Universidad de las Ciencias Informáticas

Facultad 6

Título: “Desarrollo de una aplicación generadora de Servicios Web”.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores: Juan Antonio González Cardentey

Lázaro Jesús Ribas Acosta

Tutor: Ing. Deibys Greguas Navarro

Asesor: Ing. Manuel Alejandro Gil Martín

Junio,2009

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a 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 _____.

Lázaro Jesús Ribas Acosta

_____

Juan Antonio González Cardentey

_____

DATOS DE CONTACTO

Ing. Deibys Greguas Navarro
Profesor adjunto
Instructor recién graduado en el 2008 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI).
Miembro del grupo de desarrolladores del proyecto Match Mind.
Diseñador grafico de los proyectos Telebanca y Sistema de Gestión Penitenciaria (SIGEP), ambos en actual funcionamiento.
Miembro del grupo de desarrolladores de la Dirección de Informatización de la Infraestructura Productiva de la UCI.

Nombre: Manuel Alejandro Gil Martín
Categoría Docente: Instructor
Graduado de Ingeniería Informática CUJAE, 2005
Trabajó en la Dirección de Informatización de la UCI.
Fue subgerente del proyecto SOA para PDVSA.
Actual Jefe de Grupo de Informatización de la Empresa Albet.

AGRADECIMIENTOS

Juan Antonio:

A mi Grande y Fiel amigo. A mi abuela Gladis y mi mamá Felicia. A mi esposa Mariannis.

A toda mi familia. A mis hermanos de Candelaria. A mis hermanos de la UCI. A mis compañeros y amigos de aula. A mi hermano y dúo de tesis Lázaro. A mis profes. A Deibys y a Manuel. A la UCI.

A todos aquellos que de alguna menara contribuyeron en la realización de este sueño.

Lázaro:

A mi mamá Marta y mi papa Jesús por brindarme su apoyo y cariño todo el tiempo, a mi hermana Gretell por estar a mi lado en momentos buenos y momentos malos, a mi amigo y hermano Juan Antonio, por demostrarme que los verdaderos amigos existen, a mi cuñado Luisito por compartir tanto conmigo, a mis amigos de la escuela, que fueron determinantes en la realización de este trabajo y sin los cuales no hubiera podido terminar en tiempo. A Deibys y a Manuel, como profesores que me guiaron siempre. A todas las personas que de una forma u otra me hicieron pasar momentos felices aquí en la Universidad.

DEDICATORIA

Juan Antonio:

A mi bebé Ian Daniel, quién me ha traído en sus pequeñas manos una razón más para vivir y sonreír.

Lázaro:

Dedico esta tesis a mi familia, en especial a mis padres; como fruto de todo el esfuerzo que ellos han hecho por mi formación como profesional y por ser los principales responsables de que yo me encuentre donde estoy ahora.

RESUMEN

Llevar a cabo el despliegue de una aplicación desarrollada e implantada en la Universidad de las Ciencias Informáticas(UCI) en ambientes externos como parte de la voluntad de automatizar determinadas instituciones de la sociedad cubana, implica en la mayoría de los casos la interacción con diferentes orígenes de datos, que añade complicaciones adicionales al proceso al propiciar la necesidad de transformaciones directas sobre el código además de quebrantar los principios de reutilización de los componentes.

El presente trabajo explica el proceso de desarrollo de una aplicación que genera Servicios Web a partir de diferentes orígenes de datos. Con el fin de llevar a cabo un adecuado ciclo de desarrollo y obtener resultados competentes se definen las tecnologías, la metodología que detalle el proceso de elaboración del software y las herramientas que más se ajustan al caso en cuestión inclinándose siempre por el software libre que favorece la independencia tecnológica.

PALABRAS CLAVE

Origen de Datos

Proceso

Servicios Web

Software

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo vertiginoso que poseen las tecnologías de la información y las comunicaciones, el auge que está teniendo el comercio electrónico, las formas de organización que está adquiriendo el ser humano y el modo en que se desenvuelve en la sociedad del siglo presente, demanda que se automaticen y despersonalicen en muchos casos los procesos cotidianos. Estos argumentos son cimientos que han sostenido nuevos desarrollos en el ámbito tecnológico, principalmente en el área del software, desarrollándose este con una nueva óptica y haciendo uso de nuevas arquitecturas Entre estas se destaca el uso de las arquitecturas web, usando de ellas estándares y protocolos que viabilizan la interoperabilidad de las aplicaciones sobre la red. **[1]**

En Cuba siempre han existido instituciones al tanto de estos avances, ejemplo de ello es la UCI, que como principal precursora de la idea de Nuestro Comandante de informatizar la sociedad cubana, despliega muchos de los sistemas implantados en ella en diversas instituciones del país.

Como escenario propicio para desarrollar Servicios Web por su concentración tecnológica, la presencia de una gran cantidad de aplicaciones Web y la necesidad de interacción entre las mismas, cuyo enfoque va más allá de la aplicabilidad al contexto de una Universidad, surge la necesidad de diversificar las funcionalidades de sistemas implantados, así como ampliar el acceso de los mismos a distintas fuentes de datos, que representa una ventaja adicional en el alcance externo de estas aplicaciones.

A partir de esta situación surge como **problema científico** ¿Cómo lograr el intercambio de información entre diferentes orígenes de datos y sistemas informáticos existentes?, del cual se tendrá como **objeto de estudio,** los procesos de intercambio de datos entre aplicaciones informáticas y como **campo de acción**, los Servicios Web como medio de comunicación entre sistemas Informáticos y diferentes orígenes de datos.

Para dar solución al problema planteado se plantea el siguiente **objetivo general:**

- Desarrollar una aplicación informática que genere Servicios Web a partir de diferentes orígenes de datos.

Como **objetivos específicos** se tienen:

- Identificar las funcionalidades que debe cumplir el sistema.
- Diseñar el sistema.
- Implementar la aplicación.

Con diferentes **tareas** a cumplir tales como:

- Realización de la fundamentación teórica de la aplicación.
- Investigación del estado del arte referente a soluciones informáticas que posibiliten la comunicación entre aplicaciones existentes y diferentes orígenes de datos.
- Definición de tecnologías y herramientas para el desarrollo del producto.
- Realización de las actividades pertenecientes al flujo de trabajo de requerimientos.
- Realización de las actividades pertenecientes al flujo de trabajo de arquitectura.
- Realización de las actividades pertenecientes al flujo de trabajo de implementación.
- Realización de pruebas exploratorias.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN.

En este capítulo se hace una breve descripción de que es un Servicio Web partiendo de su concepto, se habla igualmente de las ventajas y desventajas de su uso; además de otros aspectos importantes a tener en cuenta durante el desarrollo del proceso de la aplicación. También se trata acerca de tecnologías, herramientas y metodologías haciendo mención de las más importantes, dando detalles de algunas de ellas y profundizando en las que serán usadas.

1.2. ESTADO DEL ARTE.

A través de una investigación realizada no se evidenció la existencia de aplicaciones que dieran solución al problema planteado y cumplieran con el objetivo general de este trabajo. A pesar de esto es importante resaltar la existencia de aplicaciones que permiten generar la estructura de un Servicio Web a partir de la interpretación de un WSDL o el modelado de datos. Por mencionar algunas tenemos Netbeans **[2]** y Axis **[3].**

1.3. SERVICIOS WEB

La necesidad de estandarizar la comunicación entre diversas plataformas y lenguajes de programación provoca el surgimiento de los Servicios Web, estos no son más que un conjunto de estándares y protocolos. **[4]**

Ya en ocasiones anteriores pero sin éxito se crearon otros estándares, estos no tuvieron mucha acogida por su dependencia del vendedor o hacer uso de RPC (Remote Procedure Call). **[5]**

La aplicación a desarrollar generará un servicio web en código PHP, pues inicialmente se pretende que esta sea usada por aplicaciones desarrolladas en este lenguaje. En la creación del servicio web se tuvo en cuenta el uso de PDO (PHP Data Objects) este es una extensión de capa de abstracción que permite obtener datos y realizar consultas de igual forma para distintos gestores de base de datos. **[6]**

La ventaja principal que aporta el uso de los servicios web para el desarrollo de este trabajo, es que permite que la aplicación a desplegar un ambiente diferente, pueda intercambiar información con los orígenes de datos existentes, sin importar las plataformas en que estos se encuentren o sus propiedades. **[4]**

1.4. TIPO DE APLICACIÓN

Para desarrollar la aplicación se puede hablar de dos formas de desarrollo de esta, una es la del tipo Web y la otra la del tipo escritorio, a continuación veremos las ventajas que ofrecen cada una de ellas respectivamente:

1.4.1. VENTAJAS DE LAS APLICACIONES WEB

- Acceso desde cualquier lugar del mundo, sólo es necesario conexión a internet y un navegador Web.
- La aplicación no necesita instalación, se arranca con un navegador Web.
- Control absoluto sobre los usuarios.
- Actualización más fácil.
- Es una aplicación multiplataforma. **[7]**

1.4.2. VENTAJAS DE LAS APLICACIONES DE ESCRITORIO

- Navegación e interfaz de usuario más rápida que en una aplicación Web.
- Normalmente no es necesaria una conexión a internet ya que funciona localmente.

- Fácil acceso a recursos locales.

- Fácil acceso a aplicaciones locales. **[7]**

Las aplicaciones web brindan buenas ventajas para desarrollar la aplicación, pero a pesar de esto se decide desarrollar el sistema cómo una aplicación de escritorio, debido a que la aplicación cuenta con una fecha de entrega, y para el equipo de trabajo, la manera más rápida de cumplirla es desarrollándola de escritorio.

1.5. TECNOLOGÍAS, METODOLOGÍAS Y HERRAMIENTAS.

Todo proceso de desarrollo de software es complejo; por lo tanto es necesario definir las metodologías y herramientas a usar en el mismo. A continuación se muestra una explicación de la selección de cada uno de estos puntos antes mencionados para la elaboración de la aplicación.

1.5.1. METODOLOGÍA DE DESARROLLO DE SOFTWARE.

Es necesaria para los desarrolladores la realización de un software con calidad y en el menor tiempo posible, siendo indispensable el conocimiento de la organización de las actividades para cada desarrollador y para el equipo en general. Por lo tanto es importante definir una metodología para dirigir las actividades estrechamente relacionadas al proceso de desarrollo de software con el fin de hacer confiable la producción.

Hay dos tipos de metodologías las tradicionales y las ágiles, entre ellas existen algunas diferencias que permiten decidir a través de cual se desarrollará la aplicación. Entre las principales metodologías tradicionales se encuentran Rational Unified Process (RUP) y Microsoft Solution Framework (MSF) estas centran su atención en llevar a cabo una documentación exhaustiva de todo el proceso, es alto el costo a la hora de implementar un cambio y tienen poca flexibilidad en proyectos donde el entorno es volátil.

Por otra parte las metodologías ágiles se desarrollan para dar solución a las dificultades antes planteadas de las metodologías tradicionales; teniendo como base dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa.

El poder retrasar las decisiones de una manera responsable es ventajoso tanto para el equipo de desarrollo como para el cliente, las principales ventajas de retrasar las decisiones son:

- Reduce el número de decisiones de alta inversión que se toman.
- Reduce el número de cambios necesario en el proyecto.
- Reduce el coste del cambio. **[8]**

También la planificación adaptativa permite al equipo de desarrollo estar preparado para el cambio e ir tomando decisiones en el avance del proyecto pues este va quedando fraccionado. Esta planificación a corto plazo permite mostrar al cliente el avance del software además de la retroalimentación para los desarrolladores haciendo el desarrollo de la aplicación más sensible ante situaciones que aceleren o atrasen el proceso.

Las metodologías ágiles plantean algunas ideas, establecidas en el Manifiesto Ágil **[9]** y que a continuación se mencionan varias de ellas:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un Plan. **[7]**

Existen varias metodologías ágiles dentro de las cuales se encuentran las siguientes:

XP Ó PROGRAMACIÓN EXTREMA.

Para ampliar la productividad en el desarrollo de programas esta metodología se basa en una serie de valores y prácticas. Este modelo pone su preferencia en aquellos trabajos que reducen los expedientes alrededor de la programación y que dan un resultado directo.

El principal objetivo que se perseguía al crear esta metodología era encontrar un método que el proceso de desarrollo del software fuera más sencillo. **[10]**

SCRUM.

Esta metodología es fundamentalmente para aquellos proyectos en los cuales los requisitos cambien con frecuencia. Tiene dos características fundamentales. Se programan reuniones a lo largo del proceso de desarrollo del software, en las que se destacan unas reuniones diarias de 15 minutos para coordinación e integración. La otra característica es que el software se realiza mediante iteraciones llamadas sprints, estas tienen una prolongación de 30 días. **[11]**

CRYSTAL METHODOLOGIES (METODOLOGÍAS DE CRISTAL).

Las Metodologías de Cristal se caracterizan por estar enfocadas en el uso de la menor cantidad de artefactos posibles y en las personas que conforman el equipo de desarrollo. El desarrollo del software es considerado un juego en el que priman la comunicación y la cooperación, limitado además por los recursos a usar. Los equipos de desarrollo pueden estar formados de 3 a 8 miembros o 25 a 50 llevando una clasificación, Crystal Clear y Crystal Orange respectivamente. Estos equipos son muy importantes por lo que es necesario la preparación de los miembros que la integran así como las tarea de trabajo en equipo bien definidas. **[11]**

OPEN UP (OPEN UNIFIED PROCESS)

OpenUP es un Framework de procesos de desarrollo de software de código abierto. Es un proceso modelo y extensible, dirigido a gestión y desarrollo de proyectos de software basados en

desarrollo iterativo, ágil e incremental; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

Este proceso de desarrollo unificado está basado en Rational Unified Process (RUP), desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad, basándose en los principios de Adaptación, Importancia a los involucrados e interesados en los resultados del proyecto; Colaboración, Valor a la iteración y Calidad Continua.

OpenUP permite un abordaje ágil al proceso de desarrollo de software, con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles, y tareas.

OpenUP es un proceso interactivo de desarrollo de software simplificado, completo y extensible. Es un proceso para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias. **[12]**

PRINCIPIOS BÁSICOS DEL OPENUP

- Colaboración para unificar intereses y compartir conocimientos.
- Equilibrio de prioridades competentes a maximizar el valor de los involucrados con el resultado del proyecto.
- Enfoque en la articulación de la arquitectura.
- Desarrollo continuo para obtener realimentación y realizar las mejoras respectivas. OpenUP/Basic se centra en articular la arquitectura para facilitar la colaboración técnica, reducir el riesgo y minimizar el sobreesfuerzo de desarrollo.**[12]**

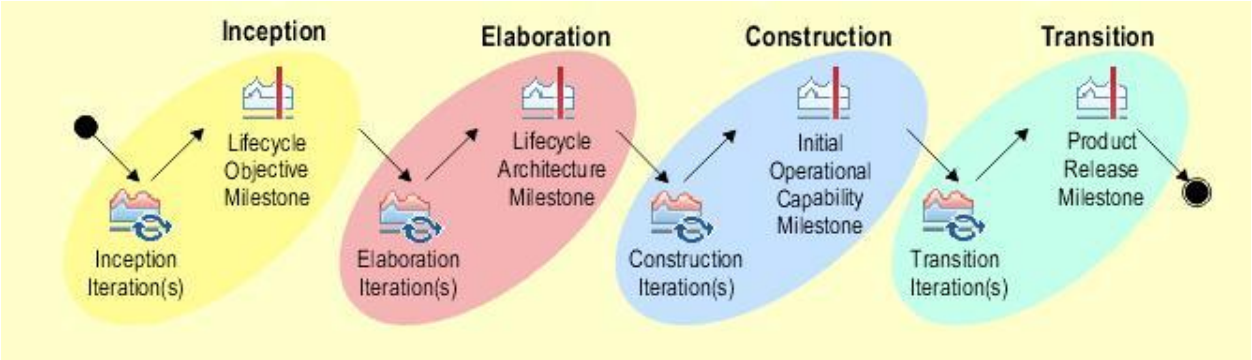


Figura 1 Fases de OpenUP

CONCEPCIÓN

En esta fase se definen el alcance y los objetivos del proyecto, se intenta lograr además una concurrencia entre los stakeholders y los objetivos del ciclo de vida del proyecto. **[13]**

ELABORACIÓN

En esta fase se definen los riesgos arquitectónicos más significativos y se establece la línea base de la arquitectura además de dejar todo bien sentado para continuar con el desarrollo del software. **[14]**

CONSTRUCCIÓN

Esta fase se enfoca en el diseño, implementación y las pruebas para desarrollar un sistema completo basado en la arquitectura definida en la fase anterior. **[15]**

TRANSICIÓN

El propósito de esta fase es verificar que el software está listo para entregarse. **[16]**

De todas las metodologías mencionadas anteriormente se aplicará la antes expuesta en detalles por ser la que más se identifica con el proceso de desarrollo de la aplicación, además existe una mayor posibilidad de tener éxito usándose en proyectos pequeños, mediante un ciclo iterativo son detectados los errores rápidamente, se limita sólo a iteraciones, documentación y diagramas

necesarios, permitiendo así un proceso de desarrollo ágil y la perspectiva de esta metodología va centrada al cliente y sus iteraciones son cortas.

