UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

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

TÍTULO: Proveedor de datos de los servicios Web de la Universidad de las Ciencias Informáticas para el sistema GeoQ-Guardián.

AUTOR: Gerardo Valdés Piña

TUTOR: MSc. Manuel Enrique Puebla Martínez

CO-TUTOR: Ing. Miguel Milán Isaac

La Habana, junio del 2014.

"Año 56 de la Revolución"

“Aquel que muere por la patria no se le dice muerto”

Hugo Rafael Chávez Frías

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____	_____
Gerardo Valdés Piña	MSc. Manuel Enrique Puebla Martínez	Ing. Miguel Milán Isaac
Firma del Autor	Firma del Tutor	Firma del Tutor

DATOS DE CONTACTO

Síntesis del Autor: Gerardo Valdés Piña
Correo Electrónico: gpina@estudiantes.uci.cu

Síntesis del Tutor: MSc. Manuel Enrique Pueblas Martínez
Correo Electrónico: mpuebla@uci.cu
Profesión: Ingeniero en Ciencias Informáticas

Síntesis del Tutor: Ing. Miguel Milán Isaac
Correo Electrónico: mmilan@uci.cu
Profesión: Ingeniero en Ciencias Informáticas

AGRADECIMIENTOS

Agradezco a toda mi familia, principalmente a mi mamá, mi papá y mi novia por ayudarme a formarme como Ingeniero en Ciencias Informáticas y mantenerme con ánimo.

A Grisel y todos mis tíos por apoyarme y ayudarme cuando mi mamá no estaba en Cuba.

A Sergio Y Mileysis por ser las primeras personas en ayudarme es esta escuela.

A Juana y Javier por ser tan atentos conmigo.

A Felipe por depositar tanta confianza en mí

A mis sobrinos.

A todos mis vecinos, en especial a Nereida, Toto y Olfito, que en paz descansen.

A todos mis compañeros de la escuela militar Antonio Maceo.

A mis tutores David Silva, Pueblas y Miguel, por haber tenido tanta paciencia conmigo.

A Sandor por la ayuda brindada en la tesis y en el proyecto.

Al profesor Daniel Echeverría por haber depositado confianza en mí.

A todos mis compañeros de aula, en especial a Noraima, Alber, Lorenzo, Yanet, Yadira…, en fin a todos.

A mi tribunal.

A todo el claustro de profesores que ayudó en mi formación como Ingeniero.

A toda aquella persona que me ayudó en la tesis.

DEDICATORIA

Dedico el resultado de la tesis a mi familia, principalmente a mi mamá, mi papá, mi hermana y mi novia.

A todas las personas que hicieron posible que yo formara parte de esta Universidad.

A todas las personas que me ayudaron cuando tuve algún problema.

A todos aquellos que depositaron confianza en mí y me hicieron sentir seguro.

RESUMEN

En el proyecto SIG-DESKTOP se desarrolla el sistema GeoQ-Guardián que permite la planificación, control y gestión de la guardia obrera-estudiantil en la Universidad de las Ciencias Informáticas (UCI). La planificación de la guardia requiere de versiones de datos actualizadas, para evitar errores en la información que se genera asociado a la planificación. Este proceso de actualización se hace de forma manual por parte de los administradores de GeoQ-Guardián, y la solución que se propone, es la implementación de un proveedor de datos que consulte los datos de los servicios Web existentes en la UCI; teniendo en cuenta que muchos de estos servicios, brindan al resto de la comunidad universitaria información compartida y segura. Para ello se realizó un estudio y caracterización del funcionamiento de los servicios Web existentes en la Universidad, y un análisis de las tecnologías y tendencias actuales en el proceso de desarrollo de software. Para el cumplimiento de las actividades de la investigación, se siguieron las fases establecidas en el marco de trabajo de la metodología “Proceso Unificado Ágil”, y en la implementación del sistema se empleó el patrón arquitectónico en capas unido a los patrones GRASP, permitiendo la separación del código por niveles. Finalmente la validación del software con pruebas de caja negra, permitió identificar un grupo de no conformidades que fueron eliminadas.

Palabras clave: patrón, proveedores de datos, servicios Web.

ABSTRACT

The project SIG-DESKTOP develops the system GeoQ-Gardian that allows the planning, control and management of the dutty guard for students and workers at the University of Information Sciences (UIS). The guard planning requires versions of updated data to avoid errors in the information generated associated with the planning. This updating process is done manually by administrators of the system GeoQ-Gardian, therefore the proposed solution is the implementation of a Data Provider to consult the data from existing Web services in the UIS; considering that many of these services give to the university community secure information sharing. For this investigation, it was made a study and characterization of the performance of existing Web services at the University, and it was analyzed current technologies and trends in software development process. In order to fulfill the research activities, it was followed the steps set out in the framework of the methodology "Process Agile Unified" and in the develops of system was used architectural pattern in layers attached to pattern GRASP, allowing tiered separation to code. Finally validation software with black box testing, identified a group of non- conformities that were eliminated.

Keywords: *data providers, pattern, Web services.*

Índice

Índice de tablas

Índice de figuras

Introducción

"El tipo de sociedad que el nuevo orden mundial ofrece, (…) las tendencias comerciales a través de medios electrónicos, las nuevas teorías organizacionales y el modus operandi del ser humano en el siglo XXI requieren la automatización de los procesos cotidianos y la despersonalización en muchos de ellos" (Machuca, 2008). Argumentos como estos han servido de pilares en el surgimiento de nuevas áreas en la tecnología, entre ellas, las Tecnologías de la Información y las Comunicaciones (TIC).

Vinculado a las TIC, la informática, es una ciencia que estudia métodos, procesos y técnicas que hacen posible el tratamiento automatizado de la información por medio de ordenadores (RAE, 2001). Esta disciplina se aplica a numerosas aristas del conocimiento o la actividad humana, imponiendo nuevos retos a alcanzar para no quedar rezagados en el camino dominante de la informatización.

En la actualidad, el potencial tecnológico se centraliza fundamentalmente en las grandes industrias monopolizadas ubicadas en territorio de los países desarrollados. Solo una pequeña parte de este parque tecnológico reside en países con vías al desarrollo. *"Datos de 2006 de la Unión Internacional de Telecomunicaciones muestran que mientras el 58.6% de los habitantes de los países desarrollados tienen acceso a Internet, en los países en vías de desarrollo apenas el 10.2% de los habitantes tiene acceso a esta tecnología"* (Lara Zucchi, 2013).

En Internet navegan dos tipos de entes: las personas que visitan páginas Web o acceden a servicios interactivos y las aplicaciones distribuidas. En la Web residen programas que intercambian datos de forma automática sin la participación humana.

Un programador puede implementar aplicaciones basadas en rutinas y datos proporcionados desde distintos orígenes empleando servicios Web. Ciertas rutinas pueden ser usadas, por ejemplo: visualizar información en una página Web, o simplemente insertar datos en un programa de predicción y ayuda a la toma de decisiones. Si el acceso a dichas rutinas y a los datos que generan se hace usando ciertos protocolos estandarizados, entonces se puede hablar de servicios Web (Universidad Tecnica del Norte, 2004).

Se viven tiempos caracterizados por la alta tasa de consumo tecnológico en los procesos productivos y en la sociedad en general, lo que explica la dependencia tecnológica que persiste en el planeta. La nueva era digital asienta la necesidad de asimilar las nuevas tecnologías de la información y realizar serios esfuerzos para superar la actual brecha digital en un mundo donde impera la desigualdad.

Cuba es uno de los países que aboga por la igualdad y el desarrollo. El Estado ha dedicado tiempo y esfuerzo para informatizar los principales sectores de la economía nacional. Del mismo modo se pretende lograr una masificación en el uso de las TIC producto a los beneficios que proporcionan, entre los que se destacan: ahorro de la actividad física y mental del hombre; disminución del tiempo de obtención de información; perfeccionamiento de las comunicaciones; eliminación de las barreras de tiempo y espacio; y asistencia de la colaboración entre empresas. La máxima dirección del país ha promovido el desarrollo de soluciones informáticas encaminadas a solucionar problemas para entidades cubanas o extranjeras.

Con el objetivo de satisfacer las exigencias de la industria cubana de software, es necesario responder a las necesidades y requisitos de los clientes. Pues cada día se incrementa el número de empresas dedicadas a la creación de software con un nivel de eficiencia elevado, resultando cada vez más difícil la competencia en el mercado entre productos informáticos.

La Universidad de las Ciencias Informáticas, tiene la particularidad de ser un centro de estudios y a la vez es una empresa de producción de software. La institución cuenta con varios centros de desarrollo. Uno de ellos es el Centro Geoinformática y Señales Digitales (GEySED), de la Facultad 6, donde se encuentra el proyecto productivo SIG-DESKTOP. En SIG-DESKTOP, se implementa el sistema GeoQ-Guardián, que permite la planificación, control y gestión de los activos de la guardia obrera-estudiantil, para ello requiere de datos reales de los trabajadores y estudiantes de la Universidad. Es importante que estos datos estén actualizados, pues la planificación de la guardia debe ser lo más real posible. El sistema se encuentra en explotación en varias áreas de la Universidad y es operado por los diferentes planificadores de cada área. Se han identificado por parte de estos planificadores un grupo de dificultades que a continuación se detallan:

- ✓ Existen personas que han causado baja de la Universidad y aparecen en la planificación realizada.
- ✓ Se realizan cambios internos entre el personal de las áreas y no es actualizada la información en tiempo.
- ✓ Se planifican personas a la misma vez en áreas distintas.
- ✓ Los reportes que se entregan no contienen información confiable.

