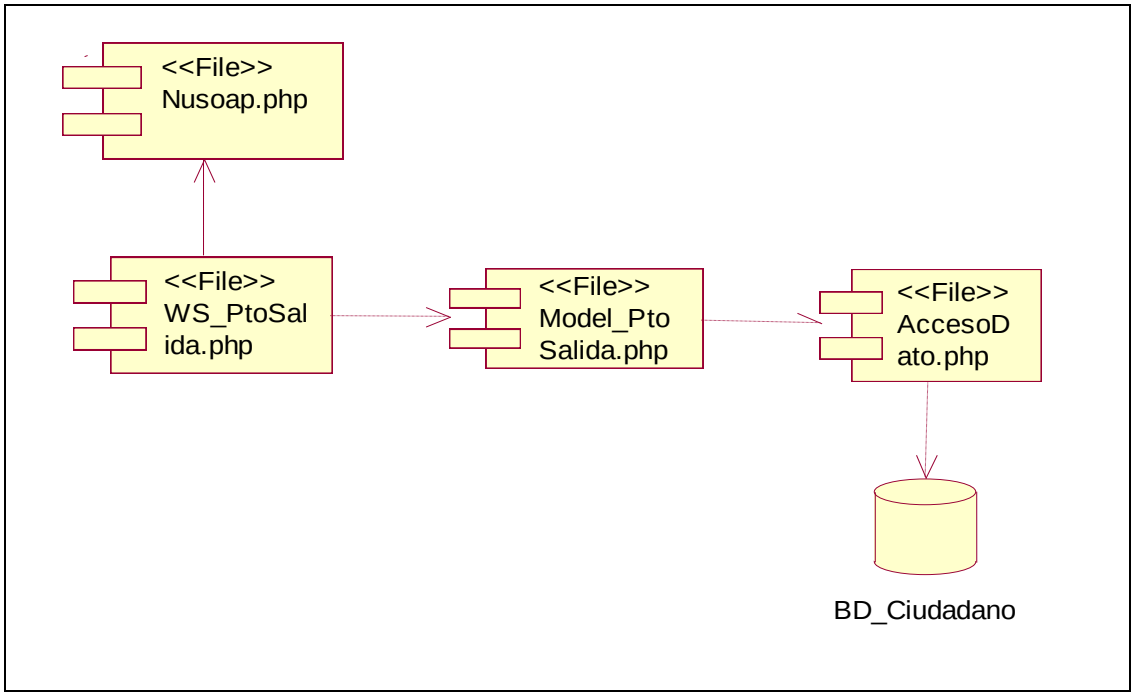


Figura 4.10 Diagrama Componente Autenticar

4.4.4. Diagrama de Componente Caso de Uso Realizar Búsqueda

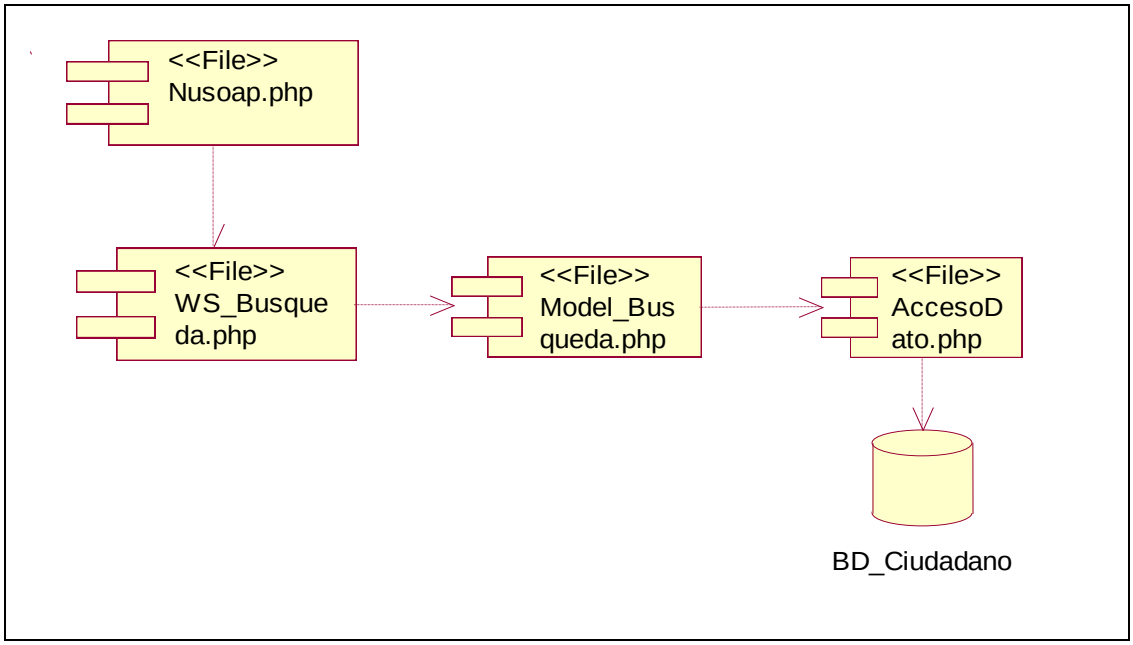


Figura 4.11 Diagrama Componente Autenticar

4.4.5. Diagrama de Componente Caso de Uso Gestionar Pase

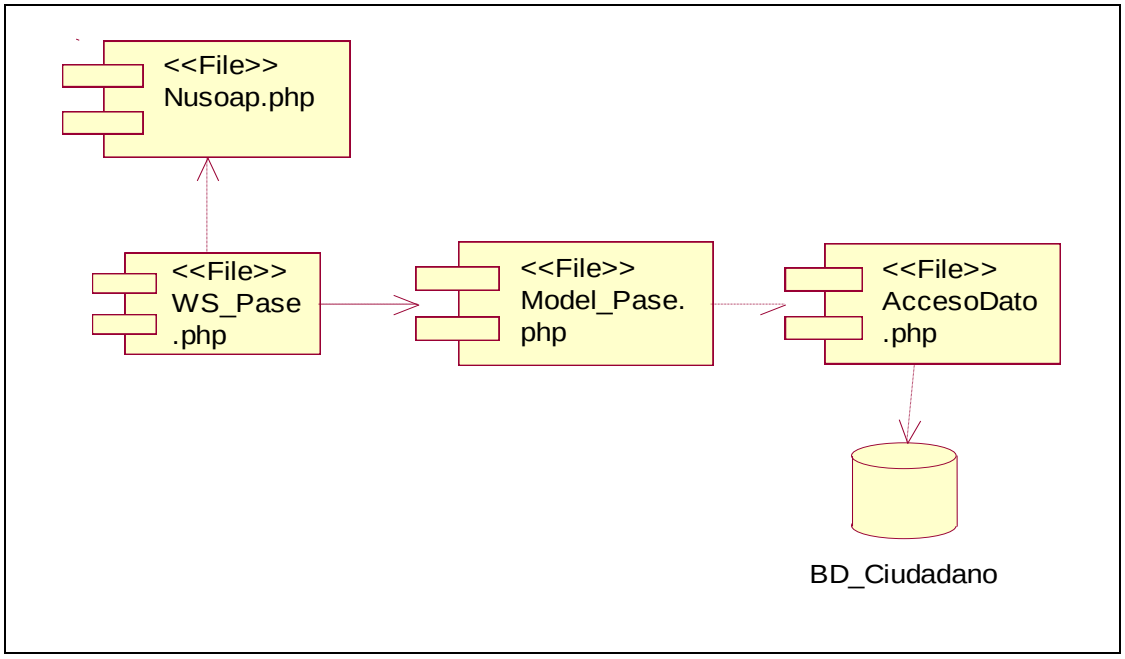


Figura 4.12 Diagrama Componente Autenticar

4.4.6. Diagrama de Componente Caso de Uso Gestionar Reporte

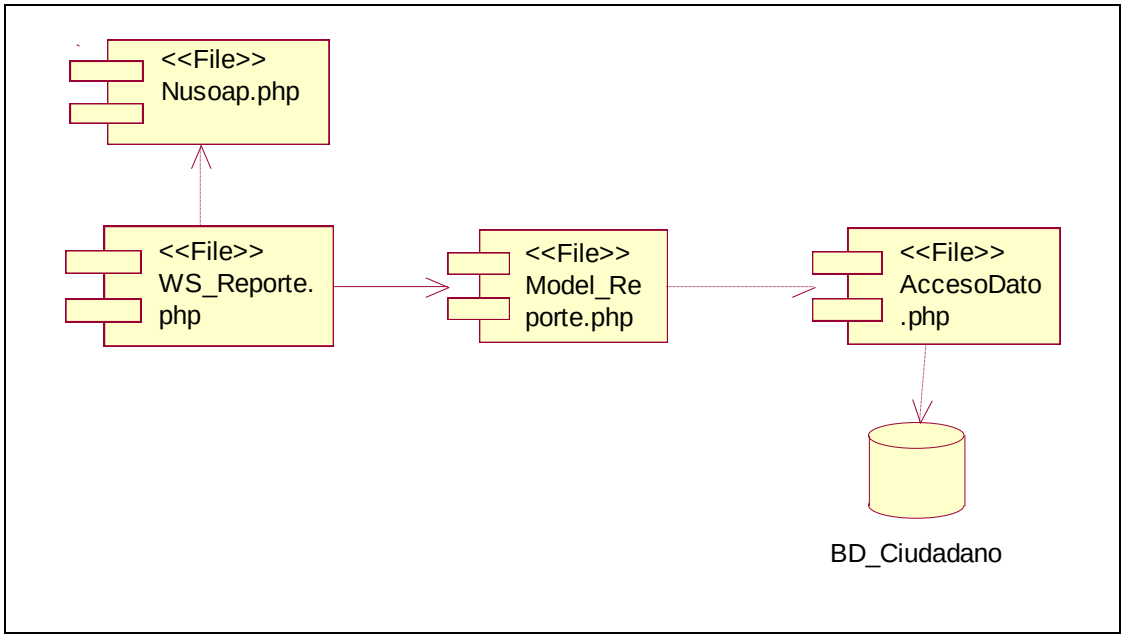