ROLES, ACTIVIDADES Y ARTEFACTOS

La metodología seleccionada para el desarrollo de la aplicación cuenta con una serie de roles, actividades y artefactos **[17],** pero debido a su flexibilidad para el proceso de desarrollo del software, fueron seleccionados los siguientes:

Analista: Las personas en este rol representan al cliente y los usuarios finales involucrados, obteniendo información desde los stakeholders para entender el problema a ser resuelto y capturar y ajustar las prioridades para los requerimientos. Ver Figura 2:

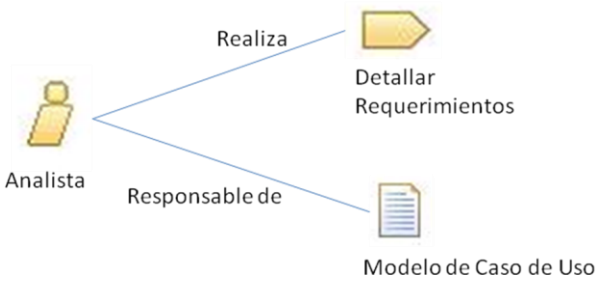


Figura 2 Analista, Actividad que realiza y Artefacto que genera.

Detallar Requerimientos: Esta tarea describe como detallar uno o varios requerimientos para el sistema.

Modelo de Caso de Uso: Este artefacto captura un modelo de las funciones del sistema y su ambiente, y sirve como un contrato entre el cliente y los desarrolladores.

Arquitecto: Este rol es el responsable de diseñar la arquitectura del software, la cual incluye tomar las principales decisiones técnicas que condicionan globalmente el diseño y la implementación del proyecto. Ver figura 3.

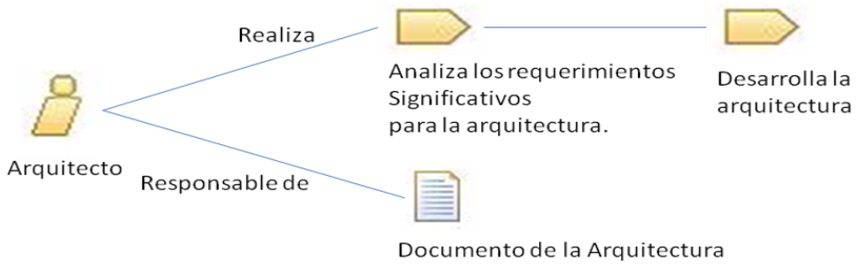


Figura 3 Arquitecto, Actividades que realiza y Artefacto que genera.

Analiza los requerimientos significativos para la arquitectura: Analiza los requerimientos arquitectónicamente significativos y define una arquitectura candidata para el sistema. Define el patrón arquitectónico, mecanismos claves, y donde es aplicable, modelando convenciones para el sistema

Desarrolla la Arquitectura: Toma decisiones concretas sobre la arquitectura para proporcionar la guía y la dirección para la iteración en el trabajo a desarrollar.

Documento de la Arquitectura: El documento de la Arquitectura describe el diseño del desarrollo de software. Este contiene el fundamento, suposiciones, explicaciones e implicaciones de las decisiones que fueron tomadas en la formación de la arquitectura.

Desarrollador: La persona en este rol es responsable de desarrollar una parte del sistema, incluyendo diseñar esta para que se ajuste a la arquitectura, posiblemente definir el prototipo de la interfaz de usuario y entonces implementar, hacer pruebas unitarias e integrar los componentes que son parte de la solución. Ver Figura 4.

Figura 4 Desarrollador, Actividades que realiza y Artefactos que genera.

Diseña la Solución: Identifica los elementos y diseña las interacciones, el comportamiento, relaciones, y datos necesarios para realizar algunas funcionalidades. Muestra visualmente un diseño para ayudar en la solución del problema y la comunicación de la solución.

Implementa la Solución: Implementa el código fuente para proporcionar nuevas funcionalidades o arreglar defectos.

Ejecuta pruebas de desarrollador: Realiza las pruebas de los componentes individuales del software para verificar que sus estructuras internas trabajen como se esperaba.

Versión operacional: Este artefacto es una versión operacional del sistema o la parte del sistema que demuestra un subconjunto de las capacidades que serán proporcionadas en el producto final.

Documento de Diseño: Este artefacto describe la realización de funcionalidades requeridas por el sistema en términos de componentes y sirve como una abstracción del código fuente.

Código Fuente: Este artefacto contiene archivos de código del software, ficheros de datos, y archivos de apoyo así como los archivos de ayuda en línea que representan las partes “crudas” de un sistema que puede ser construido.

1.5.2. HERRAMIENTAS CASE.

Se puede definir a las Herramientas CASE (Computer Assisted System Engineering) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. **[18]**

Dentro de las más usadas en la universidad se encuentran Rational Rose y Visual Paradigm. Rational Rose es una herramienta muy buena para el uso en el proceso de desarrollo de un software; sin embargo su uso fue descartado debido a que es una herramienta propietaria. Por otra parte se tiene a Visual Paradigm que es una herramienta muy potente en la visualización y diseño del software para lo cual utiliza UML como lenguaje de modelado. Se integra con las siguientes

herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic. Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal **[19]**, además es una herramienta multiplataforma. Por estas razones fue seleccionada como herramienta case, además que el trabajo en ella es mucho más sencillo que en Rational Rose. Se hará uso de esta herramienta Case en su versión 6.4.

1.5.3. LENGUAJE DE MODELADO DE SOFTWARE.

UML es actualmente uno de los lenguajes de más renombre a nivel mundial, utilizado para el modelado y construcción de sistemas de software. Este lenguaje es utilizado para gestionar sistemas, permitiendo detallar modelos de software a través de los procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. **[20]**

Surgió en 1994 como la combinación de todos los diseños anteriores permitiendo que diseñadores diferentes modelaran diseños disímiles y cada uno comprendiera los diseños de otro. Es importante destacara además que UML no es para describir métodos y procesos sino para especificarlos, este lenguaje será usado en su versión 2.0.

1.5.4. LENGUAJES DE PROGRAMACIÓN.

En el desarrollo de aplicaciones de software en la programación orientada a objetos tenemos lenguajes que han sido considerablemente usados, de los que podemos mencionar C, C++, C# y java.

C es un lenguaje de propósito general y alto nivel, es conciso, sencillo, portable **[21]**; pero presenta algunas desventajas de las cuales se pudiera mencionar que es poco estricto en la comprobación del tipo de datos, no anida funciones lo que dificulta la estructuración y la abstracción de datos, además presenta incertidumbre en la evaluación de parámetros y las listas de expresiones **[22]**. Por otra parte C++ presenta características como herencia múltiple, sobrecarga de operadores

y funciones, derivación, funciones virtuales, plantillas, gestión de excepciones **[23]** pero presenta desventajas notables, este es compilado y sólo genera un ejecutable para una plataforma en específico, es híbrido e inseguro, **[24]** además este tiene un completamiento de código pobre haciendo el trabajo para el programador más difícil.

También se tiene C# que es un lenguaje sencillo, moderno, con gestión automática de memoria, seguridad de tipos, instrucciones seguras, eficiente, compatible. **[25]** Este tiene una desventaja frente a java, y es que no es portable; aunque actualmente se desarrollan proyectos que contribuyan a la portabilidad de C# todavía quedan deficientes frente a Java; este posee una gran cantidad de herramientas, aplicaciones, frameworks y librerías libres que apoyan y facilitan el desarrollo sobre este lenguaje.

Java además de la característica antes mencionada y que lo diferencia de C#; posee otras de igual manera que lo hacen un lenguaje potente, dígase que es un lenguaje simple, distribuido, robusto, posee una arquitectura neutral, es seguro, interpretado, multithreaded (multihilos), dinámico. **[26]**

De esta manera y por las ventajas antes mencionadas que aporta se selecciona java como lenguaje de programación para el desarrollo de la aplicación.

1.5.5. Entorno de Desarrollo.

Existen varios tipos de entornos de desarrollo integrado (IDE) para el lenguaje de programación java, podemos mencionar dos que son los más usados en la UCI, Eclipse y NetBeans.

Eclipse es una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en plug-ins. Además, Eclipse da soporte a todo tipo de proyectos que abarcan desde el ciclo de vida del desarrollo de aplicaciones, incluyendo soporte para modelado. Este tiene como características principales que es un editor universal, compilación incremental de código, modifica e inspecciona valores de variables, avisa de los errores cometidos mediante una ventana secundaria y depura código que resida en una maquina remota. También se pueden mencionar como otras características que contiene un editor de texto, posee resaltado de sintaxis,

compilación en tiempo real, pruebas unitarias con Junit, control de versiones con CVS, tiene asistentes (wizards) y plug-ins. **[27]**

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito, sin restricciones de uso. **[28]**

Este en su versión 6.0, usada para el desarrollo de la aplicación, presenta algunas ventajas que a continuación se mencionan:

- Mejoras en el Editor de Código.
- Fácil actualización e instalación.
- Enlazar datos fácilmente con el Swing GUI.
- Mejoras en la Plataforma Netbeans.**[29]**

Netbeans se selecciona como entorno de desarrollo pues está más enfocado en el desarrollo de aplicaciones de escritorio, la cual es el tipo a desarrollar.

1.6. CONCLUSIONES.

En este capítulo se explicó porque será utilizado el Open UP/Basic como proceso de desarrollo de software de la aplicación y que será representado en la herramienta CASE Visual Paradigm 6.4 a través del lenguaje de modelado de software UML 2.0. Igualmente se selecciona el lenguaje java y se define que la aplicación será implementada en el entorno de desarrollo NetBeans 6.0.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. INTRODUCCIÓN.

En este capítulo se define lo que debe hacer la aplicación a través de los requerimientos funcionales y no funcionales, se realiza el diagrama de casos de uso del sistema, así como las descripciones de los casos de uso del mismo.

2.2. ACTOR DEL SISTEMA.

Los actores del sistema son personas que no son parte del mismo que pueden intercambiar información con él y que pueden ser un recipiente pasivo de información. De manera que en la situación que se desarrollará la aplicación fue encontrado el siguiente actor:

Nombre del Actor	Descripción
Administrador del Sistema	➢ Es la persona que trabajará directamente con la aplicación. ➢ Encargado de generar el Servicio Web

2.3. REQUISITOS DE LA APLICACIÓN.

Los requerimientos son una descripción de las necesidades de un producto. El objetivo principal de estos es identificar y documentar lo que se necesita en realidad, de manera que el equipo de desarrollo pueda comprender fácilmente lo que se necesita. Estos pueden clasificarse en funcionales y no funcionales. **[30]**

2.3.1. REQUISITOS FUNCIONALES.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. La aplicación a desarrollar cuenta con los siguientes requisitos:

R1: Gestionar conexión con Origen de Datos.

R1.1: Agregar conexión.

R1.2: Eliminar conexión.

R2: Crear Clases.

R3: Gestionar Métodos.

R3.1: Agregar Método.

R3.2: Modificar Método.

R3.3: Eliminar Método.

R4: Gestionar parámetro.

R4.1: Agregar parámetro.

R4.2: Eliminar parámetro.

R5: Guardar Fichero.

R6: Cargar Fichero.

R7: Generar Servicio Web.

2.3.2. REQUISITOS NO FUNCIONALES.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estos no son más que características que hacen el producto atractivo, rápido, usable o confiable.

La aplicación cuenta con los siguientes requisitos no funcionales:

Funcionalidad: A pesar de que el sistema podrá ser usado en un tiempo breve por aquellos usuarios que se encuentren capacitados en el trabajo con Servicios Web y base de datos, se debe realizar un adiestramiento previo con el objetivo de que puedan detectarse errores (en caso de que existan) y posibilitar la familiarización con la herramienta.

Apariencia o Interfaz externa: Se precisa una interfaz amigable, comprensible, interactiva, fácil de usar, profesional, clara y sencilla.

Usabilidad: El sistema podrá ser usado por aquellos usuarios que posean conocimientos básicos de Servicios Web además del trabajo con base de datos.

Soporte:

***Mantenimiento**:* El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

***Portabilidad**:* El sistema debe ser multiplataforma (ser capaz o caracterizarse por poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas).

Confiabilidad:

Tiempo medio de reparación: La reparación del sistema en caso de surgir fallas en el mismo debe realizarse en el menor tiempo posible, poniendo todos los esfuerzos en función de que no supere las 48 horas.

Ayuda y Documentación: El sistema constará con un manual de usuario donde esté presente la documentación básica que posibilite comprender su funcionamiento y las funcionalidades generales a tener en cuenta para garantizar la utilización del mismo de manera eficiente.

Software: Para el uso del sistema es necesario tener instalado la máquina virtual de java 1.5 o una versión superior y un servidor de MySQL 5.06 o superior.

Hardware:

- Se debe contar con 512 MB de memoria RAM como mínimo.
- Procesadores Pentium IV.

2.4. DIAGRAMA DE CASO DE USO DEL SISTEMA.

Los casos de uso son fragmentos de funcionalidad que el sistema ofrece y además son la manera en la que los actores lo usan. De manera más específica los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario.

Un diagrama de casos de usos del sistema está compuesto por actores, casos de uso y las relaciones que se establecen entre ellos. Ver Figura 5.