A partir de estas dificultades se realizó un análisis del proceso de consumo de datos en GeoQ-Guardián, lo que arrojó que la actualización de los datos se realiza manualmente por parte de los administradores del sistema, desde los servicios Web disponibles en la UCI hasta la Base de Datos (BD) central de GeoQ-Guardián. La ejecución de este proceso de actualización provoca que en ocasiones se duplique o se omita

información en las tablas de la BD, y en determinados momentos cuando se realiza en el sistema la gestión los datos de la planificación de la guardia, la información que se obtiene no es confiable.

Por tanto, en la investigación se identifica el siguiente problema a resolver: la actualización manual de los datos desde los servicios Web, provoca la aparición de errores en el proceso de planificación y control de la guardia obrera-estudiantil en la UCI. Se define como **objeto de estudio** el consumo de datos de los servicios Web, enmarcado en el **campo de acción** al consumo de datos de servicios Web mediante proveedores de datos.

Para solucionar el problema planteado se define como **objetivo general:** Desarrollar un proveedor de datos, para el proceso de planificación de la guardia obrera-estudiantil en el sistema GeoQ-Guardián, que sustituya las actualizaciones manuales desde los servicios Web hasta la base de datos central.

Con el fin de lograr el objetivo general propuesto, se han formulado las siguientes **preguntas de investigación**:

¿Qué estrategia emplea GeoQ-Guardián para consultar los datos desde los servicios Web?

¿Bajo qué circunstancias se pueden generar errores en la planificación de la guardia?

¿Los servicios Web en la UCI, ofrecerán toda la información actualizada que necesita el sistema para la guardia?

¿Qué estructura presentan los servicios Web de la UCI qué permitan a GeoQ-Guardián consultar la información que se publica en ellos?

Para dar respuesta a las preguntas anteriores se trazaron las siguientes **tareas de investigación**:

1. Caracterización del consumo de datos desde los servicios Web.
2. Descripción de los servicios Web existentes en la UCI.
3. Comparación de soluciones, y mecanismos análogos, existentes a un proveedor de datos.
4. Descripción de las tendencias y tecnologías actuales aplicables al proceso de negocio.
5. Diseño del proveedor de datos de los servicios Web UCI para el sistema GeoQ-Guardián.
6. Implementación del proveedor de datos de los servicios Web UCI para el sistema GeoQ-Guardián.
7. Validación de la solución propuesta.

Una vez concluida la investigación se espera obtener los siguientes resultados:

- ✓ Informe de la investigación científica.
- ✓ Documentación técnica del proceso de desarrollo.
- ✓ Proveedor de servicios Web para GeoQ-Guardián completamente funcional.

El desarrollo de la investigación estuvo regido por un conjunto de métodos científicos que se exponen a continuación:

- **Histórico - lógico:** se utilizó para conocer el comportamiento histórico de los proveedores de datos y servicios Web, así como las herramientas y tecnologías usadas para el desarrollo de la solución del problema planteado.
- **Modelación:** se usó durante las diferentes etapas del proceso de desarrollo del software para construir modelos que permitieron la creación de abstracciones, con las que se representaron los procesos identificados.
- **Análisis y síntesis:** permitió el procesamiento de toda la información que se gestionó referente a los proveedores y servicios Web, escogiéndose los datos y características más relevantes para el desarrollo de la aplicación.

El contenido de trabajo se distribuyó de la siguiente manera:

Capítulo 1. Fundamentación teórica: en este capítulo, se describe el objeto de estudio y su impacto en la UCI. Se detallan las tecnologías y tendencias actuales que fueron utilizadas para el desarrollo del sistema informático. Se describe la metodología utilizada.

Capítulo 2. Diseño del sistema propuesto: en el capítulo se describe los conceptos relacionados al modelo de dominio del problema. Se especifica las funcionabilidades del sistema propuesto y se representan los diagramas considerados por el desarrollador en la disciplina del diseño.

Capítulo 3. Implementación y Pruebas: en este capítulo se detallan los artefactos asociados al flujo de implementación del sistema y se realiza el control de la calidad mediante el proceso de prueba.

Capítulo 1. Fundamentación Teórica

Introducción

El presente capítulo está dirigido al estudio del funcionamiento de los servicios Web y cómo se distribuyen en la Universidad. Se enuncian definiciones y conceptos medulares que constituyen la base en la comprensión del objeto de estudio y en la descripción del dominio del problema. Se caracterizan las herramientas y tecnologías para el desarrollo de la solución, y se describe la metodología empleada.

1.1. Conceptos asociados al dominio del problema

- **Proveedor de datos:** un proveedor de datos posibilita la conexión a un origen de datos, obtener información y ejecutar ordenes con la misma. Lo más importante para el concepto de proveedor de datos es la transparencia que ofrece al sistema que lo usa sobre los orígenes de datos desde los que se alimenta. Para el sistema cliente tiene una sola cara (interfaz o protocolo) pero para los orígenes de datos tiene muchas ramificaciones, todas especializadas para cada origen por separado. Abstrae al sistema de lidiar con orígenes de datos distintos, actúa como traductor o software de conectividad entre aplicaciones heterogéneas y se pueden agregar como plugin o extensión.

  Los proveedores de datos ofrecen una arquitectura de acceso a datos más simple y sirven de puente entre una aplicación y un origen de datos. Cuentan con una clase encargada de establecer la conexión con este origen, y un mecanismo de almacenamiento temporal de información. Este mecanismo se puede interpretar como una caché, los datos contenidos en esta caché son estructurados en XML[1].
- **Servicio Web:** una definición formal sobre servicios Web la aporta el W3C[2], *"un servicio Web es una aplicación software identificada por una URI[3], cuyas interfaces se pueden definir, describir y descubrir, mediante documentos XML. Un servicio Web soporta interacciones directas con otros*

[1] **XML**: Extensible Markup Language («lenguaje de marcas ampliable»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

[2] **W3C**: World Wide Web Consortium, es un consorcio internacional que produce estándares para la World Wide Web.

[3] **URI**: cadena corta de caracteres que identifica inequívocamente un recurso.

agentes software utilizando mensajes XML intercambiados mediante protocolos basados en Internet" (Consortium, 2008). En otras palabras, son métodos públicos disponibles en la Web para distintas aplicaciones.

1.2. Descripción general del objeto de estudio

Las facilidades de intercambio de información que los servicios Web ofrecen, han permitido definir estructuras que permiten operar con los mismos. Dentro de las soluciones propuestas es posible identificar la capacidad de integración de datos y funcionalidades, así como el intercambio y procesamiento de información entre sistemas.

En general los servicios Web se utilizan para poder proporcionar interoperabilidad y extensibilidad entre diversos servicios y aplicaciones informáticas, para que al mismo tiempo sea posible su combinación y comunicación (García, 2009).

Una vez publicado un servicio Web, otras aplicaciones pueden hacer uso de los servicios que brinda. Cuentan con interfaces que ofrecen un conjunto de funcionabilidades que un cliente puede invocar, además de hacerse públicas en un servidor mediante un lenguaje de descripción de interfaces (WSDL[4]); este lenguaje proporciona al cliente la información necesaria para interactuar con el servicio Web, es extensible y se puede utilizar para describir, prácticamente, cualquier servicio de red.

El consumo de estos servicios se facilita con el empleo de protocolos diseñados especialmente para ellos, como es el caso de SOAP[5] (Consortium, 2004). Este protocolo define un mecanismo simple para la comunicación, en un entorno distribuido o descentralizado, entre componentes de software o aplicaciones. Se basa en la utilización de HTTP[6] para el transporte de mensajes y el lenguaje XML para la estructura del cuerpo del mensaje y representar datos independientemente de las plataformas de desarrollo.

En general, el modelo de servicios Web presenta un esquema de tres capas bien diferenciadas que se especifican gráficamente en la figura 1.

[4] **WSDL**: describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo.

[5] **SOAP**: es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

[6] **HTTP**: protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web.