Figura 4.13 Diagrama Componente Autenticar

4.4.7. Diagrama de Componente Caso de Uso Gestionar Ruta

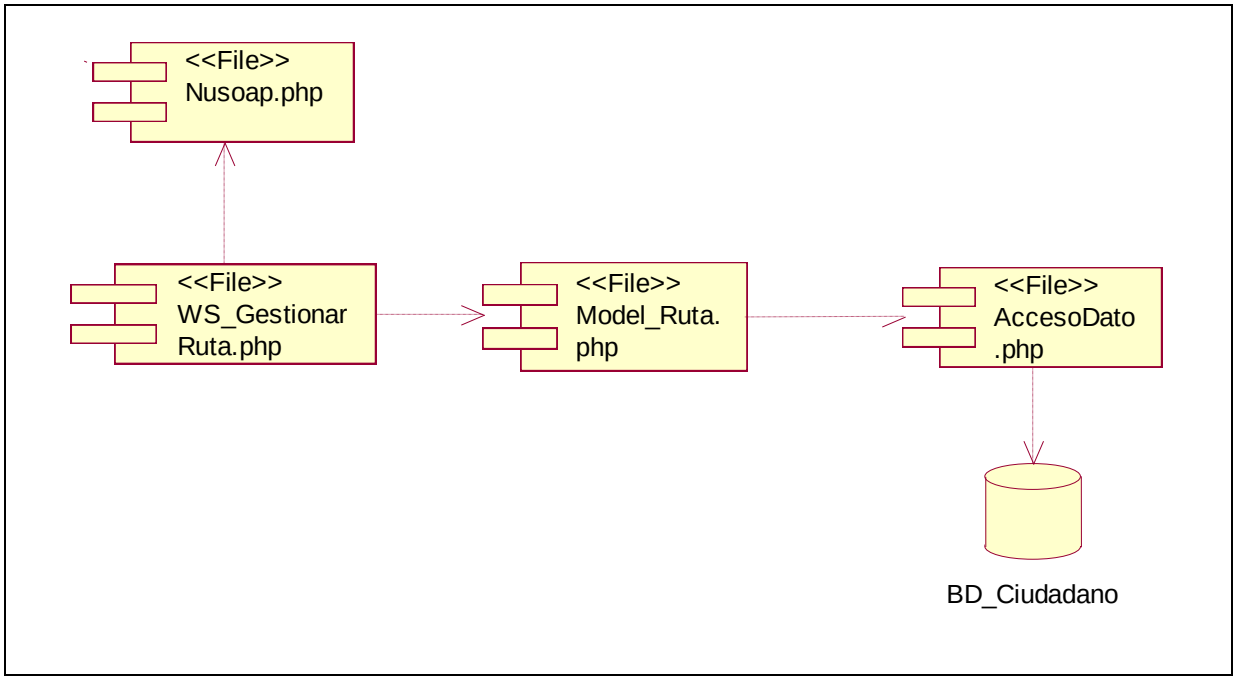


Figura 4.14 Diagrama Componente Autenticar

4.4.8. Diagrama de Componente Caso de Uso

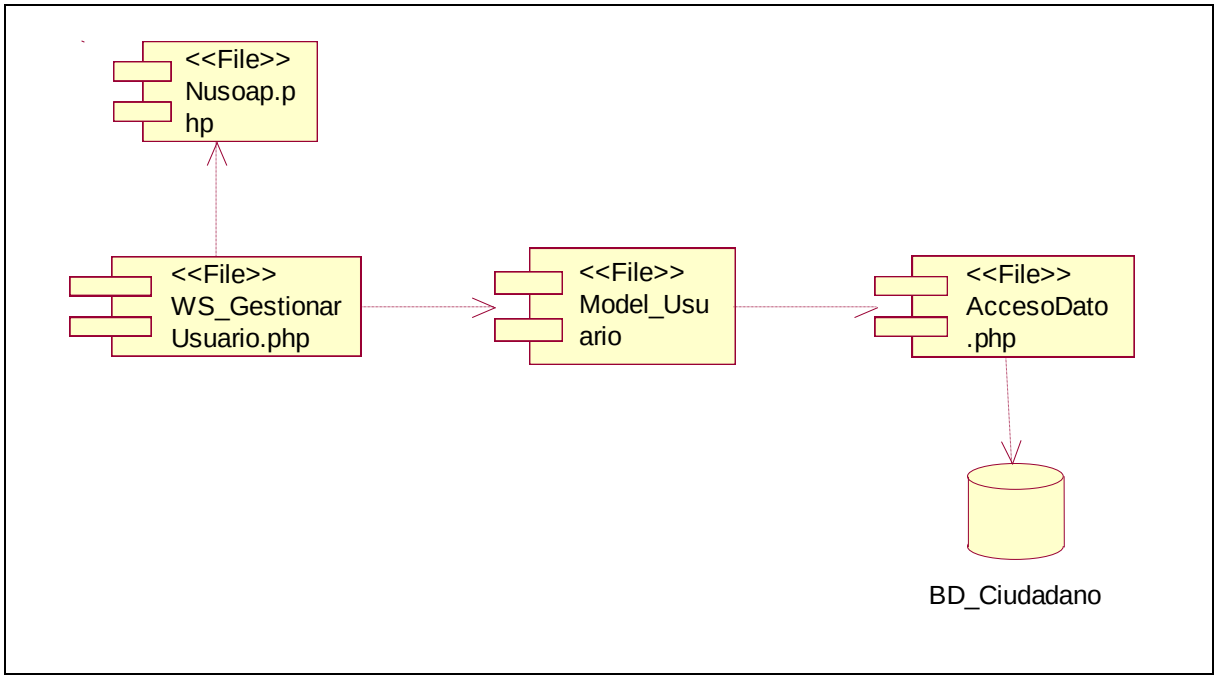


Figura 4.14 Diagrama Componente Autenticar

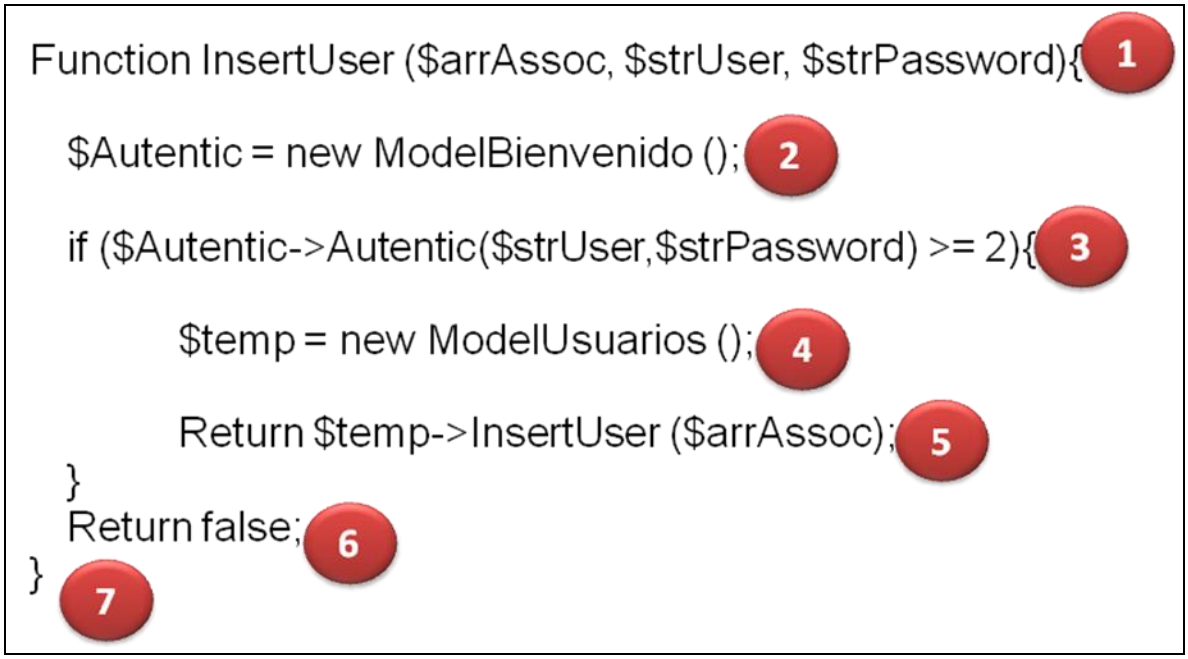
4.5. Prueba del sistema propuesto

Las pruebas se realizan a lo largo de todo el desarrollo del proyecto ya que esta puede arrojar resultados que alertarían a los desarrolladores respecto a la calidad del producto a terminar.

Existen dos tipos de prueba la de caja blanca y la prueba de caja negra. En la de caja negra se realiza una comprobación de los datos de entrada y los de salida, comprobando si los datos de salida que devuelve el sistema son los esperados de acuerdo a los datos que se le introdujeron.

En la prueba de caja blanca se realiza el traseo de un pedazo de código sentencia a sentencia asignándole valores numéricos a cada una de dichas sentencias. Posteriormente se construye un grafo con dichos valores para seguir la secuencia de pasos consecutivos por los que transcurre la ejecución del mismo.

Por último al grafo se le calcula la complejidad ciclomática la cual mide el número de decisiones lógicas en un segmento de código. A continuación se le realizará la prueba de caja blanca a uno de los servicios web desarrollados en el proyecto: “InsertUser”.