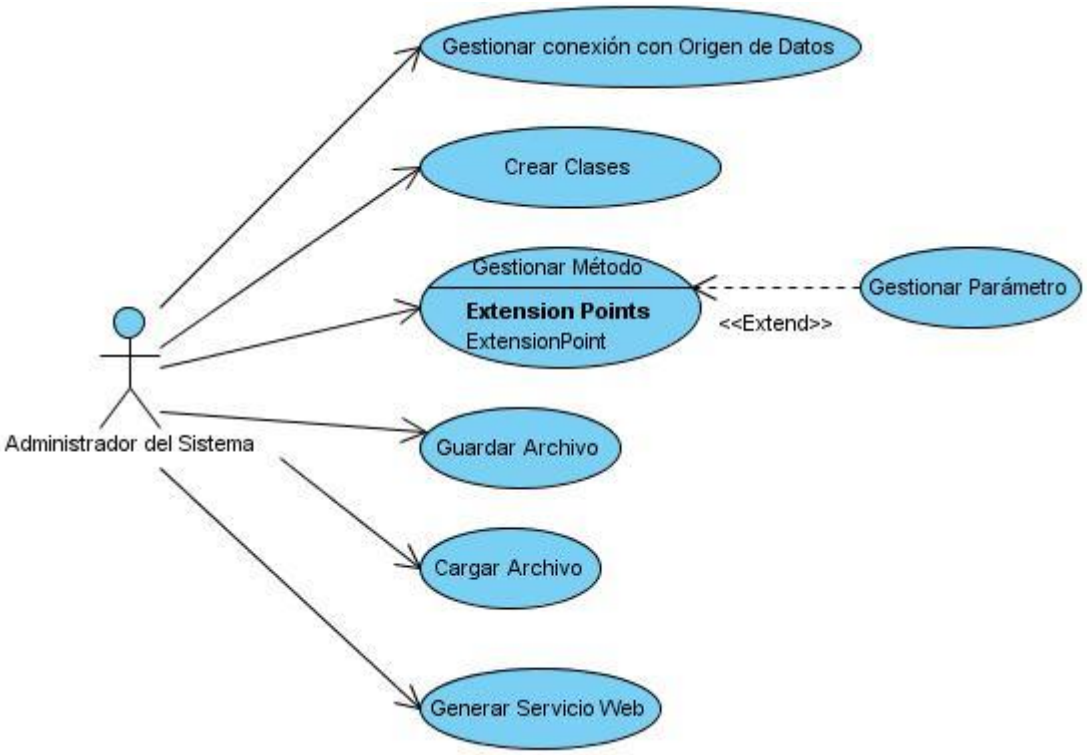


Figura 5 Diagrama de Casos de Uso del Sistema

2.5. DESCRIPCIÓN DE CASOS DE USO DEL SISTEMA.

2.5.1. DESCRIPCIÓN DEL CASO DE USO: GESTIONAR CONEXIÓN CON ORIGEN DE DATOS.

Caso de Uso:	**Gestionar conexión con Origen de Datos.**
Actores:	Administrador del sistema
Resumen:	El caso de uso inicia cuando el Administrador del sistema decide agregar una conexión o Eliminarla para administrar las conexiones que se usarán para generar el Servicio Web.
Referencia:	1. R1 2. R1.1 3. R1.2

	4. R1.3
CU asociados:	
Precondiciones:	1. El administrador del sistema debe conocer datos de autenticación de origen de datos.

Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del sistema selecciona comenzar para gestionar las conexiones.	1.1- El sistema muestra las opciones: Agregar Conexión y Eliminar Conexión.
2- El Administrador selecciona una de las opciones presentadas.	2.1- Si selecciona "Agregar Conexión" ver sección **Agregar Conexión.** - Si selecciona "Eliminar Conexión" ver sección **Eliminar Conexión.**
Sección "Agregar Conexión"	
3- El Administrador selecciona el tipo de conexión que desea agregar.	3.1- El sistema muestra una ventana en la que pide llenar los datos de la conexión a adicionar (Host, Nombre de la Base de Datos, Usuario, Contraseña, Nombre de la Conexión y Puerto (en caso de que sea necesario)).
4- El Administrador del sistema inserta los datos de la conexión a agregar.	
5- El Administrador del sistema selecciona la opción de agregar la conexión.	5.1- El sistema verifica que los datos sean correctos.
	5.2- El sistema agrega la conexión.
Sección "Eliminar Conexión"	
6- El administrador del sistema elige la conexión a borrar y luego selecciona la opción de eliminar la conexión.	6.1- El sistema elimina la conexión.
Flujos Alternos	
Sección "Agregar Conexión"	

4- El administrador del sistema inserta de forma incorrecta los datos.	4.1- El sistema muestra un mensaje de error de conexión. Ir al paso 3.1.
4- El administrador del sistema deja campos en blanco.	4.1- El sistema muestra el mensaje: "Rellene todos los campos para continuar". Ir al paso 3.1.
Sección "Eliminar Conexión"	
6- El administrador del sistema desea eliminar una conexión sin antes seleccionarla o sin existir ninguna insertada.	6.1- El sistema muestra el mensaje: "Debe marcar la conexión que desea eliminar". Ir al paso 6.
Poscondiciones	Queda creada una lista de conexiones.

2.5.2. Descripción del Caso de Uso: Crear Clases.

Caso de Uso:	Crear Clases
Actores:	Administrador del Sistema
Resumen:	El caso de uso se inicia cuando el administrador del sistema desea mapear las tablas para crear clases para utilizarlas en el Servicio Web.
Referencia:	1. R2 2. R3(inclusión)
CU asociados:	
Precondiciones:	1. El administrador del sistema debe estar conectado con al menos un origen de datos.
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador del sistema se dispone a crear las clases.	1.1. El sistema muestra una ventana donde da la opción de seleccionar las conexiones para

	mostrar sus tablas
2. El administrador selecciona la conexión de la que quiere seleccionar tablas	2.1. El sistema muestra las tablas de la conexión seleccionada.
3. El usuario va seleccionando las tablas que necesitará para la creación de las clases o las selecciona todas a las vez, también puede deseleccionarla una, o todas a la vez.	3.1. El sistema muestra las tablas.
4. El usuario selecciona la opción siguiente para mapear las tablas y convertirlas en clases.	4.1. El sistema crea las clases.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El administrador del sistema intenta adicionar una tabla sin haberla seleccionado antes	3.1. El sistema muestra un mensaje "Seleccione la tabla que desea agregar". Ir al paso 1.1
3. El administrador del sistema intenta adicionar todas las tablas sin haber mostrado antes las tablas de una conexión.	3.1. El sistema muestra un mensaje "No hay tablas que agregar" Ir al paso 1.1
3. El administrador del sistema selecciona la opción de quitar alguna tabla sin haberla seleccionado.	3.1. El sistema muestra un mensaje. "Seleccione la tabla que desea eliminar". Ir al paso 2.1.
3. El administrador del sistema desea quitar todas las tablas sin existir tablas seleccionadas.	3.1. El sistema muestra un mensaje. "No hay tablas que eliminar". Ir al paso 2.1.
Poscondiciones:	Quedan creadas las clases que se usarán para generar el Servicio Web.

2.5.3. DESCRIPCIÓN DEL CASO DE USO: GESTIONAR MÉTODOS.

Caso de Uso:	**Gestionar Métodos**
Actores:	Administrador del sistema
Resumen:	El caso de uso inicia cuando el Administrador del sistema decide agregar un método, Modificar un método o Eliminar métodos para controlar los métodos que tendrá el Servicio Web.
Referencia:	5. R3 6. R3.1 7. R3.2 8. R3.3 9. R4 (inclusión)
CU asociados:	
Precondiciones:	2. El administrador del sistema debe haber conectado la aplicación con un origen de datos. 3. El administrador del sistema debe haber creado las clases.

Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del sistema selecciona continuar para gestionar los métodos.	1.1- El sistema muestra varias opciones entre las que se encuentran: Agregar Método, Modificar Método y Eliminar Método.
2- El Administrador selecciona una de las opciones presentadas.	2.1- Si selecciona "Agregar Método" ver sección **Agregar Método.** - Si selecciona "Modificar Método" ver sección **Modificar Método.**

	- Si selecciona “Eliminar Método” ver sección **Eliminar Método.**
Sección “Agregar Método”	
3- El Administrador selecciona la opción “Agregar Método”.	3.1- El sistema muestra una ventana en la que pide llenar los datos del método a adicionar (Nombre del Método, Tipo de Dato, Nombre de Variable de Retorno, Descripción del Método).
4- El Administrador del sistema inserta los datos del método a llenar.	
5- El Administrador del sistema selecciona la opción de insertar la consulta y prueba si esta es correcta	5.1- El sistema verifica que la consulta insertada sea correcta y muestra un mensaje “La consulta se ejecutó correctamente”.
6- Ver caso de uso extendido “**Gestionar parámetro**”.	
7- El administrador del sistema selecciona agregar método.	7.1- El sistema verifica que los datos entrados sean correctos
	7.2- El sistema agrega el método.
Sección “Modificar Método”	
8- El administrador del sistema selecciona el método a modificar y luego la opción “Modificar Método”.	8.1- El sistema muestra una ventana con los datos del método a modificar. (Nombre del Método, Tipo de Dato, Nombre de Variable de Retorno, Descripción del Método, Consulta, Parámetros *(en caso de tenerlos)*).
9- El Administrador del sistema cambia los datos necesarios.	9.1- Si la modificación fue correcta el sistema guarda los cambios realizados.
Sección “Eliminar Método”	
10- El administrador del sistema marca el método a borrar y luego selecciona la opción de eliminar el método.	10.1- El sistema elimina el método.
Flujos Alternos	

Sección “Agregar Método”	
4- El administrador del sistema inserta de forma incorrecta los datos o deja los espacios en blanco.	4.1- El sistema muestra en mensaje “Debe insertar datos correctos”. Ir al paso 3.1.
Sección “Modificar Método”	
9- El administrador modifica de manera incorrecta los datos del método.	9.1- El sistema muestra en mensaje “Debe insertar datos correctos”. Ir al paso 8.1.
Sección “Eliminar Método”	
10- El administrador desea eliminar un método sin haberlo seleccionado o sin existir alguno insertado.	10.1- El sistema muestra en mensaje “Debe marcar el método que desea eliminar”. Ir al paso 10.
Poscondiciones	Queda creada la lista de métodos para generar el Servicio Web.

2.5.4. DESCRIPCIÓN DEL CASO DE USO: GESTIONAR PARÁMETRO

Caso de Uso:	**Gestionar parámetro**
Actores:	Administrador del sistema
Resumen:	El caso de uso inicia cuando el Administrador del sistema decide agregar un parámetro o Eliminar parámetros para controlar los parámetros que tendrá un método.
Referencia:	1. R4 2. R4.1 3. R4.2
CU asociados:	
Precondiciones:	1. El usuario del sistema debe haber insertado conexiones 2. Deben estar creadas las Clases.
Flujo Normal de Eventos	
Sección “General”	
Acción del Actor	**Respuesta del Sistema**
1- El Administrador del sistema selecciona	1.1- El sistema muestra las opciones:

la opción parámetros del método.	Agregar Parámetro y Eliminar Parámetro.
2- El Administrador selecciona una de las opciones presentadas.	2.1- Si selecciona “Agregar Parámetro” ver sección **Agregar Parámetro.** - Si selecciona “Eliminar Parámetro” ver sección **Eliminar Parámetro.**
Sección “Agregar Parámetro”	
	2.2- El sistema muestra una ventana en la que pide llenar los datos del parámetro a adicionar (Nombre del parámetro y el tipo de dato).
3- El Administrador del sistema inserta los datos del parámetro a agregar.	
4- El Administrador del sistema selecciona la opción de agregar el parámetro.	4.1- El sistema verifica que los datos sean correctos.
	4.2- El sistema agrega el parámetro.
Sección “Eliminar Parámetro”	
5- El administrador del sistema elige el parámetro a suprimir y luego selecciona la opción de eliminar el método.	5.1- El sistema elimina el método.
Flujos Alternos	
Sección “Agregar Parámetro”	
3- El administrador del sistema inserta de forma incorrecta los datos o deja espacios en blanco.	3.1- El sistema muestra un mensaje “Debe insertar datos correctos”. Ir al paso 2.1.
Sección “Eliminar Parámetro”	
5- El administrador del sistema selecciona la opción de eliminar el método si haberlo seleccionado.	5.1- El sistema muestra un mensaje “Debe marcar el parámetro que desea eliminar”. Ir al paso 5.
Poscondiciones	Queda creada una lista de parámetros.

2.5.5. DESCRIPCIÓN DEL CASO DE USO: GUARDAR FICHERO.

Caso de Uso:	Guardar Fichero
Actores:	Administrador del Sistema
Resumen:	El caso de uso se inicia cuando el administrador del sistema desea guardar los datos procesados en el sistema.
Referencia:	1. R5.
CU asociados:	
Precondiciones:	1. El usuario debe estar conectado al origen de datos.
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador del sistema selecciona la opción de guardar los datos entrados.	1.1. El sistema muestra la ventana para definir el nombre del fichero y la dirección a guardar.
2. El administrador del sistema inserta el nombre y selecciona la dirección donde será guardado el fichero.	2.1. El sistema guarda el fichero.
Poscondiciones:	Se guarda un fichero.

2.5.6. DESCRIPCIÓN DEL CASO DE USO: CARGAR FICHERO.

Caso de Uso:	Cargar Fichero
Actores:	Administrador del Sistema
Resumen:	El caso de uso se inicia cuando el administrador del sistema desea cargar un fichero.
Referencia:	1. R6.
CU asociados:	
Precondiciones:	1. El usuario debe estar conectado al origen de datos.

	2. Debe existir un fichero creado.
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
2. El administrador del sistema selecciona la opción de cargar un fichero.	a. El sistema muestra la ventana para cargar el fichero.
3. El administrador del sistema selecciona la dirección donde se encuentra el fichero.	a. El sistema carga el fichero.
Poscondiciones:	La aplicación carga la información guardada en el fichero.

2.5.7. DESCRIPCIÓN DEL CASO DE USO: GENERAR SERVICIO WEB.

Caso de Uso:	Generar Servicio Web.
Actores:	Administrador del Sistema
Resumen:	El caso de uso se inicia cuando el administrador del sistema desea generar un Servicio Web.
Referencia:	1. R7.
CU asociados:	
Precondiciones:	1. El usuario debe estar conectado al origen de datos. 2. Deben estar creados los métodos. 3. Deben estar creadas las Clases.
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1- El administrador del sistema decide generar el Servicio Web y selecciona la	1.1- El sistema muestra la ventana para insertar el nombre que tendrá el

opción siguiente.	Servicio Web.
2- El administrador inserta el nombre del Servicio Web.	2.1- El sistema verifica que el nombre entrado sea correcto.
3- El administrador del sistema selecciona la opción de mostrar el Servicio Web.	3.1- El sistema muestra una ventana de cómo ha quedado el Servicio Web hasta el momento.
4- El administrador del sistema revisa el Servicio Web, de quedar conforme con el resultado cierra la ventana y marca la opción de guardar el Servicio Web.	4.1- El sistema muestra la opción de buscar la dirección donde guardará el Servicio Web generado.
5- El administrador del sistema elige la dirección donde guardará el fichero y selecciona la opción finalizar.	5.2- El sistema guarda el Servicio Web en la dirección definida.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2- El administrador no escribe el nombre del Servicio Web.	2.1- El sistema muestra el mensaje “Dele un nombre correcto a su Servicio Web”. Ir al paso 1.1.
Poscondiciones:	Se genera el Servicio Web.

2.6. CONCLUSIONES.

Este capítulo muestra como se elaboraron los artefactos requeridos por Open UP para el desarrollo de la aplicación, funcionalidades que debe cumplir el software definido por los requisitos funcionales, el diagrama de casos de uso del sistema además de una descripción cada uno de los casos de uso y el actor que interactúa con ellos.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1. INTRODUCCIÓN.

A través de este capítulo se dará una breve descripción de lo que es arquitectura y además en cual estilo arquitectónico se basa el desarrollo de la aplicación se describirán los patrones de diseño además del diagrama de despliegue.

3.2. ARQUITECTURA Y ESTILOS ARQUITECTÓNICOS.

La programación se consideraba como un arte y se desarrollaba como tal, en los orígenes de la informática, siendo muy difícil para la mayoría de las personas, pero con el de cursar del tiempo se han ido desarrollando y descubriendo guías generales a través de las cuales se pueden resolver los problemas. A estas se les conoce como Arquitectura de software ya que se asemejan a los planos arquitectónicos de construcción para un edificio. Ellas indican estructura, funcionamiento e interacción entre las partes del software.

Arquitectura también se define como una vista estructural de alto nivel, ocurre muy tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales. **[31]**

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas.