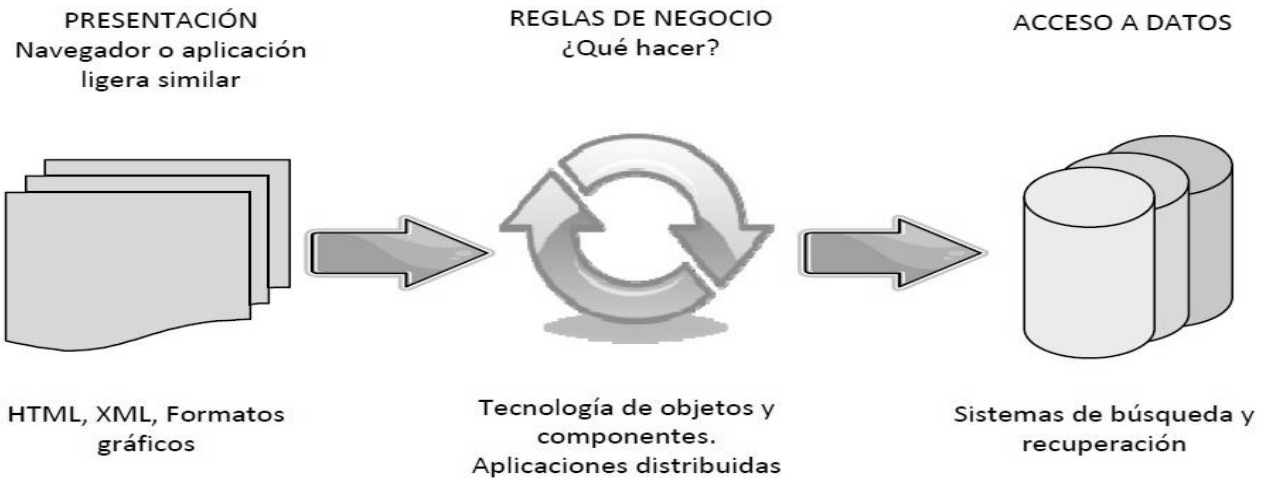


Fig 1. Modelo en tres capas para el desarrollo de servicios Web

1.2.1. Características de los servicios Web

Los servicios Web presentan además (Consortium, 2004):

- ✓ **Interoperabilidad:** pueden ser accedidos desde cualquier plataforma.
- ✓ **Tipos de datos de las interfaces:** los tipos de datos utilizados por los servicios Web son los que se utilizan por la mayoría de los lenguajes de programación.
- ✓ **Uso de los estándares de internet:** reutilizan protocolos que ya están en uso, con lo que se evita crear nuevos estándares que los soporten.
- ✓ **Soporte de cualquier lenguaje:** su desarrollo no tiene por qué estar ligado a un lenguaje determinado.

1.2.2. Plataformas y protocolos utilizados en servicios Web

Definido el servicio, se necesita un conjunto de recursos que soporten las comunicaciones dentro de los servicios Web (Consortium, 2004):

- ✓ **Infraestructura de protocolos:** los servicios Web se caracterizan por una interfaz y protocolos de negocio.
- ✓ **Transporte:** la red de comunicación se oculta detrás de un protocolo de transporte, el más común HTTP.

- ✓ **Mensajes:** una vez que se cuenta con un protocolo de transporte, se precisa un formato para la información; este rol se le asigna a SOAP.

1.3. Los servicios Web en la UCI. Situación actual de GeoQ-Guardián

Los servicios Web en la Universidad tienen como principio, la actualización diaria de los datos publicados producto a los cambios constantes en la información asociada al personal y a otras áreas. La Dirección de Informatización, encargada de la administración de estos servicios, los agrupa por negocios para que el resto de las aplicaciones de la Universidad puedan acceder a este origen de datos de acuerdo a su necesidad. Actualmente estos servicios están en fase de soporte, y la información que se está ofreciendo a algunos sistemas de la UCI no es totalmente confiable; se corre el riesgo de no contar con todos los datos actualizados, necesarios para la planificación de la guardia en el sistema GeoQ-Guardián.

La arquitectura de los servicios Web en la UCI se basa en los siguientes principios (Dirección de Informatización, 2007):

- ✓ Entorno descentralizado.
- ✓ Orientado a servicios.
- ✓ Bajo acoplamiento entre los servicios.
- ✓ Colaboración de estos servicios, para dar solución a problemas.
- ✓ Utilización de protocolos estándares, no propietarios, aprobados por la Dirección de Informatización y el Grupo Técnico Central de la Producción, como son:
    - Para la descripción de los servicios Web se utilizará WSDL.
    - Para la comunicación entre servicios Web se utilizará SOAP.
    - Para el descubrimiento y documentación de servicios Web se utilizará Universal Descripción, Descubrimiento e Integración (UDDI).
    - Para la transferencia de datos entre servicios Web se utilizará XML.

La Universidad dispone de un grupo de sistemas que se clasifican como sistemas núcleos. En el núcleo se encuentran las aplicaciones que contienen la información base de las personas, entiéndase datos como: nombre, apellido, carné de identidad, y aplicaciones de uso común, como el Sistema de Seguridad o Identificación. En el grupo de aplicaciones núcleo están (Dirección de Informatización, 2007):

1. Gestión universitaria, contiene:
    - ✓ Los datos personales de los estudiantes.

- ✓ La gestión académica de las asignaturas y cursos.

2. Assets, tiene:
   - ✓ Los datos de los trabajadores que son plantilla de la UCI.
   - ✓ La información de las áreas pertenecientes a la UCI.
3. Sistema de Tercerizados y Eventuales, contiene:
   - ✓ La Información de tercerizados, es decir, personas que trabajan en la UCI, pero no pertenecen a la misma, sino a un área externa.
   - ✓ La información de los eventuales, es decir, personas que están trabando en algún área por un período limitado de tiempo.
   - ✓ La información de áreas tercerizadas, o sea, áreas que no pertenecen a la UCI.
4. Sistema de Residencia, presenta:
   - ✓ La información de la ubicación de las personas con beca en la Universidad.
   - ✓ La información de familiares.
5. Servicios Telemáticos, presenta:
   - ✓ La información de correo, login y grupos del dominio UCI.
6. Sistema de Seguridad, brinda:
   - ✓ Interfaz para facilitar el trabajo con la seguridad.
   - ✓ Métodos para la consulta de credenciales de usuarios.
   - ✓ La funcionalidad de controlar la autorización para los servicios que lo utilicen.
   - ✓ Reportes sobre la utilización de los sistemas en la Universidad.
7. Sistema de Identificación, facilita:
   - ✓ Información sobre los solapines de las personas.
   - ✓ La gestión de las fotos.

1.3.1. Situación actual de GeoQ-Guardián

Actualmente GeoQ-Guardián cuenta con varios proveedores para acceder a los datos que existen en la BD central. Cuando un planificador desea operar con los datos de las personas, el sistema los transfiere de forma temporal a una tabla creada por los desarrolladores. Desde esta tabla, se verifica a simple vista que no haya errores de existencia en los datos consultados, con respecto a los obtenidos por el usuario planificador y a la vez comprobados con otros orígenes de datos. Los usuarios con rol de planificador,

solamente acceden a las funcionalidades que ofrece GeoQ-Guardián, no tienen permiso sobre el gestor de BD.

Pueden existir datos que no sean válidos, asociados a una persona y siguen existiendo en la BD, sin embargo, el planificador solamente tiene constancia de que los datos son incorrectos, no los puede eliminar del origen. La actualización de los datos se sustenta de la periodicidad con que los desarrolladores del sistema puedan hacerlo de forma manual, además en el proceso de actualización se corre el riesgo de dañar la estructura y composición de la información. Esta última se organiza de acuerdo a la relación entre áreas y la asignación del personal a cada una de ellas; si no se tiene conocimiento del personal asociado a cada área, las tablas de la BD no quedan correctamente conformadas y el sistema puede visualizar capas vectoriales con errores en los campos.

Un ejemplo práctico de la importancia que arroja el procesamiento de versiones de datos actualizados, es la necesidad que tienen los planificadores de generar un reporte con la información asociada a los integrantes de una posta. En el caso de que se desee saber la facultad a la que pertenece cada integrante, se corre el riesgo de consultar un dato que dejó de ser real. Puede suceder que exista algún integrante en la posta que no pertenezca a la facultad que se le asigna porque se haya cambiado de área, y sin embargo el gestor de BD está proporcionando tal información. Esta deficiencia que el cliente no puede percibir y que además genera una situación problemática, es consecuencia de la dependencia que existe entre GeoQ-Guardián y la BD para consultar los datos de las personas.

1.4. Tecnologías de acceso a servicios Web

Existen diferentes herramientas que permiten consumir un grupo de operaciones disponibles en un servicio Web, en ese caso SharpDevelop, es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) de código abierto para el desarrollo de aplicaciones .NET que permite establecer un punto de enlace a través de una URL o un WSDL; esto genera una clase proxy con la que se puede acceder a las operaciones del servicio Web.

En el caso del IDE Eclipse se puede crear un cliente java y especificarle la URL del servicio Web a consumir, esto también construye un proxy con la lista de operaciones del servicio que pueden ser invocadas. Este proceso genera además un grupo de clases encargadas de proveer la información disponible en el servicio consultado, estas pueden ser vistas en la figura 2. Las clases terminadas en Return contienen los datos que devuelve el servicio Web. La clase Locator representa el servicio en sí, y la SoapStub realizará las llamadas a las operaciones. Para los clientes del servicio Web, la clase proxy

administra la asignación de parámetros a los elementos XML y envía a continuación, el mensaje SOAP a través de la red sobre HTTP.