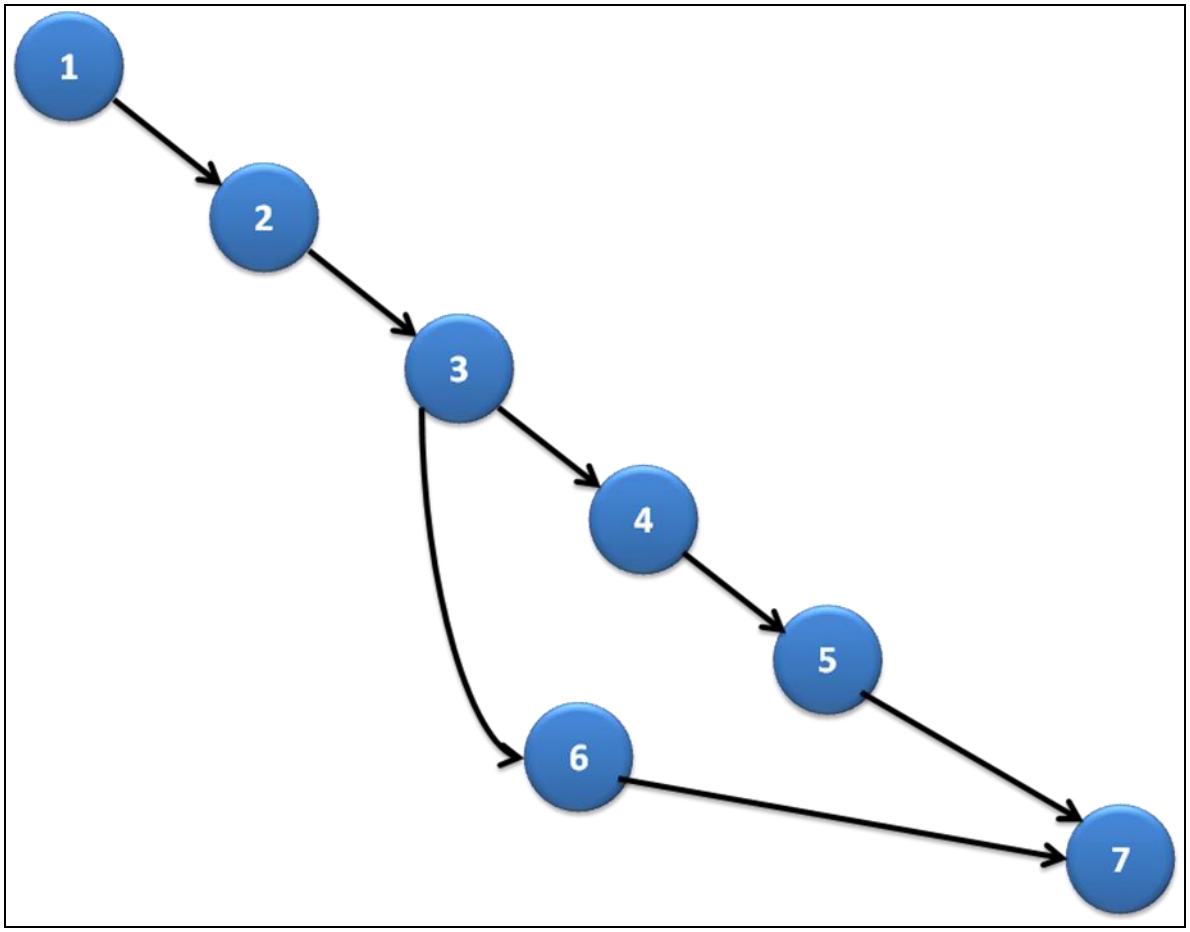


Figura 4.15 Grafo de Flujo Prueba

Cálculo de complejidad para el segmento de código anterior:

Complejidad = NumAristas – NumNodos +2
NumAristas = 7

NumNodos = 7

Complejidad = 7 – 7 + 2 = 2

4.6. Conclusiones

En este capítulo se tienen todos los elementos necesarios para la implementación del software. Se describen las clases y sus relaciones mediante los diagramas de clase, se define el diagrama de clases persistentes y el modelo de datos, para la construcción de la base de datos y por último se determinan los nodos y componentes mediante los diagramas de despliegue e implementación para poder mostrar como queda estructurada la aplicación físicamente.

Capítulo 5
Estudio de Factibilidad

5.1. Introducción

Antes de comenzar un proyecto de software y que este resulte lo esperado es necesario evaluar si es factible o no su confección. Aunque estas estimaciones no son exactas, nos dan una idea del esfuerzo a emplear, tiempo de trabajo, recursos requeridos, y cantidad de líneas de código. Esto evita gastos de recursos innecesarios en caso de que no se pueda dar continuidad a un proyecto en determinado momento, por falta de recursos y de tiempo, entre otras cosas.

Para realizar estas estimaciones la Ingeniería de Software ofrece un grupo de herramientas y técnicas entre las que se incluye COCOMO (*COnstructive COst MOdel*), y la estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner de Objectory AB, y posteriormente refinado por muchos otros autores. Donde utilizaremos una de estas para realizar el estudio de factibilidad del sistema. Además incluye un análisis de costos y de beneficios tangibles e intangibles del proyecto.

5.2. Planificación basada en puntos de casos de uso

En el método se utilizan los actores y casos de uso identificados para calcular el esfuerzo que costará desarrollar el proyecto. A los casos de uso se les asigna una complejidad basada en transacciones, que son: acción de usuario, respuesta sistema de los escenarios de los casos de uso. A los actores se les asigna una complejidad basada en el tipo de actor, es decir, si son interfaces con usuarios o si son interfaces con otros sistemas.
Para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de las asignaciones de la complejidad definida.

5.3. Pasos para realizar este proceso.

5.3.1. Paso 1. Identificar los Puntos de casos de uso Desajustados.

UUCP = UAW + UUCW
UUCP: Puntos de Casos de Uso sin ajustar.

UAW: Factor de Peso de los Actores sin ajustar.

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

Tipo de actor	Descripción	Factor de peso	Actores	Total
Simple	Otro sistema que interactúa con la aplicación a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1	1	0
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2	0	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	0	0
$UAW = \sum CantidadActores_i * Peso_i$				1

Tabla 5.1. Factor de Peso de los Actores sin ajustar

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	5	25
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	2	20
Complejo	El caso de uso tiene más de 8 transacciones.	15	0	0
$UUCW = \sum CantidadCU_i * Peso_i$				45

Tabla 5.2. Factor de Peso de los Casos de Uso sin ajustar.

UUCP = 1 + 45
UUCP = 46

5.3.2. Paso 2. Ajustar los Puntos de casos de uso.

UCP = UUCP * TCF * EF
UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

Formula para el cálculo de Factor de ambiente:

$TCF = 0.6 + 0.01 \times \Sigma (Peso_i \times Valor\ asignado_i)$

Este cálculo se realiza mediante el cómputo de un conjunto de valores de complejidad asignada a los diferentes factores con una escala de 0 a 5.

Significado de los valores

0: No presente o sin influencia.

1: Influencia incidental o presencia incidental.

2: Influencia moderada o presencia moderada.

3: Influencia media o presencia media.

4: Influencia significativa o presencia significativa.

5: Fuerte influencia o fuerte presencia.

Factor	Descripción	Peso	Valor asignado	Total
T1	Sistema distribuido	2	4	8
T2	Tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	4	4
T4	Funcionamiento Interno complejo	1	2	
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0,5	5	2,5
T7	Facilidad de uso	0,5	4	2
T8	Portabilidad	2	4	8
T9	Facilidad de cambio	1	4	4
T10	Concurrencia	1	4	4
T11	Incluye objetivos especiales de seguridad	1	3	3
T12	Provee acceso directo a terceras partes	1	3	3
T13	Se requieren facilidades especiales de entrenamiento de usuarios	1	1	1
			Sumatoria	49,5
TCF = 0.6 + 0.01 * ∑(Peso * Valor)			TCF	1,095

Tabla 5.3. Factores de Peso de Complejidad Técnica.