Estilos arquitectónicos:

- Estilos de Flujo de Datos
  - Tubería y filtros

- Estilos Centrados en Datos
  - Arquitecturas de Pizarra o repositorio

- Estilos de llamada y retorno
    - Modelo Vista Controlador (MVC)
    - Arquitectura en capas
    - Arquitecturas orientadas a objetos
    - Arquitecturas basadas en componentes

- Estilos de código móvil
    - Arquitectura de máquinas virtuales

- Estilos heterogéneos
    - Sistemas de control de procesos
    - Arquitecturas basadas en atributos

- Estilos Peer-to-Peer (punto a punto)
    - Arquitecturas basadas en eventos
    - Arquitecturas orientadas a servicios
    - Arquitecturas basadas en recursos.**[31]**

El patrón de arquitectura que se usará para el desarrollo de la aplicación será uno que se encuentra dentro del estilo de llamada y retorno, **arquitectura en capas**.

3.3. PATRÓN DE ARQUITECTURA EN CAPAS.

La **arquitectura en capas** es un estilo en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.

La arquitectura en capas tiene como principal objetivo que queden separadas la lógica del negocio de la del diseño y tiene como ventaja principal que el desarrollo se realiza en diversos niveles y en caso de tener necesidad de cambiar algo solo se ejecuta en el nivel requerido. También distribuye el trabajo del desarrollo de la aplicación por niveles para que cada equipo se centre en el

trabajo que le fue asignado y se abstraiga del resto; además las tareas están bien definidas por niveles en esta arquitectura.

Capa de Aplicación: Esta contiene todas las clases de interfaz de usuarios que representan las pantallas de la aplicación que el usuario ve. Además depende de la capa de lógica de Negocio y de la capa de acceso a datos.

Capa de Lógica del Negocio: Esta tiene todas las clases controladoras que representan el comportamiento de la aplicación, según los casos de uso. Al mismo tiempo representa la frontera del cliente con la capa de acceso a datos. La capa de Lógica del Negocio depende de la capa de acceso a Datos.

Capa de Acceso a Datos: La capa Intermedia apoya el acceso al Sistema Gestor de Base de Datos (SGBD); esta recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio. **[32]**

La vista lógica perteneciente a esta arquitectura se encuentra en el **Anexo 1**.

3.4. PATRONES DE DISEÑO.

Los patrones de diseño son utilizados para describir objetos y clases, los cuales se comunican con el fin de darle solución a un problema de diseño general, identificando clases, instancias, roles, colaboraciones y la distribución de responsabilidades. **[31]**

El uso de patrones de diseño para el desarrollo de la aplicación aporta ventajas mencionadas a continuación:

- Ofrecen una manera de reutilizar la experiencia de los desarrolladores, describiendo cómo solucionar los problemas que se presentan normalmente durante el desarrollo del producto,

- Se basan en la recopilación de los conocimientos de los especialistas en el desarrollo de software.

- Mediante los patrones de diseño es más probable no consumar los mismos errores.**[31]**

3.4.1. PATRONES DE DISEÑO GRASP

Los patrones GRASP son de gran relevancia, los mismos son utilizados para la descripción de los principios fundamentales de la concesión de responsabilidades a objetos, pero formulados en forma de patrones. **[33]**

Seguidamente serán mencionados algunos de estos patrones, además de uno adicional; evidenciando mediante un ejemplo donde fueron usados en el desarrollo del sistema.

3.4.1.1. PATRÓN GRASP: EXPERTO

Este patrón asigna la responsabilidad al experto en la información, en otras palabras, la responsabilidad es asignada a la clase que contiene la información necesaria. Estas responsabilidades son situadas en el diseño orientado a objetos. Además la encapsulación se mantiene pues los objetos para ejecutar las tareas usan su propia información. **[34]**

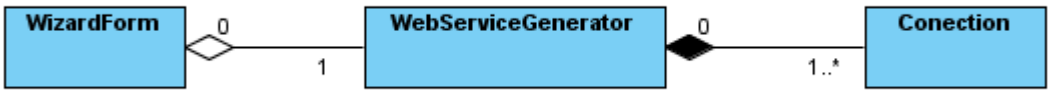


Figura 6 Ejemplo Patrón Experto – Bajo Acoplamiento

En la figura 6 se evidencia este patrón ya que la clase *WizardForm* tiene una instancia de *WebServiceGenerator*, la que tiene la responsabilidad de manipular la clase *Conection* en el momento que necesite acceder al origen de datos.

3.4.1.2. PATRÓN GRASP: ALTA COHESIÓN

El uso de este patrón permite que cada clase tenga como responsabilidad trabajar en una misma parte de la aplicación y sin que tenga mucha complejidad. En otras palabras una clase tiene definida una responsabilidad en un área específica que le permite colaborar con otras realizando así las tareas. **[33]**

Este patrón se ve evidenciado en el **anexo 1,** puesto que cada paquete contiene las clases especializadas en dicha función; por ejemplo en el paquete de lógica del negocio se encuentran las clases encargadas de las principales funcionalidades del sistema.

3.4.1.3. PATRÓN GRASP: BAJO ACOPLAMIENTO

Este patrón consiste en asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. **[33]** Ver como ejemplo la figura 6 que muestra la relación que existe entre las clases *WizardForm, WeServiceGenerator y Conection***.**

3.4.1.4. PATRÓN GRASP: CREADOR

El uso de este patrón en el diseño del sistema permite que una clase tenga la responsabilidad de crear una instancia en otra. Además guía en la asignación de las responsabilidades relacionadas en la creación de objetos. **[34]**

El uso de este patrón se ve presente en la clase controladora *WebServiceGenerator* que es la encargada de crear grupos de conections, methods y class. Ver figura 7.

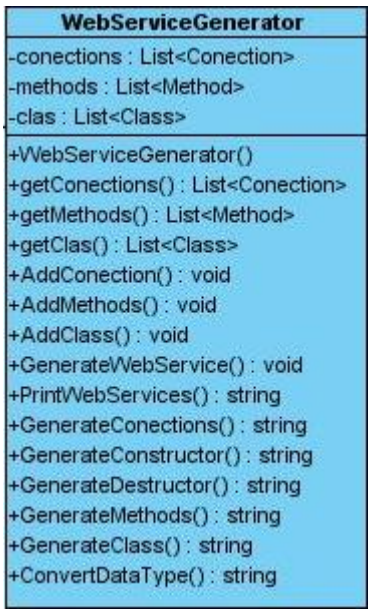


Figura 7 Ejemplo Patrón Creador

3.4.1.5. PATRÓN GRASP ADICIONAL: POLIMORFISMO

Este patrón tiene como objetivo poner el mismo nombre a servicios en disímiles objetos, esto es cuando hay semejanza entre ellos o se relacionan de alguna manera. **[35]**

Este patrón se encuentra evidenciado entre las clases *Conection, MySqlConection* y *PostgreConection;* estas dos últimas tienen los mismos métodos que a través del polimorfismo tienen diferentes funcionalidades. Ver figura 8.

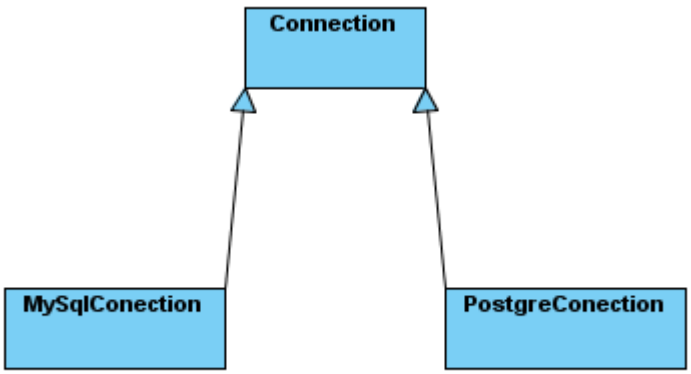


Figura 8 Ejemplo Patrón Polimorfismo

3.5. DIAGRAMA DE CLASES DEL DISEÑO

Las clases del diseño representan una abstracción de una o varias clases en la implementación del sistema, estas definen los objetos, con los cuales se implementan los casos de uso. En ellas se muestra la estructura estática del modelo, en particular, los elementos que existen como clases, su estructura interna y sus relaciones con otras clases; además muestran clases, interfaces, paquetes del diseño, subsistemas del diseño, sus relaciones y no muestran información temporal.

En los **Anexos 2,3 y 4** se muestran los diagramas de clases del diseño de las capas aplicación, lógica del negocio y acceso a datos, respectivamente.

3.6. MODELO DE DESPLIEGUE

Un diagrama de despliegue muestra cómo quedará desplegado un sistema, en otras palabras, los componentes, que se representan como nodos y las relaciones entre ellos, en términos de protocolos. Estos componentes pueden ser computadoras o dispositivos y los protocolos de comunicación pueden ser diversos en dependencia de la necesidad.

El diagrama de despliegue para la aplicación a desarrollar es el siguiente. Ver figura 9**.**

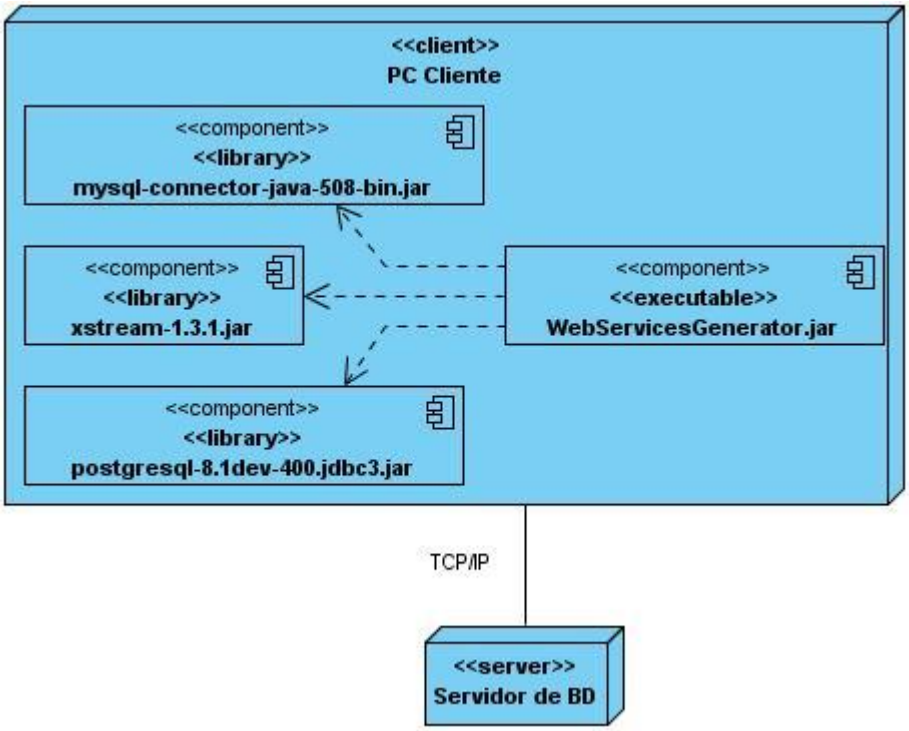


Figura 9 Diagrama de Despliegue

3.7. CONCLUSIONES

En este capítulo se hace una descripción de la arquitectura que se aplica al sistema, se elabora la vista lógica y los diagramas de clases del diseño; además se habla de los patrones de diseño usados y se muestra también el diagrama de despliegue.

CAPÍTULO 4: IMPLEMENTACIÓN

4.1. INTRODUCCIÓN

En el presente capítulo se evidencia como fue implementada la aplicación haciendo uso de componentes, para esto se enseña el diagrama de componentes. Se muestra además el código fuente de los principales métodos implementados, pantallas de la aplicación y una serie de pruebas exploratorias para la validación del software.

4.2. DIAGRAMA DE COMPONENTES

Un diagrama de componentes contiene generalmente componentes, interfaces y las relaciones que se establecen entre ellos. Además puede tener paquetes con el objetivo de agrupar los elementos del modelo. Además enseña la forma en que quedan organizados los componentes y sus dependencias lógicas, estos pueden ser ejecutables, binarios o código fuente. **[36]**

A continuación se muestra el diagrama de componentes, ver figura 10.

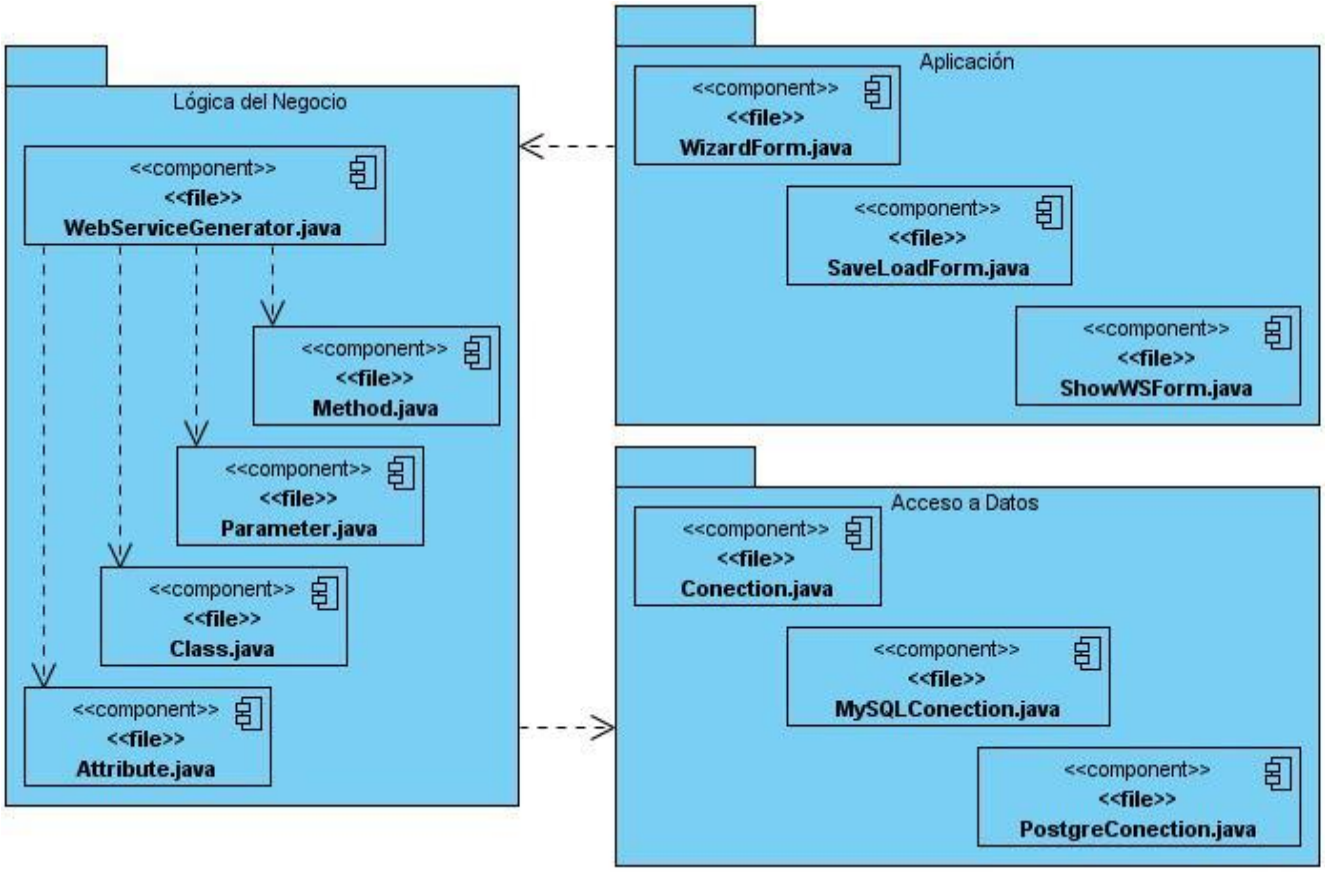


Figura 10 Diagrama de Componentes

4.3. CÓDIGO FUENTE

A continuación de manera detallada es explicado el código perteneciente a cada unas de las clases representadas en los diagramas de clases del diseño representados en los **Anexos 2 ,3 ,4.**

4.3.1. DESCRIPCIÓN DEL CÓDIGO DE LAS CLASES DE APLICACIÓN.

Las clases de aplicación son WizardForm.java, ShowWSForm.java y SaveLoadForm.java. La primera de las mencionadas es la encargada de manejar el código referente a lo visual del sistema. En esta clase se puede hablar del método *CreateDBTree()*, este tiene como objetivo mostrar de

forma jerárquica el contenido de cada una de las base de datos a las que el usuario se encuentre conectado, a continuación se muestra el código.

```
public void CreateDBTree() {
    DefaultMutableTreeNode root = new DefaultMutableTreeNode("Bases de Datos");
    DefaultTreeModel model = new DefaultTreeModel(root);
    for(int i = 0; i < web.getConections().size(); i++) {
      Conection con = web.getConections().get(i);
        DefaultMutableTreeNode dbName = new DefaultMutableTreeNode(con.getDbName());
        model.insertNodeInto(dbName, root, i);
        try {
          for(int j = 0; j < con.ObtainTables().size(); j++) {
            String table = con.ObtainTables().get(j);
            DefaultMutableTreeNode tableName = new DefaultMutableTreeNode(table);
            model.insertNodeInto(tableName, dbName, j);
            for(int k = 0; k < con.ObtainFields(table).size(); k++) {
              String[] fields = con.ObtainFields(table).get(k);
              DefaultMutableTreeNode fieldName = new DefaultMutableTreeNode(fields[0]+" -
- "+fields[1]);
              model.insertNodeInto(fieldName, tableName, k);
            }
          }
        }
        catch(SQLException sql) {
          JOptionPane.showMessageDialog(new JPanel(), sql.getMessage(), "Error de
Conexion", JOptionPane.ERROR_MESSAGE);
        }
      jTdb.setModel(model);
    }
  }
```

4.3.2. DESCRIPCIÓN DEL CÓDIGO DE LAS CLASES DE LA LÓGICA DEL NEGOCIO.

Las clases controladoras son *Attribute.java, Class.java, Method.java, Parameter.java WebServiceGenerator.java*; esta última clase tiene como objetivo crear instancias de las demás clases.