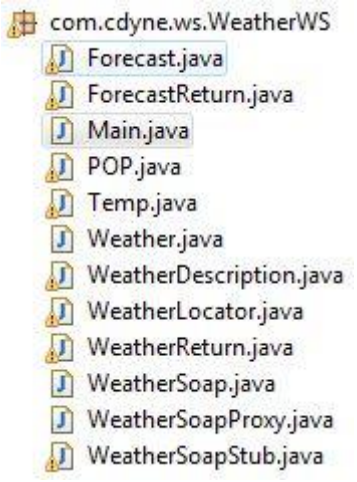


Fig 2. Clases del proxy

SoapUI es otra herramienta que permite a través de una interfaz gráfica, obtener información de los servicios Web, generando peticiones para cada método del servicio y visualizando las respuestas enviadas por el servidor. Permite identificar las operaciones asociadas con el servicio y hacer pruebas de funcionamiento, sin la necesidad de escribir código para crear clientes que consuman los servicios Web. Con esta herramienta se puede ver que datos y en que formato van a devolver los servicios Web los resultados del método utilizado, además de probar los servicios y ver que parámetros son necesarios para su uso (Junta de Andalucía , 2013).

Java API (Interfaz de programación de aplicaciones) *for* XML *Web Services* (JAX-WS), es un API estándar de Java para implementar e invocar servicios Web. Especifica un *mapping* de WSDL a Java, esto permite que las implementaciones de JAX-WS proporcionen un compilador de WSDL Java que genere diferentes proxys para invocar servicios Web. El modelo de ejecución se basa en recibir las peticiones SOAP sobre HTTP que envían los clientes, invocar la operación correspondiente al servicio Web y devolver una respuesta SOAP sobre HTTP con el resultado de la operación ejecutada (IBM, 2011).

1.4.4. Conclusiones parciales

El empleo de las herramientas mencionadas facilita el acceso a la información disponible en los servicios Web y basan su funcionamiento en la construcción de un recurso que actúa como un proxy, responsable de administrar el flujo de mensajes entre una aplicación cliente y el servicio consultado. La mayoría de estas herramientas son API integradas a Java y C#, no a Qt. El trabajo con el proxy incluye un grupo de clases auxiliares que no es necesario tenerlas en cuenta. La API QNetworkAccessManager en Qt puede sustituir el flujo de trabajo del proxy de una forma más simple, sin la necesidad de tener que generar una clase por cada operación existente en el servicio Web. Lo común en estas herramientas es el uso de estructuras estándares, como son los mensajes SOAP por la red sobre HTTP; lo que sirve de referencia para la construcción del sistema final, debido a que el negocio consiste en realizar llamas a un grupo de operaciones disponibles en un servicio Web.

1.5. Lenguaje de Programación.

La evolución de los lenguajes de programación continúa, tanto en la industria como en investigación y mientras más complejos sean los sistemas informáticos, nuevas versiones serán desarrolladas. Un caso específico es C++, lenguaje de propósito general, al que se le ha añadido nuevos tipos de datos, librerías y demás recursos que justifican su uso en el desarrollo de la plataforma GeoQ-Guardián.

1.5.1. ¿Por qué usar C++ como lenguaje de programación?

C++ es un lenguaje de programación con mecanismos que permiten la manipulación de objetos y se adapta a múltiples situaciones. Tiene como peculiaridad el trabajo con punteros, que serán útiles en la asignación de espacio en memoria una vez creados los objetos de diferentes tipos. La sobrecarga de sus operadores, posibilitará la asignación de diferentes valores a una misma variable. Permite además, a diferencia de otros lenguajes la herencia múltiple, necesaria en la declaración de clases que derivan de clases bases, donde estas últimas poseen funciones que pueden ser utilizadas sin la necesidad de reimplementarlas en las clases derivadas. Un caso de la herencia múltiple, es la interfaz visual que utiliza todas las funciones de la clase genérica MainWindow. Posee variadas bibliotecas externas que potencian y facilitan el trabajo a los programadores, ejemplo de esto es QtSoap que brinda la posibilidad de consumir servicios Web y además sirvió de objeto de estudio para el desarrollo de la solución propuesta. Es un lenguaje potente que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Luego de valorar este lenguaje de programación y con el fin de obtener un desarrollo eficiente del componente (proveedor de datos) en materia, se decide seleccionarlo como la mejor opción; por sus características y por su uso en el framework Qt y en QtCreator como IDE, establecidos estos en el expediente del proyecto SIG-DESKTOP y tratados a continuación.

1.6. Qt como framework en el desarrollo de GeoQ-Guardián

El término framework, hace alusión a una estructura de software compuesta por elementos personalizables e intercambiables para el desarrollo de una aplicación. Está asociado a un determinado tipo de aplicaciones, lo que implica que su alcance esté acotado. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta, configurable y abstracta.

Un caso específico es el framework multiplataforma Qt, que permite el desarrollo de interfaces gráficas de usuario y está incluido entre las herramientas libres más potentes para el desarrollo de casi cualquier tipo de software. Uno de los rasgos principales que presenta es el empleo del lenguaje nativo C++, lo cual permite generar aplicaciones sustentadas en la fortaleza de este lenguaje. Otra característica notable es la excelente documentación disponible para cualquier usuario que comience a trabajar con la herramienta. Cuenta además con una arquitectura lista para plugins, lo que permite el desarrollo de software mediante la integración con otros tipos de productos y framework. Cuenta con la integración de los siguientes componentes:

- ✓ **Librerías Qt:** son clases en C++.
- ✓ **QtDesigner:** para crear formularios visualmente utilizando QtAssistant que permite un acceso rápido a la documentación.
- ✓ **QtLinguist:** permite la traducción rápida de programas.
- ✓ **Qmake:** simplifica el proceso de construcción de proyectos en las diferentes plataformas soportadas.
- ✓ **QtCreator:** es un IDE (esto es, editor + compilador + depurador) bastante completo, potente, eficiente y abierto, que permite el desarrollo rápido de aplicaciones en Windows, Mac OS y Linux (Summerfield, 2008).

1.6.1. QtCreator como IDE de desarrollo

QtCreator es un IDE multiplataforma, se ejecuta en los sistemas operativos de escritorio de Windows, Linux/X11 y Mac OS X y permite a los desarrolladores crear aplicaciones para múltiples escritorios y plataformas de dispositivos móviles. Proporciona un grupo de funcionabilidades (Didia, 2013):

- ✓ Editor de código C++.
- ✓ Diseñador de interfaz de usuario integrado.
- ✓ Control de versiones.

Entre las características fundamentales de QtCreator se encuentran:

- ✓ Utiliza el lenguaje de programación orientado a objetos C++.
- ✓ Permite realizar programación visual y programación dirigida por eventos.
- ✓ Se basa en Qt, una librería multiplataforma y gratuita para la creación de interfaces gráficas, programación Web, multihilo y bases de datos.

1.7. Proceso Unificado Ágil para el desarrollo del sistema

El Proceso Unificado Ágil (AUP), surge a raíz de un grupo de criterios e inconformidades de los equipos de trabajo con respecto al exceso de disciplinas y documentación que generan el Proceso Unificado de Rational; aun cuando el producto a desarrollar no es de gran envergadura.

1.7.1. ¿Por qué se usó AUP?

El Proceso Unificado Ágil es una versión simplificada del Proceso Unificado de Rational (RUP). Describe la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún son válidos en RUP. AUP aplica técnicas que incluyen desarrollo dirigido por pruebas, modelado ágil, gestión de cambios ágil, y refactorización de base de datos para mejorar la productividad (Universidad Union, 2014).
"El proceso AUP establece un modelo más simple que el de RUP por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP" (Universidad Union, 2014). Existen varios autores que definen AUP cómo una variante de la metodología ágil eXtreme Programming (XP) pero con los artefactos de RUP que estime el equipo de trabajo.
AUP establece cuatro fases que se ejecutan de forma consecutiva (Universidad Union, 2014):

- ✓ **Incepción:** tiene como objetivos establecer una comprensión común cliente/equipo de desarrollo y definir una o varias arquitecturas para la construcción del sistema.
- ✓ **Elaboración:** los objetivos de esta fase son, lograr que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y validar la arquitectura.
- ✓ **Construcción:** a medida que se va desarrollando el sistema es probado en el ambiente operacional.
- ✓ **Transición:** el sistema se somete a pruebas de validación y aceptación y finalmente se despliega.

Las disciplinas son ejecutadas de manera iterativa, definiendo las actividades que se realizan por los miembros del equipo para construir, validar y liberar software funcional que cumpla con las necesidades de los involucrados. Las disciplinas son:

- ✓ **Modelo:** el objetivo es entender el negocio de la organización, el problema de dominio que se aborda y establecer una solución viable para resolver este problema.
- ✓ **Aplicación:** transforma el o los modelos en código ejecutable y realizar un nivel básico de pruebas.
- ✓ **Prueba:** esta disciplina exige una evaluación objetiva que garantice la calidad del producto. Esto incluye la búsqueda de defectos, validaciones funcionales del sistema y verificar que se cumplan los requisitos.
- ✓ **Despliegue:** exige como objetivo la prestación y ejecución del sistema y que el mismo esté a disposición de los usuarios finales.

Teniendo en cuenta los rasgos particulares de esta metodología, se seleccionó como propuesta para el desarrollo del proveedor de datos de servicios Web para GeoQ-Guardían. Las condiciones en el entorno de trabajo favorecen su empleo, se cuenta con la presencia a tiempo completo del líder y demás integrantes del proyecto, que actúan como clientes en el levantamiento de requisitos. La aplicación que se debe desarrollar como solución a la problemática actual, no requiere de un exceso de documentación pues es más importante el resultado. El número de personas implicadas en el desarrollo del producto no accede de 2 y el tiempo de entrega estimado siguiendo la experiencia de otros equipos de desarrollo, es de 5 meses.