Cálculo del Factor de Ambiente (EF):

EF = 1.4 - 0.03 * Σ ($Peso_i$ * $Valor_i$) (Valor un número del 0 al 5)

Factor	Descripción	Peso	Valor asignado	Total
E1	Familiaridad con el modelo de proyecto utilizado	1,5	2	3
E2	Experiencia en la aplicación	0,5	2	1
E3	Experiencia en la orientación a objetos.	1	4	4
E4	Capacidad del analista líder.	0,5	3	1,5
E5	Motivación.	1	5	5
E6	Estabilidad de requerimientos	2	4	8
E7	Personal Part–Time	-1	5	-5
E8	Dificultad del lenguaje de programación	-1	3	-3
			Sumatoria	14,5
	EF = 1.4 - 0.03 * ∑(Peso * Valor)		EF	0,965

Tabla 5.4. Factores de Peso del Factor de Ambiente

Luego:

UCP = UUCP * TCF * EF	UCP	48,60705

5.3.3. Paso 3. Calcular esfuerzo del Flujo de Trabajo Implementación.

E = UCP * CF

E: esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: factor de conversión.

Para calcular **CF**:

CF = 20 horas-hombre (si $Total_{EF} \leq 2$)

CF = 28 horas-hombre (si $Total_{EF} = 3$ ó $Total_{EF} = 4$)

CF = abandonar o cambiar proyecto (si $Total_{EF} \geq 5$)

$Total_{EF}$ = Cant EF < 3 (entre E1 –E6) + Cant EF > 3 (entre E7, E8)

Luego:

Factor de conversión		CF	28
UCP = UUCP * TCF * EF		UCP	48,60705

E = UCP * CF		E	1360,9974

5.3.4. Paso 4. Calcular esfuerzo de todo el proyecto.

Actividad	Porcentaje %	Horas-Hombres
Análisis	30	816,59844
Diseño	20	544,39896
Implementación	50	1360,9974
Pruebas	10	272,19948
Sobrecarga (otras actividades)	10	272,19948
Total	120	2721,9948

Esfuerzo Total (Horas--Hombre)	ET1	2721,9948
Esfuerzo Total (Mes--Hombre)	ET2	18,9027417
Salario Promedio	SM	50
Cantidad de Hombres	CH	2
Costo Hombre--Mes	CHM	100
Costo Total	Costo	945,137083

5.4. Beneficios tangibles e intangibles

El desarrollo de estas librerías de servicios web ofrecerá grandes beneficios para la Universidad de las Ciencias Informáticas entre los que estarán:

Incrementa la interoperabilidad entre aplicaciones ya que cualquier aplicación que necesite utilizar uno de los servicios ya implementados con anterioridad para la reservación del pase, solo tendría que realizar su pedido.

Se logra una escasa dependencia entre los clientes (aplicaciones que soliciten alguno de los servicios de nuestro web service) y nuestros servicios web. Ya que el cliente no necesita conocer nada acerca de la implementación de nuestros servicios, salvo la definición de WSDL.

Gracias a la independencia de lenguaje de programación que existe entre el cliente y los servicios web, se facilita la migración hacía otra plataforma, ya que solo sería necesario modificar la aplicación cliente, sin necesidad de tocar la implementación de los servicios.
Los servicios web permiten que existan múltiples modos de invocación, ya que estos soportan tanto invocación estática como dinámica. Y múltiples estilos de comunicación: comunicación síncrona (RPC) y asíncrona (Mensajería). Además de que presentan una alta extensibilidad ya que al estar basados en XML son fáciles de adaptar, extender y personalizar.

Además de que con la implantación de nuestros servicios se esta beneficiando en gran medida a una posible implementación en el futuro de una arquitectura SOA en la universidad. Para cuando esto suceda ya la aplicación encargada de la gestión del pase estará acorde con las bases y especificaciones de dicha arquitectura.

5.5. Análisis de costos y beneficios

Teniendo encuentra lo planteado en el epígrafe anterior podemos decir que el desarrollo de este proyecto no ofrece grandes gastos pero si tiempo de desarrollo, puesto que la poca experiencias en el leguaje de programación y la técnica a emplear por parte de los desarrolladores, provocan la necesidad de dedica tiempo al estudio de las mismas. Al mismo tiempo esta solución puede beneficiar a la UCI en el modelo que se quiere implementar para la estructuración de las capas y proyectos dentro de la universidad.

El desarrollo de todo producto informático tiene un costo, este viene dado por la medida de los gastos de desarrollo y la puesta en funcionamiento. Para corregir estas pérdidas materiales tenemos que enfocarnos hacia las ventajas que nos proporciona el software. En el caso de este trabajo su costo esta infundado en $ 945,137083.

5.6. Conclusiones

Este capítulo describe el estudio realizado de la factibilidad del proyecto basados en la evaluación por punto de caso de uso, teniéndose en cuanta los factores que influyen en el desarrollo del Software.

Ofrece datos de sumo interés como es el costos del proyecto, tiempo de producción especificado por hombre-mes, valoración con respectó a los cálculos expuesto aparte de exponer beneficios y fiabilidad. Datos que permitirán valorar entre otras cosas si el proyecto es fiable o no.