De todas las clases antes mencionadas la última contiene los métodos más significativos los cuales serán mostrados a continuación.

```
public void AddClass(List<String[]> listTables, List<String> listTable) {
    clas = new ArrayList<Class>();
    percent = -1;
    mapTable = "";
    mapField = "";
    Conection con = null;
    String className = "";
    String table = "";
    List<String[]> listFields = null;
    List<Attribute> listAttribute = null;
    List<String[]> listFK = null;
    try {
      for (int i = 0; i < getConections().size(); i++) {
        con = getConections().get(i);
        if (con instanceof PostgreConection) {
          listFK = ((PostgreConection) (con)).ObtainForeignKeys();
          for (int j = 0; j < listTables.size(); j++) {
            if (listTables.get(j)[1].equals(getConections().get(i).getConName())) {
              listAttribute = new ArrayList<Attribute>();
              listFields = new ArrayList<String[]>();
              table = listTables.get(j)[0];
              percent = ((j+1) * 100)/listTables.size();
              mapTable = "Mapeando tabla " + table; // para el progress bar
              System.out.println(percent);
              className = table.replaceFirst(table.substring(0, 1), table.substring(0,
1).toUpperCase());
              listFields = ((PostgreConection) (con)).ObtainFields(listTables.get(j)[0]);
              for (int k = 0; k < listFields.size(); k++) {
                String[] field = listFields.get(k);
                mapField = "Campo " + field[0]; // para el progress bar
                //System.out.println(mapField);
                Attribute attribute = null;
                boolean flag = false;
                for (int l = 0; l < listFK.size(); l++) {
                  //Comprueba que el campo es llave foranea
                  while (!listFK.isEmpty() && listFK.get(l)[2].equals(table) &&
listFK.get(l)[3].equals(field[0]) && listTable.contains(listFK.get(l)[0])) {
```

```
                        String attributeName = listFK.get(l)[0].replaceFirst(listFK.get(l)[0].substring(0, 1),
listFK.get(l)[0].substring(0, 1).toUpperCase());
                        String attributeDataType = listFK.get(l)[0].replaceFirst(listFK.get(l)[0].substring(0,
1), listFK.get(l)[0].substring(0, 1).toUpperCase()) + "[]";
                        attribute = new Attribute(attributeName, attributeDataType);
                        listAttribute.add(attribute);
                        if (l < listFK.size() - 1) {
                           l++;
                        } else {
                           break;
                        }
                     }
                     if (!listFK.isEmpty() && listFK.get(l)[0].equals(table) &&
listFK.get(l)[1].equals(field[0])) {
                        flag = true;
                     }
                  }
                  if (!flag || con.isPrimaryKey(table, field[0])) {
                     String attributeName = table + "." + field[0];
                     String attributeDataType = ConvertDataType(field[1]);
                     attribute = new Attribute(attributeName, attributeDataType);
                     if (con.isPrimaryKey(table, field[0])) {
                        listAttribute.add(0, attribute);
                     } else {
                        listAttribute.add(attribute);
                     }
                  }
               }
               Class classs = new Class(con.getConName(), className, listAttribute);
               clas.add(classs);
            }
         }
      } else {
         listFK = ((MySQLConection) (con)).ObtainForeignKeys();
         for (int j = 0; j < listTables.size(); j++) {
            con = getConections().get(i);
            if (listTables.get(j)[1].equals(getConections().get(i).getConName())) {
               listAttribute = new ArrayList<Attribute>();
               table = listTables.get(j)[0];
               percent = ((j+1) * 100)/listTables.size();
               mapTable = "Mapeando tabla " + table; // para el progress bar
               System.out.println(percent);
```

```
                className = table.replaceFirst(table.substring(0, 1), table.substring(0, 1).toUpperCase());
                listFields = ((MySQLConection) (con)).ObtainFields(listTables.get(j)[0]);
                for (int k = 0; k < listFields.size(); k++) {
                  String[] field = listFields.get(k);
                  mapField = "Campo " + field[0]; // para el progress bar
                  Attribute attribute = null;
                  boolean flag = false;
                  for (int l = 0; l < listFK.size(); l++) {
                    //Comprueba que el campo es llave foranea
                    while (!listFK.isEmpty() && listFK.get(l)[2].equals(table) &&
listFK.get(l)[3].equals(field[0]) && listTable.contains(listFK.get(l)[0])) {
                      String attributeName = listFK.get(l)[0].replaceFirst(listFK.get(l)[0].substring(0, 1),
listFK.get(l)[0].substring(0, 1).toUpperCase());
                      String attributeDataType = listFK.get(l)[0].replaceFirst(listFK.get(l)[0].substring(0,
1), listFK.get(l)[0].substring(0, 1).toUpperCase()) + "[]";
                      attribute = new Attribute(attributeName, attributeDataType);
                      listAttribute.add(attribute);
                      if (l < listFK.size() - 1) {
                        l++;
                      } else {
                        break;
                      }
                    }
                    if (!listFK.isEmpty() && listFK.get(l)[0].equals(table) &&
listFK.get(l)[1].equals(field[0])) {
                      flag = true;
                    }
                  }
                  if (!flag || con.isPrimaryKey(table, field[0])) {
                    String attributeName = field[0];
                    String attributeDataType = ConvertDataType(field[1]);
                    attribute = new Attribute(attributeName, attributeDataType);
                    if (con.isPrimaryKey(table, field[0])) {
                      listAttribute.add(0, attribute);
                    } else {
                      listAttribute.add(attribute);
                    }
                  }
                }
                Class classs = new Class(con.getConName(), className, listAttribute);
                clas.add(classs);
              }
            }
```

```
            }
        }
    } catch (Exception ex) {
        System.out.println("Mensaje: " + ex.getMessage());
    }
}
```

Este método tiene como objetivo que a partir de las tablas de una base de datos generar una clase PHP por cada una.

El método que se muestra a continuación tiene como objetivo guardar en el disco duro el Servicio Web una vez generado, y para eso se le pasan por parámetros el nombre, y la dirección a guardar.

```
public void GenerateWebService(String name, String location) {

    PrintWriter pw;
    try {
        pw = new PrintWriter(location + name + "WS.php");
        pw.print(PrintWebServices("lachi"));
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    }
}

public String PrintWebServices(String name) {
    String result = "<?php \n"
      + "/** \n"
      + " * Configuraciones para la ejecución del fichero. \n"
      + " */ \n\n"
      + "ini_set ( 'max_execution_time', '0' ); \n"
      + "ini_set ( 'memory_limit', '1000M' ); \n"
      + "ini_set ( 'soap.wsdl_cache', 0 ); \n\n"
      + GenerateClass() + "\n"
      + "class " + name + "WS { \n\n"
      + GenerateConections() + "\n"
      + "\t/** \n"
```

```
            + "\t * Constructor de la Clase " + name + "WS. \n"
            + "\t */ \n\n"
            + "\tpublic function __construct() { \n"
            + "\t\ttry { \n"
            + GenerateConstructor()
            + "\t\t} \n"
            + "\t\tcatch ( PDOException $e ) { \n"
            + "\t\t\t throw new SoapFault ( 'Mensaje', 'Problemas de connexión con la fuente de datos:' .
$e->getMessage () ); \n"
            + "\t\t } \n"
            + "\t} \n\n"
            + "\t/** \n"
            + "\t * Destructor de la clase de la fachada de " + name + "WS. \n"
            + "\t */ \n\n"
            + "\tpublic function __destruct() { \n"
            + GenerateDestructor()
            + "\t} \n\n"
            + GenerateMethods()

            + "} \n"
            + "?>";
        return result;
    }
```

El método mostrado anteriormente contiene el estándar de un Servicio Web en PHP, o sea el formato con que quedará al ser generado. Dentro de este se encuentra el llamado a los métodos *GenerateConections(), GenerateConstructor(), GenerateDestructor(), GenerateMethods() y GenerateClass()*, métodos que aportarán la información que resta al método *PrintWebServices(String name)* que después de pasado el parámetro del nombre del Servicio Web, lo generará completamente.

4.3.3. DESCRIPCIÓN DEL CÓDIGO DE LAS CLASES DE ACCESO A DATOS.

Las clases de acceso a datos son *Connection.java, MySQLConection.java, PostgreConection.java;* de ellas la clase *Connection.java* es padre de *MySQLConection.java* y *PostgreConection.java,* estas hechas para conectar a una base de datos en MySQL y PostgreSQL respectivamente. De estas clases se mostrarán los métodos más significativos y la descripción de los mismos.

En la clase *Connection.java* se encuentran los métodos abstractos:

```
public abstract List<String> ObtainTables()
public abstract List<String[]> ObtainFields(String table)
public abstract boolean isPrimaryKey(String table, String field)
public abstract List<String[]> ObtainForeignKeys()
public abstract boolean TestQuery(String query)
```

Cada uno de estos es redefinidos en los en las clases hijas *MySQLConection.java* y *PostgreConection.java.* Ahora se mostrarán ejemplos de estos métodos y para qué son usados, sólo de una clase, en este caso *PostgreConection.java*; puesto que son clases muy similares, solo se diferencian en las consultas que se realizan a cada una de las bases de datos.

El método a mostrar a continuación tiene el objetivo de obtener una lista de nombres de las tablas pertenecientes a la base de datos a la que se conectará la aplicación.

```
@Override
  public List<String> ObtainTables() throws SQLException {

    List<String> result = new ArrayList<String>();
    try {
      Statement stat = this.getConect().createStatement();
```

```
        ResultSet query = stat.executeQuery("SELECT table_name FROM information_schema.tables WHERE table_schema = 'public'");
        while (query.next()) {
          result.add(query.getString(1));
        }
        return result;
      } catch (SQLException sqle) {
        while (sqle != null) {
          //System.out.println("Mensaje: " + sqle.getMessage());
          //System.out.println("Vendor error: " + sqle.getErrorCode());
          throw new SQLException("Mensaje: " + sqle.getMessage() + "\n" + "Vendor error: " + sqle.getErrorCode());
          //sqle = sqle.getNextException();
        }
      }
      return result;
    }
```

El siguiente método es para obtener los campos de una tabla de las que se obtuvieron con la conexión a la base de datos.

```
      @Override
        public List<String[]> ObtainFields(String table) throws SQLException {

          List<String[]> result = new ArrayList<String[]>();
          try {
            Statement stat = this.getConect().createStatement();
            ResultSet query = stat.executeQuery("SELECT column_name, data_type FROM information_schema.columns WHERE table_name = '" + table + "'");
            while (query.next()) {
              String[] aray = new String[2];
              aray[0] = query.getString(1);
              aray[1] = query.getString(2);
              result.add(aray);
            }
            return result;
          } catch (SQLException sqle) {
            while (sqle != null) {
              //System.out.println("Mensaje: " + sqle.getMessage());
              //System.out.println("Vendor error: " + sqle.getErrorCode());
              throw new SQLException("Mensaje: " + sqle.getMessage() + "\n" + "Vendor error: " + sqle.getErrorCode());
              //sqle = sqle.getNextException();
```

```
            }
        }
        return result;
}
```

El método que se muestra a continuación es para probar si una consulta funciona correctamente, pasándole esta por parámetro.

```
    @Override
      public boolean TestQuery(String quer) throws SQLException {
        String result = " ";
        try {
          Statement stat = this.getConect().createStatement();
          ResultSet query = stat.executeQuery(quer);
           while(query.next()) {
            result = query.getString(1);
            if(result.equals(" "))
              return true;
            else
              return false;
          }
        } catch (SQLException sqle) {
          //while (sqle != null) {
          //System.out.println("Mensaje: " + sqle.getMessage());
          //System.out.println("Vendor error: " + sqle.getErrorCode());
          throw new SQLException("Mensaje: " + sqle.getMessage() + "\n" + "Vendor error: " +
sqle.getErrorCode());
          //sqle = sqle.getNextException();
          //}
        }
        return false;
      }
```

Seguidamente el método que se muestra devuelve una lista de arreglos de String con los datos de cada llave foránea de la base de datos.

```
@Override

public List<String[]> ObtainForeignKeys() throws SQLException {
   List<String[]> result = new ArrayList<String[]>();
   try {
      Statement stat = this.getConect().createStatement();
      ResultSet query = stat.executeQuery("SELECT FK.table_name, FKU.column_name, UK.table_name, UKU.column_name FROM Information_Schema.Table_Constraints AS FK INNER JOIN Information_Schema.Key_Column_Usage AS FKU " +
            "ON FK.constraint_type = 'FOREIGN KEY' AND FKU.constraint_name = FK.constraint_name INNER JOIN Information_Schema.Referential_Constraints AS RC ON RC.constraint_name = FK.constraint_name INNER JOIN " +
            "Information_Schema.Table_Constraints AS UK ON UK.constraint_name = RC.unique_constraint_name INNER JOIN Information_Schema.Key_Column_Usage AS UKU ON UKU.constraint_name = UK.constraint_name");
      while (query.next()) {
         String[] aray = new String[4];
         aray[0] = query.getString(1);
         aray[1] = query.getString(2);
         aray[2] = query.getString(3);
         aray[3] = query.getString(4);
         result.add(aray);
      }
      return result;
   } catch (SQLException sqle) {
      while (sqle != null) {
         //System.out.println("Mensaje: " + sqle.getMessage());
         //System.out.println("Vendor error: " + sqle.getErrorCode());
         throw new SQLException("Mensaje: " + sqle.getMessage() + "\n" + "Vendor error: " + sqle.getErrorCode());
         //sqle = sqle.getNextException();
      }
   }
   return result;
}
```

El siguiente método pasándole como parámetros la tabla y el campo, dice si este es llave primaria de dicha tabla.

```
@Override
public boolean isPrimaryKey(String table, String field) throws SQLException {
  String result = " ";
  try {
    Statement stat = this.getConect().createStatement();
    ResultSet query = stat.executeQuery("SELECT table_name, column_name, constraint_name FROM information_schema.constraint_column_usage WHERE table_name = '" + table + "' AND constraint_name = 'PK_" + table + "' AND column_name = '" + field + "'");
    while (query.next()) {
      result = query.getString(1);
      if(result.equals(" "))
        return false;
      else
        return true;
    }
  } catch (SQLException sqle) {
    while (sqle != null) {
      //System.out.println("Mensaje: " + sqle.getMessage());
      //System.out.println("Vendor error: " + sqle.getErrorCode());
      throw new SQLException("Mensaje: " + sqle.getMessage() + "\n" + "Vendor error: " + sqle.getErrorCode());
      //sqle = sqle.getNextException();
    }
  }
  return false;
}
```

4.4. VALIDACIÓN DEL SISTEMA.

Para validar el sistema se desarrollaron una serie de pruebas exploratorias que fueron aplicadas con el objetivo de verificar un correcto cumplimiento de las descripciones textuales de los casos de uso del sistema. A continuación se muestran las pruebas realizadas por cada uno de ellos.