1.8. Lenguaje unificado de modelado (UML) como herramienta para el diseño del sistema

El lenguaje unificado de modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un modelo, incluyendo aspectos conceptuales tales como: procesos de negocios y funciones del sistema, esquemas de bases de datos y componentes de software reutilizables. Este lenguaje de modelado ha sido diseñado para modelar cualquier tipo de proyecto, tanto informático como de arquitectura, o de cualquier otro ámbito (López, 2008).

Los objetivos de UML son muchos pero se pueden resumir en sus funciones:

- ✓ Visualizar: permite expresar de una forma gráfica el plano de un sistema.
- ✓ Especificar: permite definir las características de un sistema antes de su construcción.
- ✓ Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Un modelo UML está compuesto por tres clases de bloques de construcción:

- ✓ Elementos: son abstracciones reales o ficticias (objetos, acciones).
- ✓ Relaciones: relacionan los elementos entre sí.
- ✓ Diagramas: son colecciones de elementos con sus relaciones.

1.9. Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering - Ingeniería del Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas unidas al modelado con UML permite aplicar la metodología de análisis y diseño orientados a objetos y hacer abstracciones del código fuente, en un nivel donde la arquitectura y el diseño se tornan más fáciles de entender y modificar (López, 2008).

1.9.1. Visual Paradigm como herramienta CASE en el proceso de desarrollo

Visual Paradigm es una herramienta para el desarrollo de aplicaciones utilizando modelado UML. Esta herramienta facilita la generación de código desde diagramas, genera documentación y además es multiplataforma.

Dentro de las características y funcionalidades que lo destacan, entre las herramientas de su tipo que hace que sea tomado como herramienta CASE en la investigación, se encuentran las siguientes:

- ✓ Permite la creación de diagramas de proceso de negocio, decisión y actor de negocio.
- ✓ Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, prueba y despliegue.
- ✓ Es multiplataforma, soportado por Windows/Linux/Mac OS X.
- ✓ Trae consigo la generación de código, lo que agiliza el trabajo de los implementadores.

Conclusiones del capítulo

Se han analizado un grupo de aplicaciones que no solucionan la situación actual de GeoQ-Guardián, pero presentan formas de funcionamiento que pueden ser adaptadas a la solución propuesta. La Universidad no dispone de un servicio Web que contenga toda la información para la planificación de la guardia, por lo que un aspecto importante a tener en cuenta para el cumplimiento del objetivo general; es garantizar el procesamiento de la información proveniente de diferentes servicios Web. Para el desarrollo del software se tuvo en cuenta las herramientas y tecnologías utilizadas por GeoQ-Guardián por ser la plataforma base sobre la cual se desarrollará la solución propuesta. Por último, el trabajo con el lenguaje de programación C++ permite aprovechar la relación de integridad y compatibilidad entre el IDE QtCreator y el framework Qt.

Capítulo 2. Diseño de la solución propuesta

Introducción

En este capítulo se representan los artefactos necesarios para describir el funcionamiento de la aplicación. Se describen los requisitos funcionales y no funcionales, seguido se conforman los casos de uso y sobre la base de los casos de uso más críticos, se modela el flujo de trabajo del sistema en los diagramas de clases del diseño basados en la arquitectura seleccionada.

2.1. Modelo del dominio en el Desarrollo del Software

El modelo del dominio es una especie de representación de los conceptos u objetos más significativos dentro del dominio del problema, que incluye las clases de objetos, sus atributos y asociaciones. No enfatiza en el modo interno de cómo se ejecutan las operaciones, sino en la relación entre objetos del contexto operacional (Jacobson y otros, 2000). En otras palabras, el modelo del dominio ayuda a comprender el problema que se supone que el sistema resuelva en relación a su entorno.

Para realizar el modelo del dominio, se utilizan los diagramas de clases del UML. Para representar el contexto, hay que tener presente que se trata de realizar un modelo del entorno del software y no del software (Falgueras, 2003).

Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio. Las clases del dominio aparecen en tres formas (Jacobson y otros, 2000):

- ✓ Objetos del negocio, representan “cosas” que se manipulan en el negocio, como pedidos, cuentas y contratos.
- ✓ Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- ✓ Sucesos que ocurrirán o han ocurrido.

2.1.1. Entorno de trabajo del sistema GeoQ-Guardián

Teniendo en cuenta el entorno de trabajo donde se desarrolla el sistema y después de un estudio de la situación problemática, no es posible identificar con claridad los actores implicados en el negocio. La

estructura de la plataforma GeoQ-Guardián posibilita el trabajo con sus componentes de forma individual y la inclusión de nuevos componentes, por lo que a modo de facilitar el desarrollo de una aplicación que resuelva la problemática; se hace viable el enfoque solamente en elementos específicos del sistema, sin tener que modificar el resto. No es del interés del cliente, el funcionamiento interno de la solución, sino como se ejecuta en su entorno y quienes la operan. Estas son algunas de las razones que justifican el empleo de un modelo del dominio como medio para la modelación del problema que se supone resolver.

2.2. Modelo del dominio

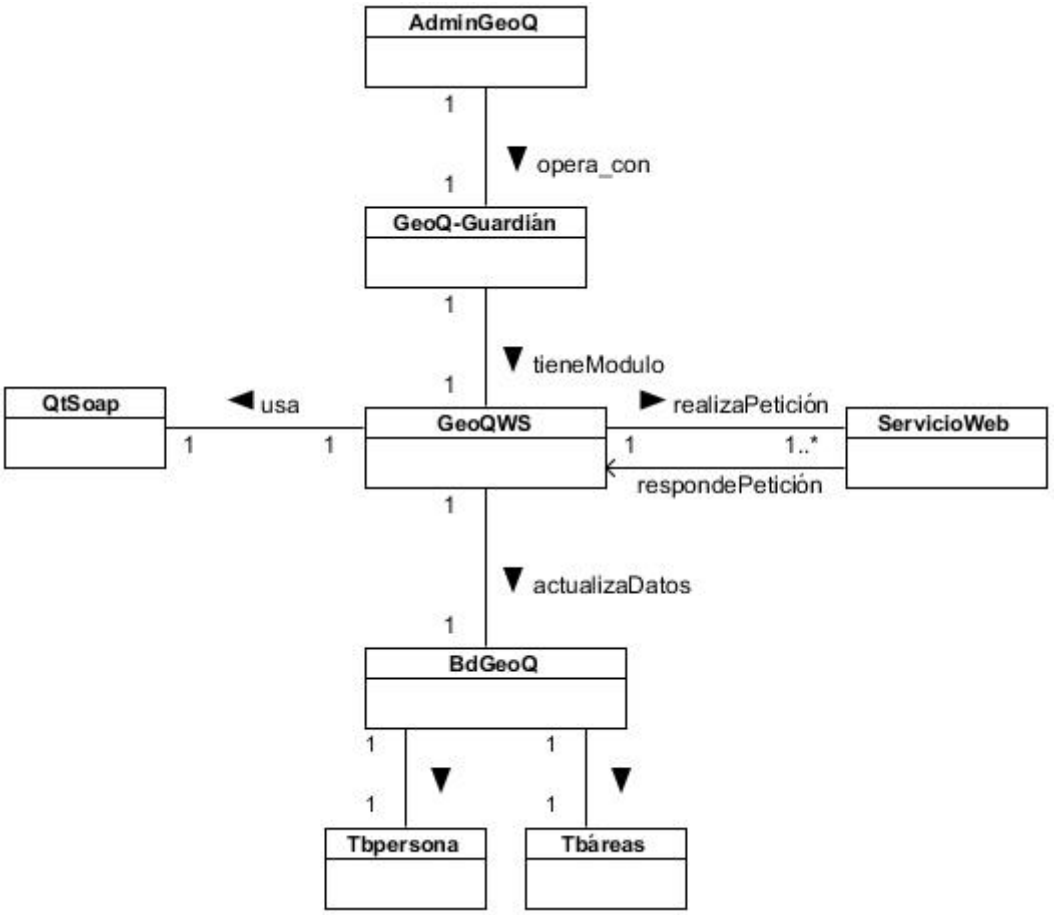


Fig 3. Modelo de dominio del negocio

2.2.1. Glosario de Términos del Dominio

El glosario de términos se genera cuando el negocio del dominio no es muy extenso y no es necesario desarrollar un modelo de objetos para el dominio. Tanto el glosario como el modelo del dominio ayudan a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común (Jacobson y otros, 2000). A continuación se describen los objetos identificados:

AdminGeoQ: usuario con el rol de administrador que posee todos los permisos sobre la aplicación GeoQ-Guardián, es el encargado de actualizar los datos en la BD central.

GeoQ-Guardián: sistema utilizado para la planificación de la guardia obrera-estudiantil.

QtSoap: librería utilizada por el componente GeoQWS para conectarse a los servicios Web.

GeoQWS: componente utilizado por GeoQ-Guardián para consultar y procesar los datos de los servicios Web.

ServicioWeb: grupo de servicios Web publicados por la Dirección de Informatización de la UCI para que las aplicaciones de la Universidad puedan consultarlos, según su negocio.

BdGeoQ: BD central de GeoQ-Guardián que contiene los datos utilizados para la planificación de la guardia.