CONCLUSIONES

- La utilización de la metodología RUP avala que el desarrollo del software quede estructurado por fases, con la calidad requerida, además de reducir el tiempo de desarrollo.
- El uso de SOAP permite que en la implementación los servicios queden definidos y listos para la comunicación entre cualquier plataforma y lenguaje.
- Los Servicios Web para la realización del pase brindan un conjuntos de mejoras en la implementación de la aplicación:

    1. Permite la comunicación de distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes.
    2. Pude ser ejecutada sobre cualquier plataforma.
    3. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

- El paradigma Programación Orientado a objeto garantiza una mejor codificación organización y diseño entre los servicios y aplicaciones.

- Se lograron definir, diseñan e implementar todos los Web Service requeridos para la aplicación del pase. Al mismo tiempo de estimar y computar los factores que influyen en el proceso, para avalar el proyecto.

RECOMENDACIONES

1. Expandir el uso de aplicaciones que estén estructuradas y basadas en el funcionamiento de Servicios Web.
2. Mantener y fomentar en la universidad el uso de aplicaciones distribuidas, que contribuyen a la construcción de la ciudad digital.
3. Adentrarse en el estudio de los beneficios de los servicios web para la construcción de aplicaciones con fines multiplataforma.

REFERENCIAS BIBLIOGRÁFICAS

1. G. B. Ivar Jacobson, James Rumbaugh. *El Proceso Unificado de Desarrollo de Software [Hoja 10].* 2000.

2. Proenza, Yuniel Eliades. *Arquitectura de Seguridad para Aplicaciones Web Empresariales [Hoja 7].* 2006.

3. Departamento de Ingeniería de Software. [En línea] 2006. http://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf7.

4. SMTP. [En línea] http://www.programacionweb.net/articulos/articulo/?num=412.

5. Intenet, "TCP\IP". [En línea] 2006. http://www.learnthenet.com/spanish/glossary/tcpip.htm.

6. Seguridad SW. [En línea] http://www.desarrolloweb.com/articulos/1538.php.

7. Santafé - Argentina. [En línea] http://www.softwarejr.com.ar/software-transporte.htm.

8. Martínez, Gerardo Moreno. "Ingeniería de Software y UML". [En línea] http://www.monografias.com/trabajos5/insof/insof.shtml.

9. Definicion. [En línea] 31 de 08 de 2005. [Citado el: 31 de 05 de 2007.] http://searchsmb.techtarget.com/sDefinition/0,,sid44_gci516025,00.html.

10. *Conferencia1: Introducción a la Ingeniería de Software.* Unversidad de las Ciencias Informáticas (UCI). 2004 - 2005.

11. Loyola, Vera Karol. MMU-Chile. [En línea] http://www.mmug.cl/articulos.php?id=287&tod=1.

12. Molpeceres, Alberto. Procesos de desarrollo: RUP, XP, FDD. [En línea] 15 de 12 de 2002. http://www.javahispano.org/articles.article.action?id=76.

13. —. Procesos de desarrollo: RUP, XP, FDD. [En línea] 15 de 12 de 2002. http://www.javahispano.org/articles.article.action?id=76.

14. Carlos Reynoso - Nicolas Kicillof, Universidad de Buenos Aires. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] 03 de 2004. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#24.

15. Carlos Reynoso - Nicolás Kicillof, Universida de Buenos Aires. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] 03 de 2004. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#24.

16. Intruducción a los Web Services en PHP. desarrolloweb.com. [En línea] http://www.desarrolloweb.com/articulos/1852.php.

17. Marañón, G. A. *Seguridad en servicios web.* [En línea] 05 de 02 de 2003. http://www.instisec.com/publico/verarticulo.asp?id=70.

18. Rail Europe. Página Web Oficial de los Ferrocarriles Europeos. [En línea] 2000. http://espanol.raileurope.com.

19. Aerocaribbean. La aerolínea regional cubana. [En línea] http://www.aero-caribbean.com/].

BIBLIOGRAFIA

2002. **Biblioteca UCI. [En línea] 2002. [Citado el: 14 de Mayo de 2007.] http://bibliodoc.uci.cu/pdf/reg02819.pdf.**

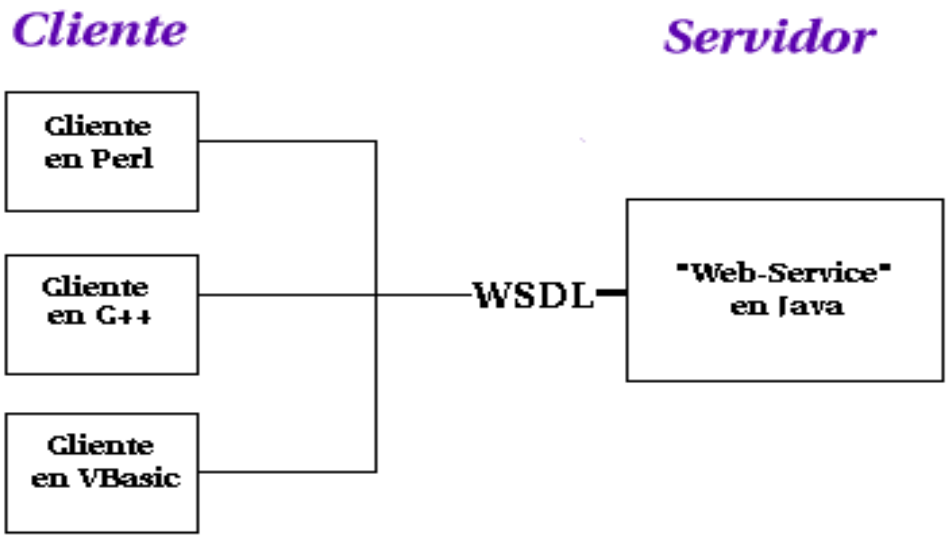
Ivar Jacobson, Grady Booch, Jame Rumbauch. 1999. ***El Proceso Unificado De Desarrollo De Software Tomo(I, II).*** **Ciudad de la Habana, Cuba : Felix Varela, 1999.**