4.4.1. PRUEBA AL CASO DE USO: GESTIONAR CONEXIÓN CON ORIGEN DE DATOS.

Para desarrollar las pruebas a este caso de uso contamos con los datos de conexión de dos bases de datos una en PostgreSQL y otra en MySQL a continuación se encuentran imágenes de las entradas de los datos y de la respuesta del sistema respectivamente. Ver Figura 11 y 12 para el caso de PostgreSQL. Ver Figura 13 y 14 para el caso de MySQL.

Figura 11 Vista para conectarse a PostgreSQL

Figura 12 Resultado de conexión a PostgreSQL

En esta figura se muestra insertada de forma correcta la conexión realizada a una base de datos, de esta misma forma para una conexión en MySQL como se muestra en la figura 14,

Figura 13 Vista para conectarse a MySQL

Figura 14 Resultado de conexión a MySQL

4.4.2. PRUEBA AL CASO DE USO: CREAR CLASES.

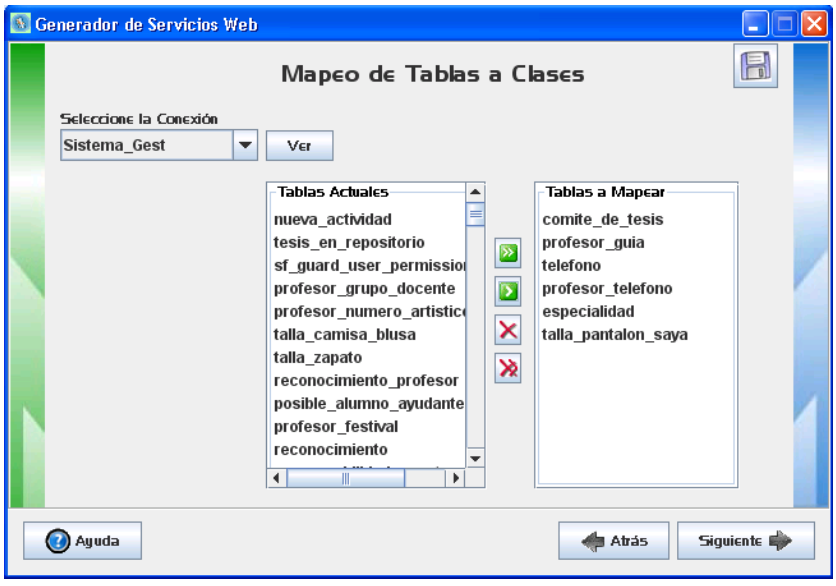


Figura 15 Vista Mapeo de tabla a Clases

Este caso de uso queda probado al tener todas las tablas seleccionadas y escoger la opción siguiente, pues es aquí donde son mapeadas las tablas y convertidas a clases. De manera que si el usuario puede pasar a la siguiente vista de la ventana se ha cumplido satisfactoriamente el caso de uso *Crear Clases*.

4.4.3. PRUEBA AL CASO DE USO: GESTIONAR MÉTODO.

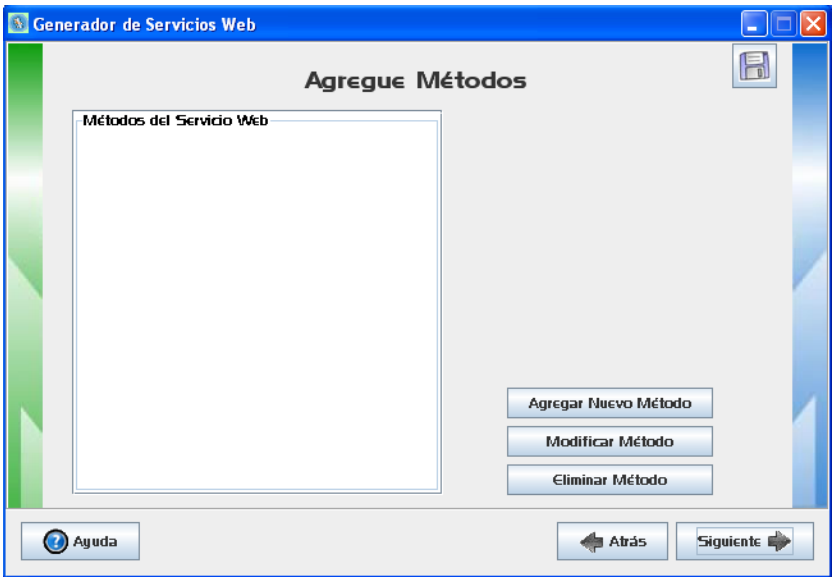


Figura 16 Vista Gestionar Métodos

En la figura 16 se muestra la forma en que se gestiona un método. Seguidamente veremos el flujo de eventos para cuando se quiere agregar un método.

Primeramente se insertan los datos del método, esto se ve representado en la figura 17.

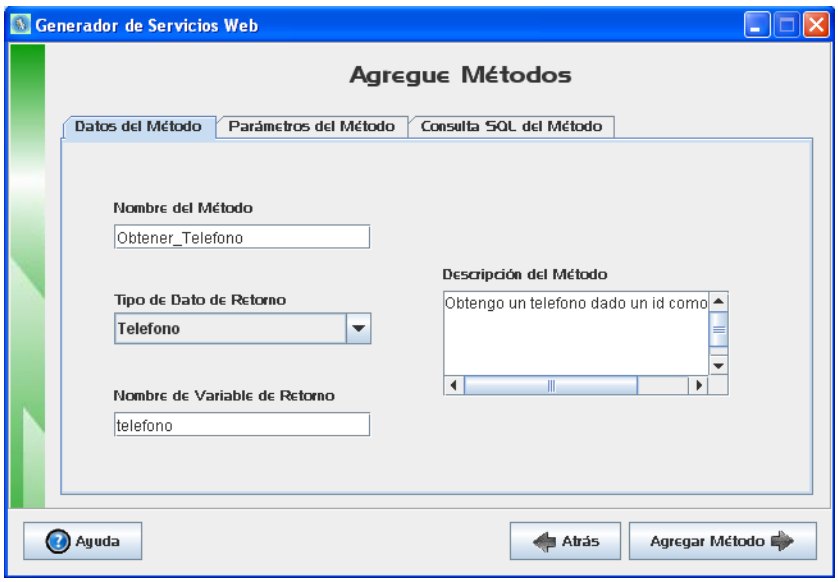


Figura 17 Vista Agregar Métodos - Datos del Método

4.4.3.1. PRUEBA AL CASO DE USO: GESTIONAR PARÁMETRO.

Antes de continuar con el flujo de eventos de agregar un método, debemos pasar por el caso de uso extendido Gestionar Parámetro, por este se pasa sólo en caso de ser necesario como así su tipo lo indica. Para efectuar una prueba al flujo de eventos de agregar un parámetro, se hará una prueba del mismo, mostrados en las figuras 18 y 19, mostrando en la figura 19 como se logra satisfactoriamente el resultado esperado.

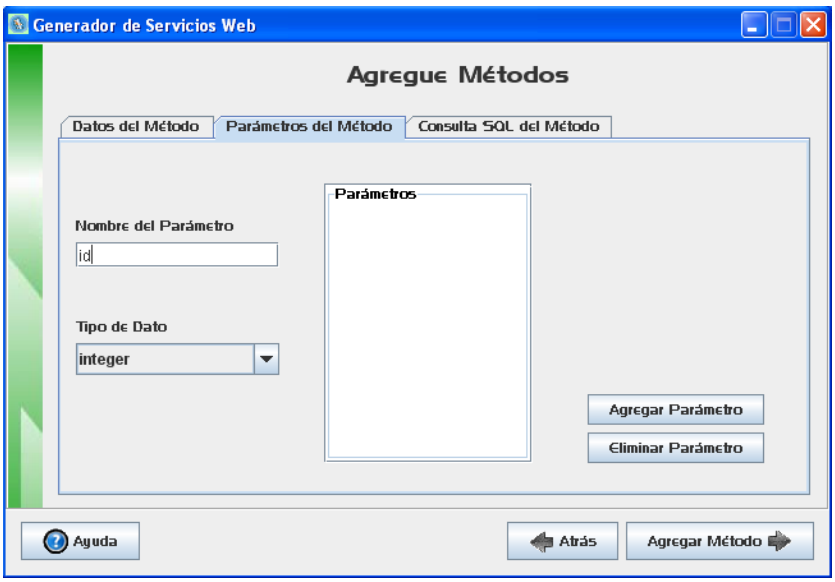


Figura 18 Vista Agregar Métodos - Parámetros del Método.

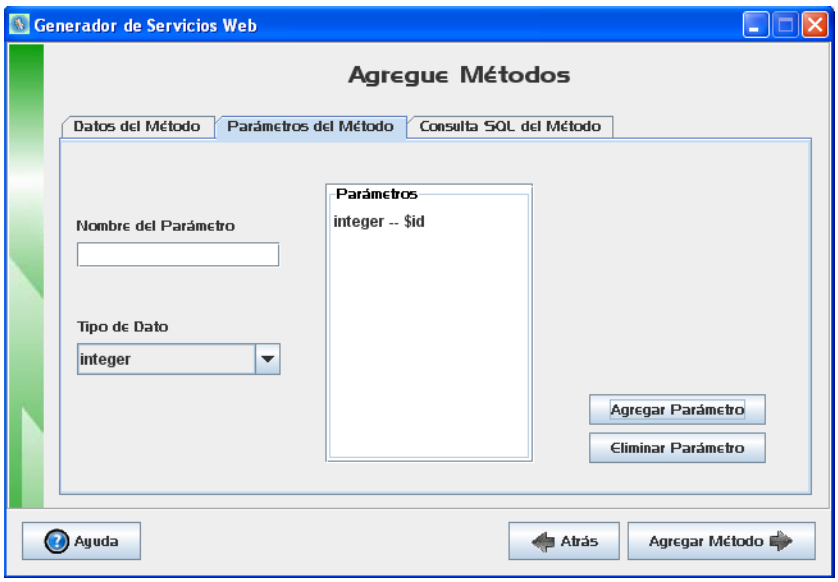


Figura 19 Vista Agregar Métodos - Parámetros del Método resultado.

Mostrado de esta manera el flujo de eventos agregar parámetro, se puede continuar con el comenzado anteriormente, del cual este es extendido. Retomando, se puede pasar al paso de agregar la consulta del método, del cual la figura 20 muestra su resultado.

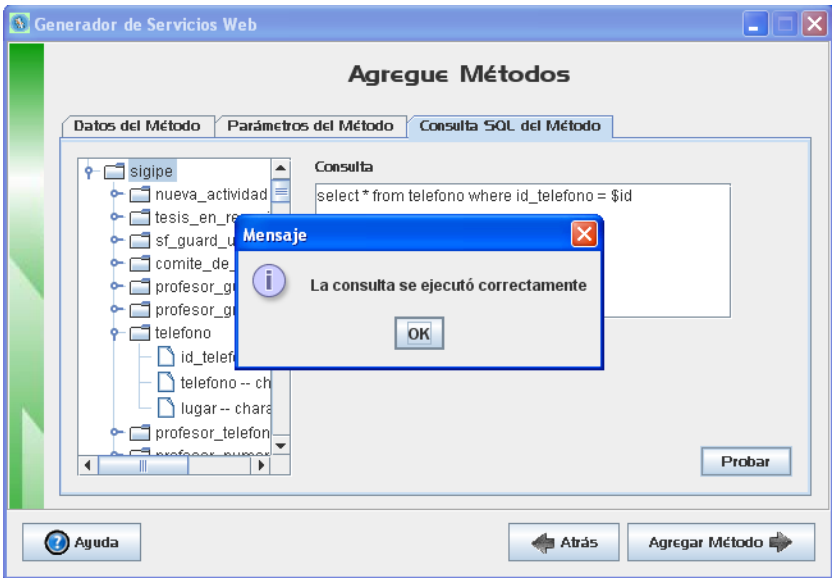


Figura 20 Vista Agregar Métodos - Consulta del Método.

Teniendo ya los datos necesarios para agregar el método, se procede a la opción de agregar el mismo, mostrado así la figura 21 el resultado deseado.

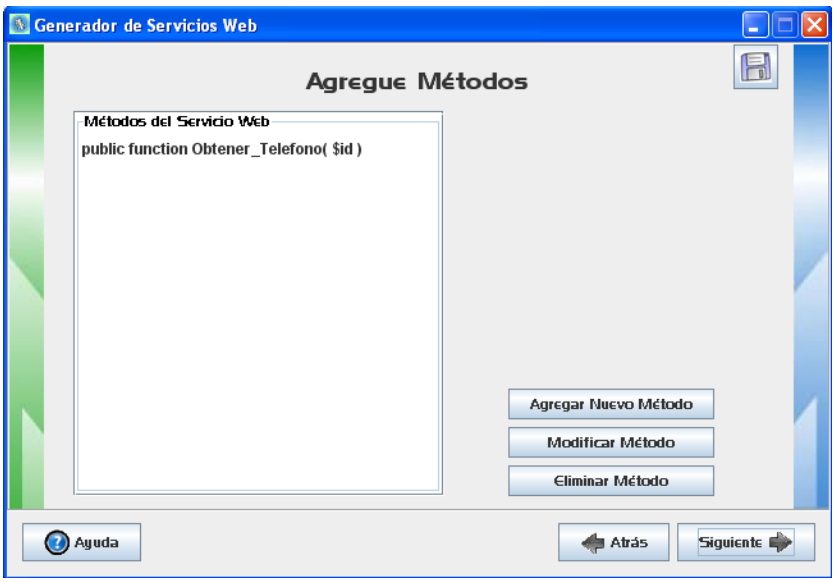


Figura 21 Vista Agregar Métodos Resultado.

4.4.4. PRUEBA AL CASO DE USO: GENERAR SERVICIO WEB.

Finalmente, después de haber cumplido con todos los pasos anteriores se puede generar el Servicio Web, para esto se deberá poner el nombre que este llevará. Ver figura 22.