Tbpersona: tabla de la BD central de GeoQ-Guardián que contiene los datos de la persona.

Tbáreas: tabla de la BD central de GeoQ-Guardián que contiene los datos de las áreas.

2.3. Requisitos Funcionales

2.3.1. Estrategia de captura de requisitos

Como método de apoyo para la captura de requisitos se tuvo en cuenta la tormenta de ideas, herramienta de trabajo en grupo basada en la creatividad y exposición de ideas sobre el problema a resolver y su posible solución; donde ningún criterio deberá ser desechado. La entrevista también cómo técnica de recopilación de información, permitió contactar con el líder del proyecto SIG-DESKTOP y administradores de los servicios Web en la Universidad; para recibir información puntual de las necesidades reales sobre los proveedores de datos y funcionamiento de los servicios Web (Ver Anexo1).

Requisitos Funcionales (RF)

Se identificaron los siguientes requisitos funcionales (RF):

RF1: Visualizar descripción de un servicio Web.

El usuario selecciona la dirección del servicio Web que desea consumir y el sistema visualiza la descripción del servicio Web.

Campo de entrada:

- ✓ Dirección del servicio Web seleccionada por el usuario (cadena de texto, obligatorio).

Campo de salida:

- ✓ El sistema visualiza la descripción del servicio Web.

RF2: Listar operaciones de un servicio Web.

Luego de seleccionada la dirección del servicio Web y visualizada su descripción, el sistema lista las operaciones que pueden ser ejecutadas por el usuario.

Campo de entrada:

- ✓ Dirección del servicio Web seleccionada por el usuario (cadena de texto, obligatorio).

Campo de salida:

- ✓ Se muestran las operaciones listadas en una tabla.

RF3: Consultar operación de un servicio Web.

El sistema debe consultar la operación seleccionada por el usuario y retornar los datos del servicio Web correspondiente a esa operación.

Campo de entrada:

- ✓ Operación seleccionada por el usuario (cadena de texto, obligatorio).

Campo de salida:

- ✓ Se muestra en una tabla los datos asociados a la operación seleccionada.

RF4: Crear capa vectorial.

El sistema debe crear capas vectoriales con los datos obtenidos de la operación selecciona por el usuario.

Campo de entrada:

- ✓ Datos asociados a la operación consultada por el usuario (cadena de texto, obligatorio).

Campo de salida:

- ✓ Capa creada con los datos de la operación.

RF5: Registrar dirección de un servicio Web.

El sistema debe permitir al usuario adicionar la dirección de un servicio Web.

Campo de entrada:

- ✓ Dirección del servicio Web (cadena de texto, obligatorio).

Campo de salida:

- ✓ El sistema muestra en una tabla la dirección adicionada.

RF6: Modificar dirección de un servicio Web.

El sistema debe permitir al usuario modificar la dirección de un servicio Web.

Campo de entrada:

- ✓ Dirección del servicio Web selecciona por el usuario en la tabla.

Campo de Salida:

- ✓ Dirección modificada.

RF7: Eliminar dirección de un servicio Web.

El sistema debe permitir al usuario eliminar la dirección de un servicio Web.

Campo de entrada:

- ✓ Dirección del servicio Web selecciona por el usuario en la tabla.

2.4. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto usable, rápido o confiable. Las cualidades que el producto debe tener son:

2.4.1. Usabilidad

- ✓ El sistema podrá ser usado por usuarios con conocimientos básicos en el manejo de computadoras.
- ✓ El sistema tendrá una correcta arquitectura de la información, con iconografía sugerente a la función a realizar y con colores acordes al resto de la aplicación GeoQ-Guardián.
- ✓ Las funcionalidades del sistema estarán agrupadas por zona de interés o grupos relacionados.

2.4.2. Rendimiento

Teniendo en cuenta que la solución se basa en consumir datos de un servicio Web:

- ✓ Cuando se consulta alguna operación de un servicio Web que contenga toda la información de las personas, el tiempo de respuesta no debe exceder los 7 000 milisegundos (7 segundo). En cualquier otro caso el tiempo de respuesta a las peticiones deben estar en un rango de 30 a 95 milisegundos.
- ✓ Si es necesario consultar varios operaciones de un servicio Web, el tiempo de respuesta no debe exceder a las 21 000 milisegundos (21 segundo).

2.4.3. Portabilidad

- ✓ La aplicación debe ser compatible con los sistemas operativos de Microsoft Windows a partir de su versión Windows 7 y en plataformas GNU/Linux a partir de su versión 12.04.

2.4.4. Seguridad

- ✓ La información manejada por el sistema estará protegida de acceso no autorizado y divulgación.

2.4.5. Apariencia o Interfaz Externa

- ✓ El sistema tendrá una interfaz gráfica fácil de usar por el usuario, por lo que no es necesaria una capacitación extensa para su uso.

2.4.6. Hardware

El sistema requiere de los siguientes elementos de hardware para cumplir con las especificaciones mencionadas en el requisito de rendimiento.

- ✓ Al menos 512 MB de memoria RAM, recomendado 1 GB.
- ✓ Procesador Pentium 512 MHz o superior.
- ✓ Al menos 2 GB libres de disco duro.
- ✓ Una velocidad de transmisión de datos por la red de :
    - Si es cable UTP, recomendado 100 Mbps.
    - Si es inalámbrica, recomendado 54 Mbps.

2.4.7. Restricciones de diseño e implementación

- ✓ Diseño sencillo, con pocas entradas, donde no sea necesario mucho entrenamiento para utilizar el sistema.
- ✓ El producto de software final debe diseñarse sobre una arquitectura en capas.

- ✓ Se debe lograr un producto altamente configurable y extensible, teniendo en cuenta que se desarrollará sobre la plataforma base GeoQ-Guardián.

2.4.8. Interfaces de Software

La construcción de la aplicación funcionará bajo los conceptos de arquitectura en capas.
Las PCs clientes:

- ✓ El proveedor de datos de los servicios Web UCI, debe ser un sistema multiplataforma que puede ejecutarse en los siguientes sistemas operativos:
  - GNU/Linux.
  - Windows 7 o superior.

2.4.9. Disponibilidad

- ✓ A los usuarios autorizados se les garantizará el acceso a la información.
- ✓ La información y las funcionalidades del sistema estarán disponibles siempre y cuando se trabaje en un escenario donde existan conexiones a red, cableada o inalámbrica.

2.5. Modelo del sistema propuesto

A continuación se presenta el modelo del sistema partiendo de la definición de actor del negocio:
"No todos los actores representan personas. Pueden ser actores, sistemas o hardware externo que interactuará con el sistema" (Jacobson y otros, 2000). El modelo del sistema representa la interacción de cualquier ente, interno o externo al sistema. En la tabla 1 se describen los actores implicados en el sistema.

Tabla 1. Descripción de los actores del sistema

Actores	Descripción
	Interviene en el proceso de: Visualizar descripción de un servicio Web, Listar operaciones de un servicio Web, Consultar operación de un servicio Web, Registrar dirección de un servicio Web, Modificar dirección de un servicio Web y Eliminar dirección de un servicio Web.

2.6. Diagrama de Casos de uso del sistema

El diagrama de casos de uso del sistema modela las funcionalidades del sistema desde el punto de vista de usuarios externos llamados actores. A continuación se muestra el diagrama de casos de uso del sistema donde se representan los actores y su relación con el sistema:

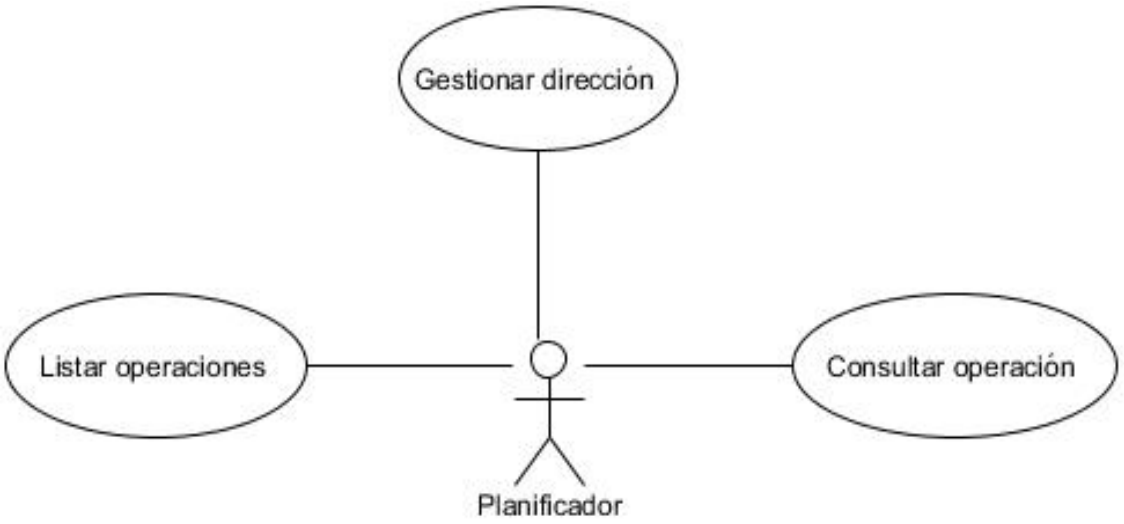


Fig 4. Diagrama de casos de uso del sistema

2.7. Descripción textual de los casos de uso del sistema

2.7.1. Caso de uso: Listar operaciones

Caso de Uso:	Listar operaciones
Actores:	Planificador
Resumen:	El caso de uso comienza cuando el planificador selecciona en la tabla de direcciones, la dirección del servicio Web que desea consultar. El caso de uso termina cuando el sistema visualiza la descripción del servicio Web y las operaciones referentes al servicio consultado.
Precondiciones:	• Debe existir en la tabla de direcciones, una dirección seleccionada. • La dirección seleccionada deben contener operaciones.