Larman, Craig. 1999. ***UML Y Patrones, Introduccion al analisis y diseño orientado a objeto Tomo(I, II).*** **Cuidad de la Habana, Cuba : Felix Varela, 1999.**

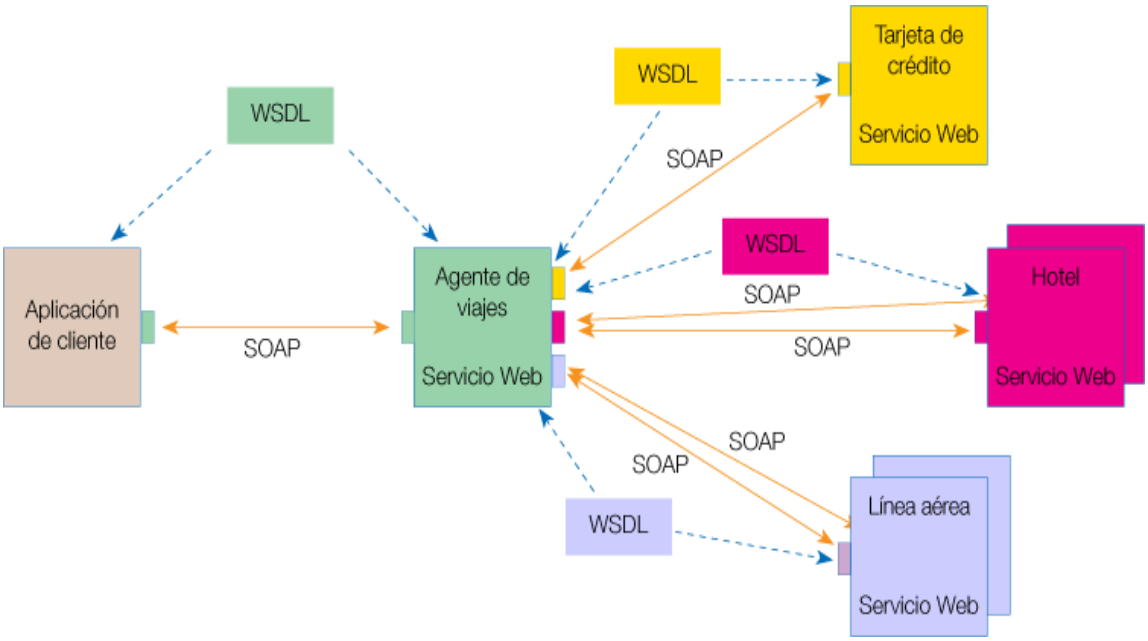
Pressman, Roger S. 2002. ***Ingniería del Software, un enfoque práctico Tomo (I, II).*** **Ciudad de la Habana, Cuba : Felix Varela, 2002.**

ANEXOS

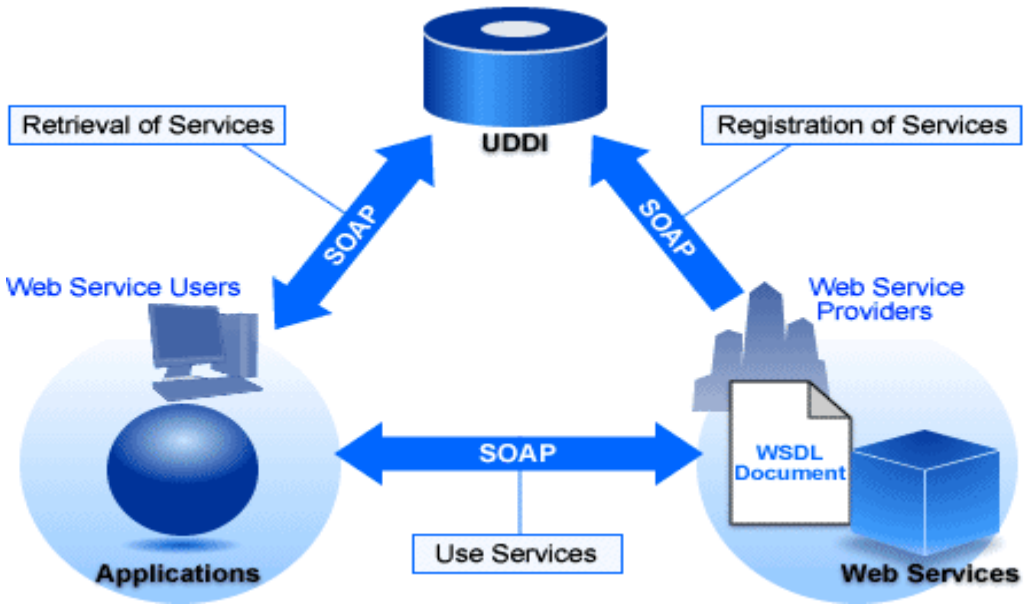
Anexo 1 Lógica de Web Services



Anexo 2 Interoperabilidad entre Web Services



Anexo 3 Representación de Arquitectura SOA



GLOSARIO

PHP: (Personal Home Page) Es un lenguaje interpretado de alto nivel impregnado en páginas HTML y ejecutado en el servidor.

WEB SERVICE: Sistema de software diseñado para soportar interacción máquina-a máquina sobre una red. Posee una interfaz descrita en un formato procesable por máquina.

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.

RPC: (*Remote Procedure Call*, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.