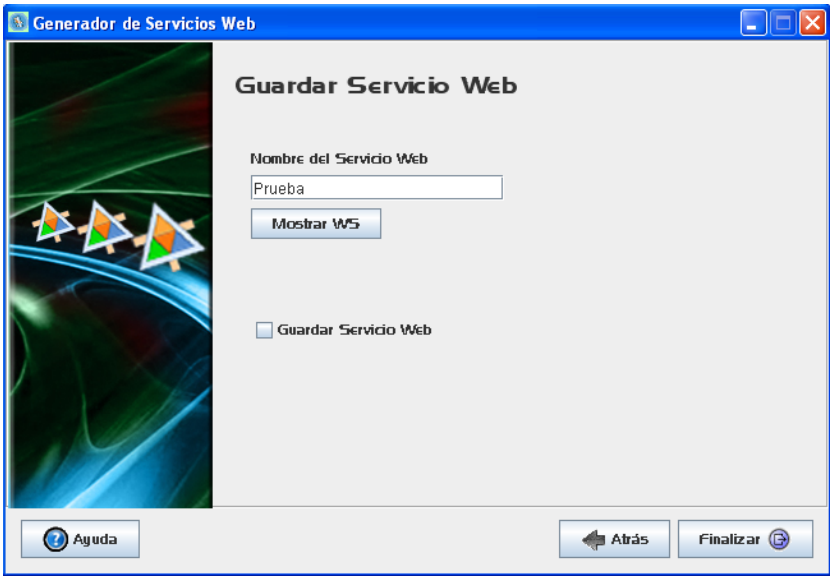


Figura 22 Vista Guardar Servicio Web – Nombre.

Se puede además visualizar el resultado antes de generar. Ver figura 23.

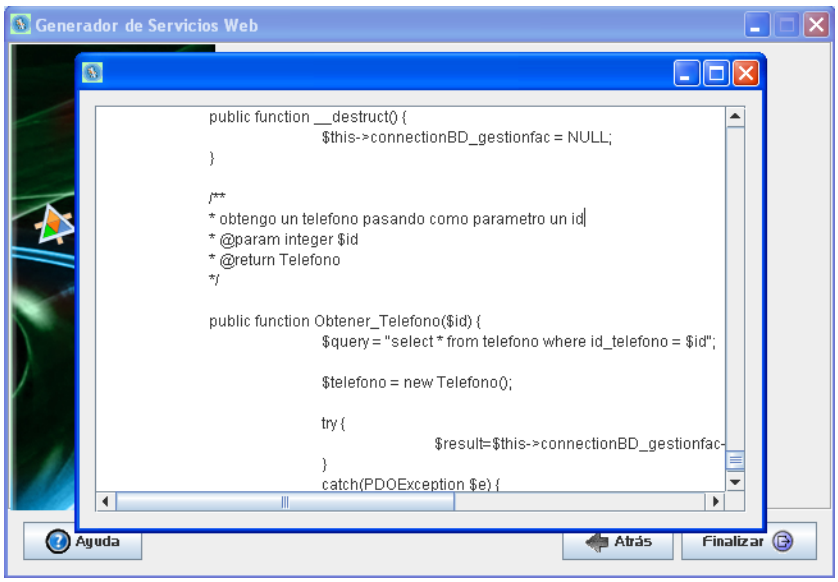


Figura 23 Muestra del Servicio Web resultante.

Ya conforme con el resultado, se marca la opción de guardar el Servicio Web y se activa la opción de seleccionar la dirección a guardar, después de esto se finaliza y se cumple así el propósito de la aplicación. Ver Figura 24.

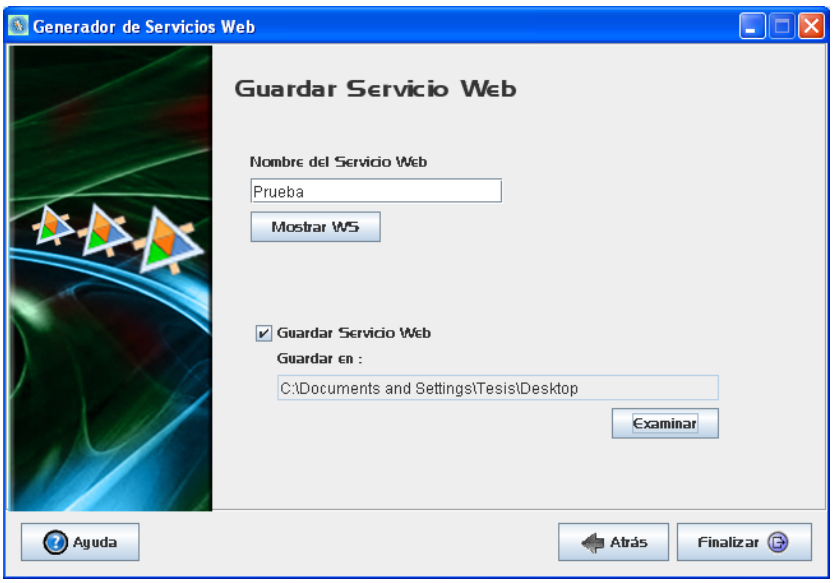


Figura 24 Vista Guardar Servicio Web - Finalizar.

4.5. CONCLUSIONES.

En este capítulo se define el diagrama de componentes para mostrar como quedó implementada la aplicación en término de componentes. Se hace una descripción del código fuente de los principales métodos implementados. También se muestran las principales pantallas de la aplicación a través de una serie de pruebas exploratorias que validan la descripción de los casos de uso más significativos.

CONCLUSIONES GENERALES

La realización del presente trabajo ha permitido el desarrollo de una aplicación capaz de generar servicios web. El software se crea principalmente, con el propósito de ser usada por aplicaciones desarrolladas y desplegadas en la UCI al querer implantarlas en otros centros externos a ella. Esta pudiera usarse además por cualquier aplicación que se quiera desplegar que necesite interactuar con diferentes orígenes de datos. Un uso correcto de esta aplicación puede aportan grandes beneficios en campo de intercambio de información entre aplicaciones y orígenes de datos existentes.

El proceso de desarrollo del software se llevó a cabo haciendo uso de la metodología OpenUP, atreves de la cual se determinaron los principales requisitos funcionales, lo que permitió la realización del diseño del sistema. A su vez teniendo en cuenta la arquitectura en capas y con la guía de patrones de diseño se llevo a cabo la implementación de una aplicación de escritorio haciendo uso del lenguaje de programación java.

Por lo antes planteado se puede concluir que tanto el objetivo general como los objetivos específicos fueron cumplidos de manera satisfactoria. Esto fue validado a atreves de pruebas exploratorias que muestran la veracidad de la afirmación anterior.

RECOMENDACIONES

- Lograr que la aplicación genere código para otros lenguajes como Java y CSharp permitiendo ampliar su uso por diversos sistemas.

- Agregarle a la aplicación la funcionalidad de hacer gráfico el diseñador de consultas SQL.

- Conseguir que la aplicación genere automáticamente el WSDL.

- Implementar una versión de tipo web para que se logre una comunicación universal con la aplicación desarrollada.

REFERENCIAS BIBLIOGRAFICAS

[1] Morales Machuca, Carlos Andrés. [En línea] [Citado el: 18 de febrero de 2009.] http://camoralesma.googlepages.com/articulo2.pdf.

[2] Autonecrología. [En línea] [Citado el: 12 de junio de 2009.] http://www.autonecrologia.net/?p=123.

[3] Gabilos. Aragon Digital Business Ecosystem . [En línea] [Citado el: 12 de junio de 2009.] http://dbe.ita.es/viewcvs/*checkout*/dbe/dev/implementers4/gabilos/docs/AXIS.pdf?rev=HEAD.

[4] Interoffice. [En línea] [Citado el: 28 de febrero de 2009.] http://www.interoffice.com.mx/index.php?option=com_content&view=article&id=3:webservices&catid=3:newsflash&Itemid=56.

[5] DESARROLLOWEB. [En línea] [Citado el: 28 de febrero de 2009.] http://www.desarrolloweb.com/articulos/1883.php.

[6] Phppro. [En línea] [Citado el: 11 de junio de 2009.] http://www.phpro.org/tutorials/Introduction-to-PHP-PDO.html.

[7] haztuprograma. [En línea] [Citado el: 10 de junio de 2009.] http://www.haztuprograma.com/tiposAplicaciones.html.

[8] Salis Álvares, Camilo Javier y Figuero Díaz, Roberth Gustavo. MyGnet. [En línea] [Citado el: 23 de febrero de 2009.] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515.

[9] agilemanifiesto. [En línea] [Citado el: 12 de junio de 2009.] http://www.agilemanifesto.org/principles.html.

[10] clubdevelopers. [En línea] [Citado el: 23 de febrero de 2009.] www.clubdevelopers.com/prog/articulos/xp/downloads/xp.pdf .

[11] willydev. [En línea] [Citado el: 23 de febrero de 2009.] http://www.willydev.net/descargas/prev/TodoAgil.pdf.

[12] cbasqa.wordpress. [En línea] [Citado el: 28 de febrero de 2009.] Citado de: http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.

[13] epf.eclipse. [En línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/inception_phase,_0hmKgBOMEduCNqgZdt_OaA.html.

[14] epf.eclipse. [En línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/elaboration_phase,_2plxwBOMEduCNqgZdt_OaA.html.

[15] epf.eclipse. [En línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/construction_phase,_48EKsBOMEduCNqgZdt_OaA.html.
[16] epf.eclipse. [En línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/transition_phase,__ca5UBOMEduCNqgZdt_OaA.html.
[17] epf.eclipse. [En línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/rolesets/openup_basic_roles,_TZIJ0O8NEdmKSqa_gSYthg.html.
[18] Murillo Alfaro, Félix. inei. [En línea] [Citado el: 28 de febrero de 2009.] http://www1.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5106/Libro.pdf.
[19] versionero. [En línea] [Citado el: 3 de marzo de 2009.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
[20] paraisolinux. [En línea] [Citado el: 3 de marzo de 2009.] http://paraisolinux.com.ar/herramientas-para-modelado-uml/.
[21] esi. [En línea] [Citado el: 10 de junio de 2009.] http://www.esi2.us.es/~vivas/fi1ii/Introduccion/Introduccion.pdf.
[22] Orbita. [En línea] [Citado el: 10 de junio de 2009.] http://orbita.starmedia.com/~prog201eq1/AGO-20-1.HTM.
[23] juanfec.lcc. [En línea] [Citado el: 10 de junio de 2009.] http://juanfc.lcc.uma.es/EDU/PM/2.POO-Presentacion.pdf.
[24] Sánchez, Jorge. [En línea] [Citado el: 10 de junio de 2009.] http://www.jorgesanchez.net/programacion/transparencias/java1.pps.
[25] clikear. [En línea] [Citado el: 3 de marzo de 2009.] http://www.clikear.com/manuales/csharp/c10.aspx.
[26] webtaller. [En línea] [Citado el: 3 de marzo de 2009.] http://www.webtaller.com/manual-java/caracteristicas-java.php .
[27] kybele. [En línea] [Citado el: 19 de mayo de 2009.] http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf.
[28] ufg.edu. [En línea] [Citado el: 4 de marzo de 2009.] http://wwwisis.ufg.edu.sv/wwwisis/documentos/TE/636.0812-H557s/636.0812-H557s-Bga.pdf.
[29] pilp.edu. [En línea] [Citado el: 25 de mayo de 2009.] http://www.pilp.edu.ar/pilpweb/contenido/documentopilp/Documento39.pdf.

[30] mitecnologico. [En línea] [Citado el: 19 de enero de 2009.] http://www.mitecnologico.com/Main/RequerimientosFuncionalesYNoFuncionales.
[31] teleformacion.uci.cu. [En línea] [Citado el: 14 de mayo de 2009.] http://eva.uci.cu/file.php/259/CURSO_2008-2009/Materiales_Basicos/Semana_3/Conf/Conferencia_2_de_Arquitectura_2009ok.doc.
[32] oasis. [En línea] [Citado el: 10 de junio de 2009.] http://oasis.cisc-ug.org/letzhune/cisc/tutoriales/tercero/Programacion%20por%20capas.doc.
[33] inf.utfsm.cl. [En línea] [Citado el: 14 de mayo de 2009.] http://www.inf.utfsm.cl/%7Evisconti/ili236/Documentos/08-Patrones.pdf.
[34] ldc.usb.ve. [En línea] [Citado el: 14 de mayo de 2009.] http://www.ldc.usb.ve/~teruel/ci3711/patron3a/index.html.
[35] eisc.univalle.edu.co. [En línea] [Citado el: 14 de mayo de 2009.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/exposiciones2005B/polimorfismoFabricacionPura_G01/presentacion.ppt.
[36] tvdi.det.uvigo.es. [En línea] [Citado el: 14 de mayo de 2009.] http://tvdi.det.uvigo.es/~avilas/UML/node49.html.

BIBLIOGRAFÍA

- agilemanifiesto. [En Línea] [Citado el: 12 de junio de 2009.] http://www.agilemanifesto.org/principles.html.
- Autonecrología. [En Línea] [Citado el: 12 de junio de 2009.] http://www.autonecrologia.net/?p=123.
- cbasqa.wordpress. [En Línea] [Citado el: 28 de febrero de 2009.] Citado de: http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.
- clikear. [En Línea] [Citado el: 3 de marzo de 2009.] http://www.clikear.com/manuales/csharp/c10.aspx.
- clubdevelopers. [En Línea] [Citado el: 23 de febrero de 2009.] www.clubdevelopers.com/prog/articulos/xp/downloads/xp.pdf .
- DESARROLLOWEB. [En Línea] [Citado el: 28 de febrero de 2009.] http://www.desarrolloweb.com/articulos/1883.php.
- eisc.univalle.edu.co. [En Línea] [Citado el: 14 de mayo de 2009.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/exposiciones2005B/polimorfismoFabricacionPura_G01/presentacion.ppt.
- epf.eclipse. [En Línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/inception_phase,_0hmKgBOMEduCNqgZdt_OaA.html.
- epf.eclipse. [En Línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/elaboration_phase,_2plxwBOMEduCNqgZdt_OaA.html.
- epf.eclipse. [En Línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/construction_phase,_48EKsBOMEduCNqgZdt_OaA.html.
- epf.eclipse. [En Línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/transition_phase,__ca5UBOMEduCNqgZdt_OaA.html.
- epf.eclipse. [En Línea] [Citado el: 5 de marzo de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/rolesets/openup_basic_roles,_TZIJ0O8NEdmKSqa_gSYthg.html.
- esi. [En Línea] [Citado el: 10 de junio de 2009.] http://www.esi2.us.es/~vivas/fi1ii/Introduccion/Introduccion.pdf.