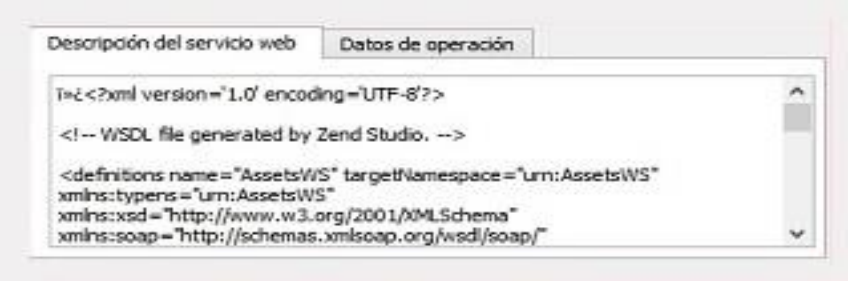
Referencias:	RF1, RF2
Prioridad:	Crítico

Flujo Normal de Eventos

Sección "Visualizar descripción de un servicio Web"

Acción del actor	Respuesta del Sistema
1.1 El actor selecciona la dirección del servicio Web que desea consultar.	1.2 El sistema visualiza la descripción del servicio Web. En caso contrario ver flujos alternos 1.2.

Prototipo de Interfaz



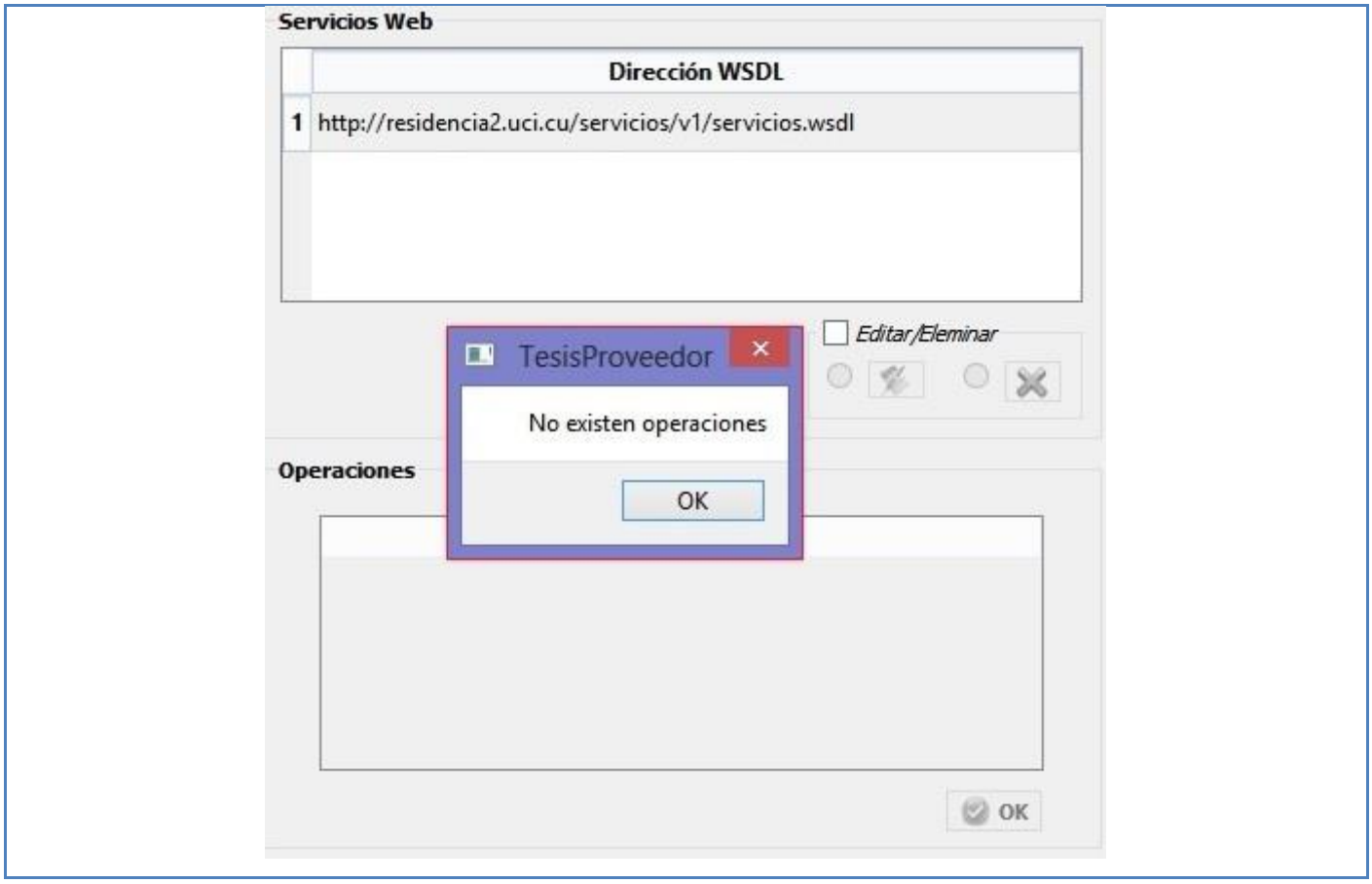
Flujos Alternos

Acción del actor	Respuesta del sistema
	1.2 Si el servicio Web no contiene una descripción, el sistema muestra un mensaje informativo.

Prototipo de Interfaz

Flujo Normal de Eventos **Sección "Listar operaciones de un servicio Web"**	
Acción del actor	**Respuesta del Sistema**
2.1 Realizar paso 1.1 de la sección "Visualizar descripción de un servicio Web".	2.2 El sistema visualiza en la tabla de operaciones, las operaciones correspondientes al servicio Web seleccionado, finalizando el caso de uso. En caso contrario ver flujos alternos 2.2.
Prototipo de Interfaz	

Flujos Alternos	
Acción del actor	**Respuesta del sistema**
	2.2 Si el servicio Web seleccionado no contiene operaciones, el sistema muestra un mensaje. Finalizando el caso de uso.
Prototipo de Interfaz	



El resto de las descripciones de los casos de uso del sistema se pueden consultar en el anexo 2.

2.8. Arquitectura del Sistema

2.8.1. Patrón arquitectónico. Arquitectura en capas

Un patrón arquitectónico expresa un esquema de organización estructural para sistemas de software. Proporciona un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y guías para organizar las relaciones entre ellos (Teniente López, y otros, 2003).

El patrón arquitectónico en capas, establece una organización de forma jerárquica tal que cada capa proporciona servicios a la capa inmediata superior, y se sirve de las prestaciones que le brinda la inmediata inferior (Reynoso, y otros, 2004). La comunicación solamente puede darse entre componentes de la misma capa o entre componentes de capas adyacentes. Garantiza un software mantenible, pues los

cambios realizados en un segmento de código no deberían influir en el resto del sistema. Los componentes de cada capa, pueden ser utilizados y reemplazados por implementaciones alternativas, cumpliendo con la reusabilidad, separación de interfaz e implementación. Las responsabilidades similares deberían agruparse en una capa, para favorecer la comprensibilidad, mantenibilidad y cohesión en el sistema (Teniente López, y otros, 2003).

Producto a los beneficios que reporta esta arquitectura, se garantiza la independencia de cada subsistema en la implementación de los requisitos a cumplir por la aplicación final. Como una primera vista genérica del software se pudiera establecer un grupo de capas o niveles, donde cada uno por separado responda a funcionalidades de validación y visualización de los datos al cliente, y por otra parte las reglas de negocio a tener en cuenta para el consumo y envío de datos desde los servicios Web.

La plataforma GeoQ-Guardían está basada en esta arquitectura, propiciando una mejor integración del software que se desea construir. El uso de este patrón permite en los primeros flujos de trabajo, reemplazar algunas reglas establecidas entre las capas vectoriales de GeoQ-Guardían para la manipulación de los datos obtenidos de los servicios Web. Este proceso se muy abstracto y se tiende a tomar conclusiones apresuradas por parte de los desarrolladores, de la forma de integración de estas capas. A continuación se representa en la figura 5 la arquitectura mencionada en su variante tres capas; capa de presentación, capa de negocio y capa de acceso a dato.