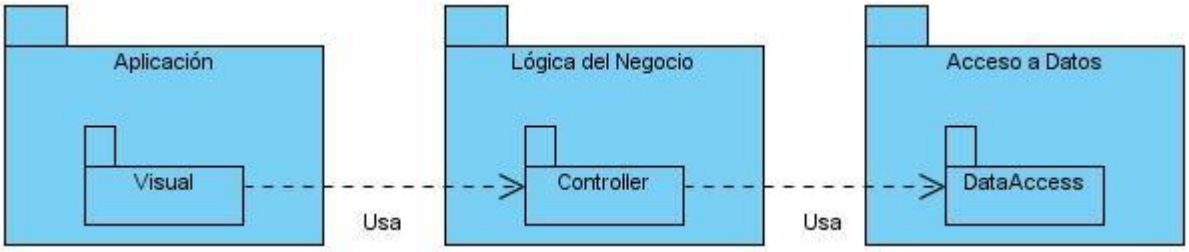
- **Gabilos.** Aragon Digital Business Ecosystem . [En Línea] [Citado el: 12 de junio de 2009.] http://dbe.ita.es/viewcvs/*checkout*/dbe/dev/implementers4/gabilos/docs/AXIS.pdf?rev=HEAD.
- haztuprograma. [En Línea] [Citado el: 10 de junio de 2009.] http://www.haztuprograma.com/tiposAplicaciones.html.
- inf.utfsm.cl. [En Línea] [Citado el: 14 de mayo de 2009.] http://www.inf.utfsm.cl/%7Evisconti/ili236/Documentos/08-Patrones.pdf.
- Interoffice. [En Línea] [Citado el: 28 de febrero de 2009.] http://www.interoffice.com.mx/index.php?option=com_content&view=article&id=3:webservices&catid=3:newsflash&Itemid=56.
- juanfec.lcc. [En Línea] [Citado el: 10 de junio de 2009.] http://juanfc.lcc.uma.es/EDU/PM/2.POO-Presentacion.pdf.
- kybele. [En Línea] [Citado el: 19 de mayo de 2009.] http://kybele.escet.urjc.es/documentos/HC/Exposiciones/EclipseIDE.pdf.
- ldc.usb.ve. [En Línea] [Citado el: 14 de mayo de 2009.] http://www.ldc.usb.ve/~teruel/ci3711/patron3a/index.html.
- mitecnologico. [En Línea] [Citado el: 19 de enero de 2009.] http://www.mitecnologico.com/Main/RequerimientosFuncionalesYNoFuncionales.
- **Morales Machuca, Carlos Andrés.** [En Línea] [Citado el: 18 de febrero de 2009.] http://camoralesma.googlepages.com/articulo2.pdf.
- **Murillo Alfaro, Félix.** inei. [En Línea] [Citado el: 28 de febrero de 2009.] http://www1.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5106/Libro.pdf.
- oasis. [En Línea] [Citado el: 10 de junio de 2009.] http://oasis.cisc-ug.org/letzhune/cisc/tutoriales/tercero/Programacion%20por%20capas.doc.
- Orbita. [En Línea] [Citado el: 10 de junio de 2009.] http://orbita.starmedia.com/~prog201eq1/AGO-20-1.HTM.
- paraisolinux. [En Línea] [Citado el: 3 de marzo de 2009.] http://paraisolinux.com.ar/herramientas-para-modelado-uml/.
- Phppro. [En Línea] [Citado el: 11 de junio de 2009.] http://www.phpro.org/tutorials/Introduction-to-PHP-PDO.html.
- pilp.edu. [En Línea] [Citado el: 25 de mayo de 2009.] http://www.pilp.edu.ar/pilpweb/contenido/documentopilp/Documento39.pdf.
- **Salis Álvares, Camilo Javier and Figuero Díaz, Roberth Gustavo.** MyGnet. [En Línea] [Citado el: 23 de febrero 2009.]

http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515.

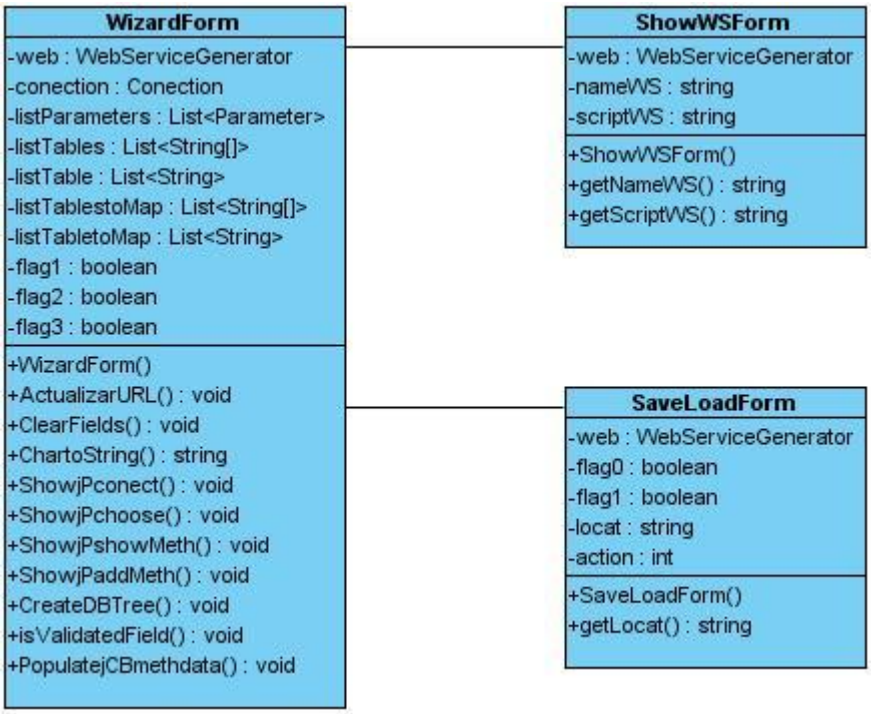
- **Sánchez, Jorge.** [En Línea] [Citado el: 10 de junio de 2009.] http://www.jorgesanchez.net/programacion/transparencias/java1.pps.
- teleformacion.uci.cu. [En Línea] [Citado el: 14 de mayo de 2009.] http://eva.uci.cu/file.php/259/CURSO_2008-2009/Materiales_Basicos/Semana_3/Conf/Conferencia_2_de_Arquitectura_2009ok.doc.
- tvdi.det.uvigo.es. [En Línea] [Citado el: 14 de mayo de 2009.] http://tvdi.det.uvigo.es/~avilas/UML/node49.html.
- ufg.edu. [En Línea] [Citado el: 4 de marzo de 2009.] http://wwwisis.ufg.edu.sv/wwwisis/documentos/TE/636.0812-H557s/636.0812-H557s-Bga.pdf.
- versionero. [En Línea] [Citado el: 3 de marzo de 2009.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
- webtaller. [En Línea] [Citado el: 3 de marzo de 2009.] http://www.webtaller.com/manual-java/caracteristicas-java.php .
- willydev. [En Línea] [Citado el: 23 de febrero de 2009.] http://www.willydev.net/descargas/prev/TodoAgil.pdf.

ANEXOS

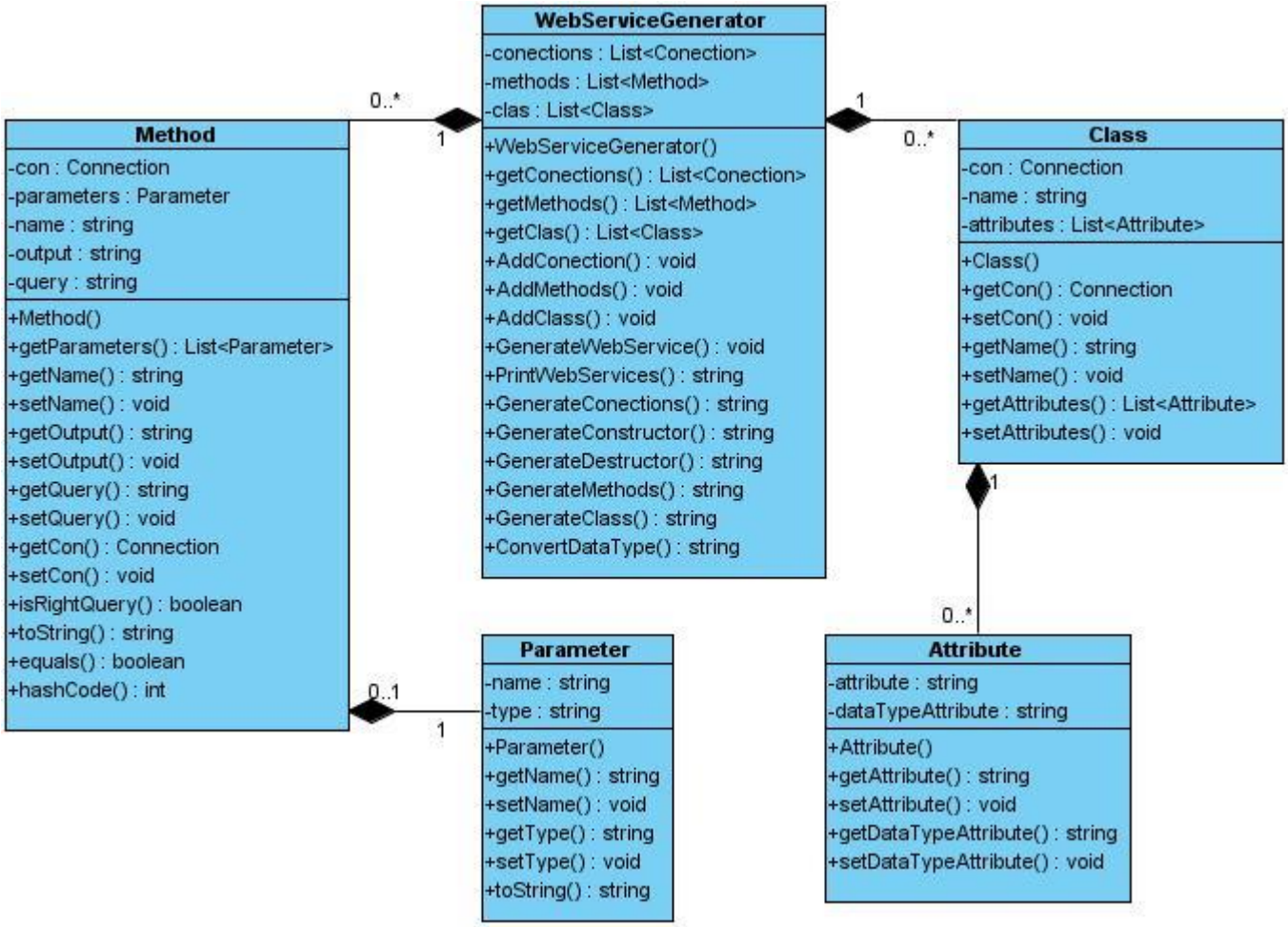
Anexo 1.



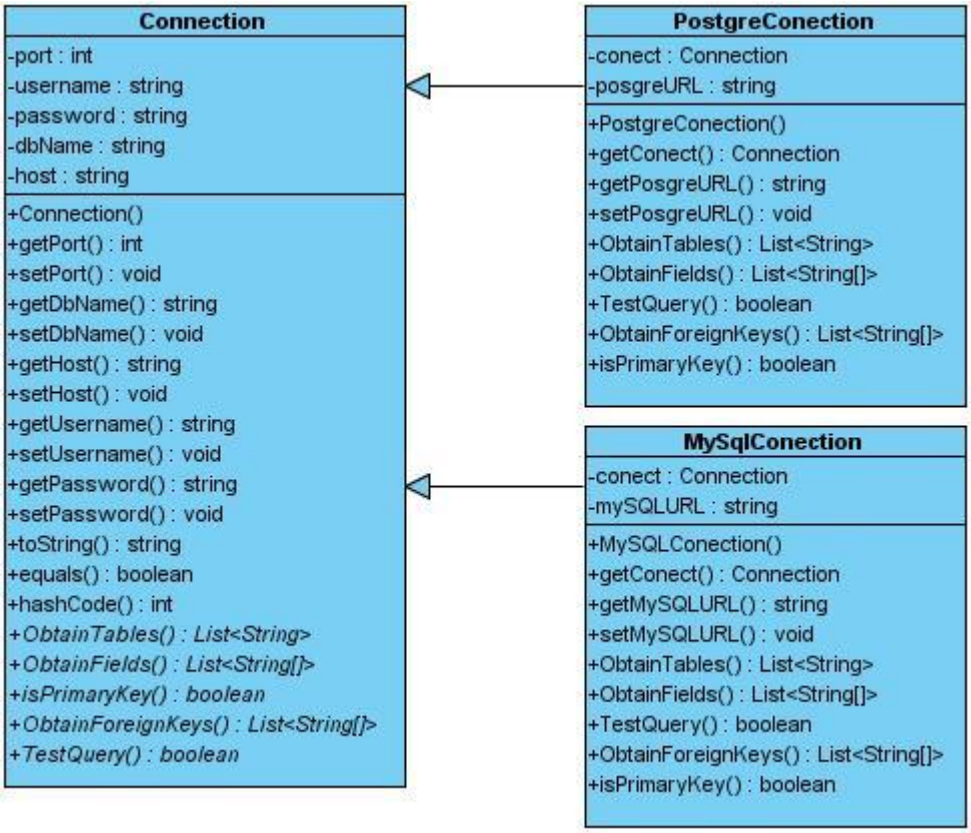
Anexo 2



Anexo 3


WebServiceGenerator
-conections : List<Conection>
-methods : List<Method>
-clas : List<Class>
+WebServiceGenerator()
+getConections() : List<Conection>
+getMethods() : List<Method>
+getClas() : List<Class>
+AddConection() : void
+AddMethods() : void
+AddClass() : void
+GenerateWebService() : void
+PrintWebServices() : string
+GenerateConections() : string
+GenerateConstructor() : string
+GenerateDestructor() : string
+GenerateMethods() : string
+GenerateClass() : string
+ConvertDataType() : string
Method
-con : Connection
-parameters : Parameter
-name : string
-output : string
-query : string
+Method()
+getParameters() : List<Parameter>
+getName() : string
+setName() : void
+getOutput() : string
+setOutput() : void
+getQuery() : string
+setQuery() : void
+getCon() : Connection
+setCon() : void
+isRightQuery() : boolean
+toString() : string
+equals() : boolean
+hashCode() : int
Class
-con : Connection
-name : string
-attributes : List<Attribute>
+Class()
+getCon() : Connection
+setCon() : void
+getName() : string
+setName() : void
+getAttributes() : List<Attribute>
+setAttributes() : void
Parameter
-name : string
-type : string
+Parameter()
+getName() : string
+setName() : void
+getType() : string
+setType() : void
+toString() : string
Attribute
-attribute : string
-dataTypeAttribute : string
+Attribute()
+getAttribute() : string
+setAttribute() : void
+getDataTypeAttribute() : string
+setDataTypeAttribute() : void
0..*
1
1
0..*
1
0..*
0..1
1

Anexo 4


Connection
-port : int
-username : string
-password : string
-dbName : string
-host : string
+Connection()
+getPort() : int
+setPort() : void
+getDbName() : string
+setDbName() : void
+getHost() : string
+setHost() : void
+getUsername() : string
+setUsername() : void
+getPassword() : string
+setPassword() : void
+toString() : string
+equals() : boolean
+hashCode() : int
+ObtainTables() : List<String>
+ObtainFields() : List<String[]>
+isPrimaryKey() : boolean
+ObtainForeignKeys() : List<String[]>
+TestQuery() : boolean
PostgreConection
-conect : Connection
-posgreURL : string
+PostgreConection()
+getConect() : Connection
+getPosgreURL() : string
+setPosgreURL() : void
+ObtainTables() : List<String>
+ObtainFields() : List<String[]>
+TestQuery() : boolean
+ObtainForeignKeys() : List<String[]>
+isPrimaryKey() : boolean
MySqlConection
-conect : Connection
-mySQLURL : string
+MySQLConection()
+getConect() : Connection
+getMySQLURL() : string
+setMySQLURL() : void
+ObtainTables() : List<String>
+ObtainFields() : List<String[]>
+TestQuery() : boolean
+ObtainForeignKeys() : List<String[]>
+isPrimaryKey() : boolean

GLOSARIO

Stakeholder: Representa un grupo de personas cuyas necesidades deben ser satisfechas por el proyecto. Esto es un rol que podría ser desempeñado por cualquiera que esté (o potencialmente estará) materialmente afectado por el resultado del proyecto.

Open Source: Calificación de software que cumple una serie de requisitos, principalmente aquel que permite una libre redistribución, distribuye el código fuente, y permite modificaciones y trabajos derivados.

Protocolo: Se conoce como protocolo de comunicaciones a un conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre sistemas.

Estándar: Modelo que se toma de referencia para realizar un proceso o alcanzar un resultado.

Metodología: La rama de la metodología, dentro de la ingeniería de software, se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

Framework: en el desarrollo de software, es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Software: Conjunto de instrucciones y datos codificados para ser leídas e interpretadas por una computadora. Estas instrucciones y datos fueron concebidos para el procesamiento electrónico de datos.

Interfaz: Es un método para facilitar la interacción del usuario con el ordenador o la computadora a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas) además de texto.

Código Fuente: El código fuente de un programa informático (o software) es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa.

Multithreaded (multihilos): Característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

Wizards: Programa que mediante pantallas interactivas facilitan y sirven de guía para el uso de otros programas o aplicaciones.

RPC (Remote Procedure Call ó 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.

WSDL: son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web.