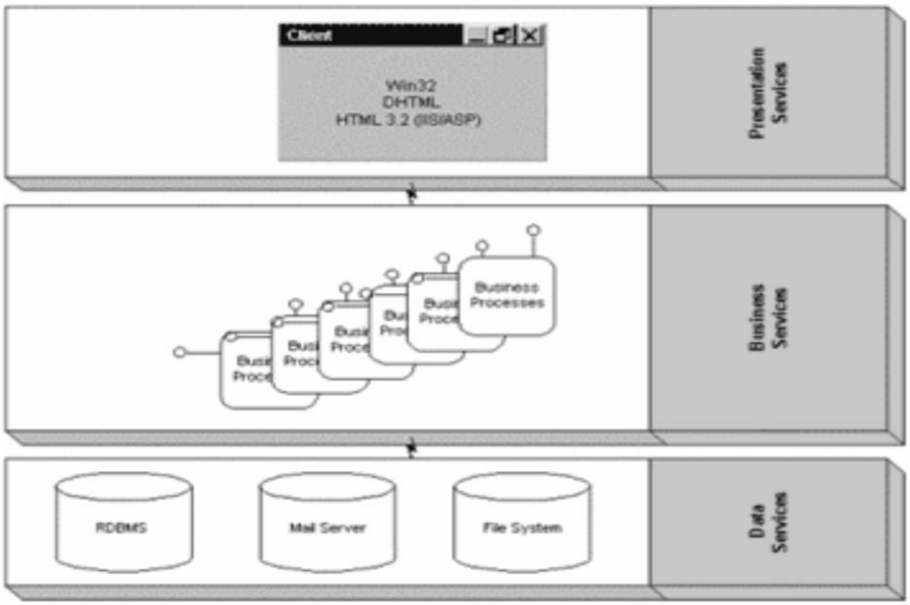


Fig 5. Arquitectura en tres capas

2.9. Modelo de diseño

El modelo de diseño se crea a partir del modelo de análisis, representa un esquema para la implementación y se adapta al entorno de trabajo elegido. Define clasificadores (clases, subsistemas e interfaces), relaciones entre esos clasificadores y colaboraciones que llevan a cabo los casos de uso. A diferencia del modelo de análisis que se centra en investigación del problema y no en la manera de definir la solución (Larman, 1999), el modelo de diseño es más “físico” y permite establecer una mejor vista de la arquitectura del sistema.

El modelo de análisis realiza abstracciones y evita resolver ciertos problemas y tratar ciertos requisitos. El resultado de una arquitectura puede conseguirse mediante la modificación de la estructura del modelo de análisis durante la transición al modelo de diseño (Jacobson y otros, 2000).

2.9.1. Diagrama de Clases del Diseño

Los diagramas de clases representan un conjunto de objetos y sus relaciones, identificados en los requerimientos funcionales y casos de uso de un escenario de negocios de un sistema. Cubren una vista de diseño estático desde la perspectiva de casos reales o prototípicos (Noguez y Valdivia, 2003). A continuación se representa el diagrama de clases del diseño del caso de uso listar operaciones y se explica su funcionamiento. Es resto de los diagramas se pueden consultar en el anexo 3.

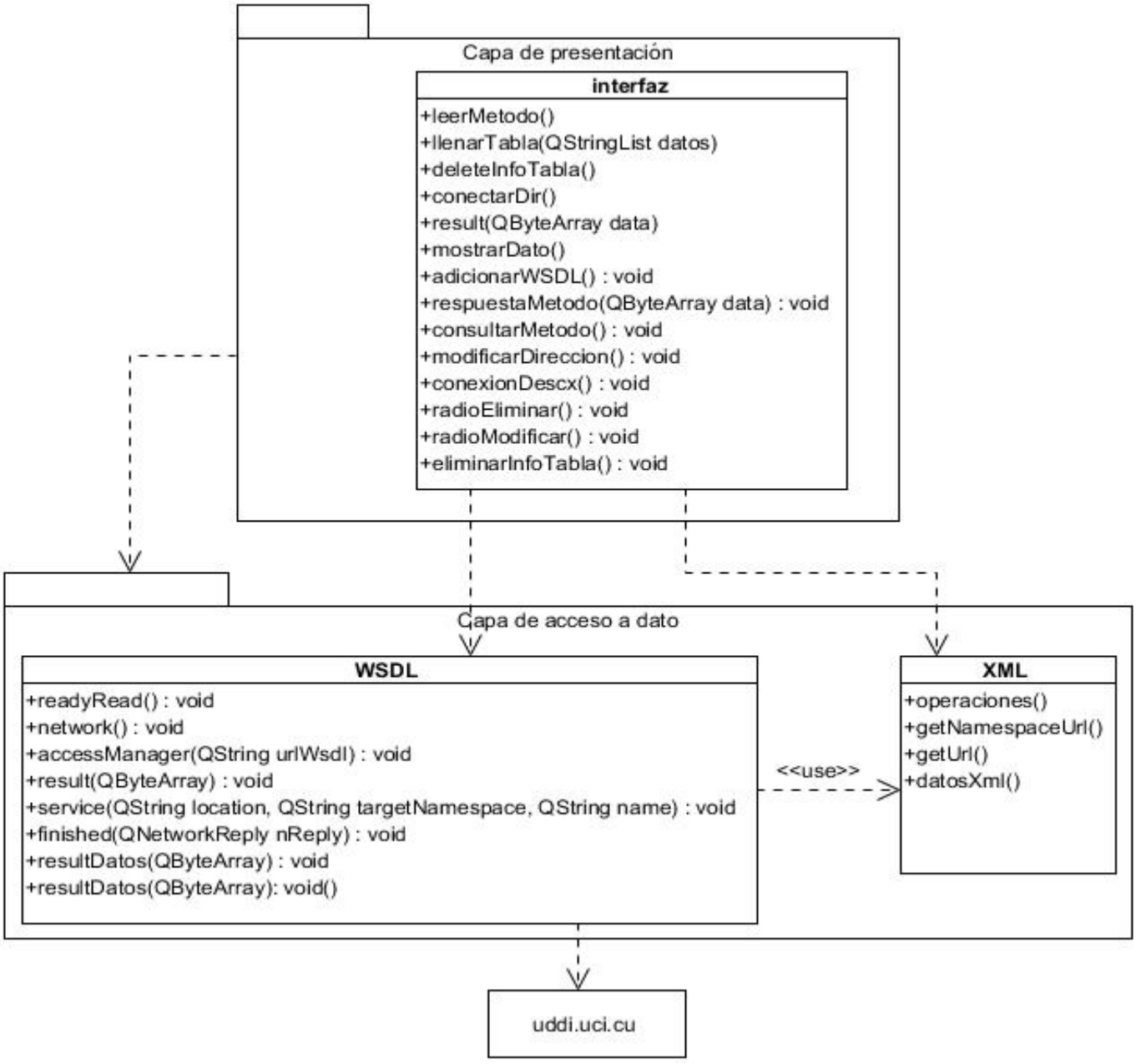


Fig 6. Diagrama de clases del diseño del caso de uso listar operaciones

Sobre la base de la arquitectura seleccionada en su variante 2 capas, se representa en la figura 6 la descripción del proceso del caso de uso listar operaciones. La capa 1, capa de presentación del negocio, permitió una fusión entre las clases encargadas de manipular objetos y eventos de la interfaz de la

aplicación, y las reglas del negocio. La capa 2, capa de acceso a datos, es responsable de consultar los datos disponibles en los servicios Web. En el ejemplo solo se representan las entidades que participan en el cumplimiento del caso de uso listar operaciones. En la capa de presentación se incluye un formulario responsable de visualizar toda la información necesaria para el usuario, y la clase interfaz es la encargada de capturar la selección por parte del usuario del servicio Web a consumir. Una vez capturada la dirección a consultar, se envía la solicitud a la capa 2, directamente a la clase WSDL, y se consultan los datos del origen uddi.uci.cu. Finalmente, la información consumida del servicio Web es mostrada al usuario mediante la clase interfaz y el formulario.

2.10. Patrones de diseño utilizados

"La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. Una implementación hábil se funda en los principios cardinales que rigen un buen desafío orientado a objetos" (Larman, 1999). En los Patrones Generales de Software para Asignación de Responsabilidades (General Responsibility Assignment Software Patterns, GRASP por sus siglas en inglés) se codifican algunos de ellos, que se aplican al preparar los diagramas de interacción, cuando se asignan las responsabilidades, o durante ambas actividades (Larman, 1999). Los patrones GRASP utilizados en el diseño del sistema se describen a continuación.

Creador: permite asignarle a una clase, la responsabilidad de crear instancias de otra. En este caso la clase interfaz es la encargada de manipular objetos de las clases XML y WSDL. Haciendo uso de objetos de este tipo, se pueden acceder a las operaciones asociadas a ellos. Además la clase interfaz que a la vez es una clase controladora de eventos entre el usuario y el sistema, tiene la responsabilidad de implementar el flujo de datos que se muestran en el formulario de la aplicación.

Experto: es asignado a la clase que cuenta con la información necesaria para cumplir una responsabilidad. La presencia de este patrón se evidencia en la clase XML, encargada de portar la descripción del servicio Web con sus operaciones y los datos asociados a las operaciones consultadas por el usuario.

Bajo acoplamiento: el acoplamiento es la medida de la fuerza con que una clase está conectada a otras clases, las conoce o recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras

(Larman, 1999); para mantener un bajo acoplamiento entre clases se debe garantizar una alta cohesión, con el fin de que las clases no dependan en gran medidas entre ellas. Un ejemplo de la presencia de este patrón es en la clase entidad XML, que contiene todas las operaciones asociadas a los servicios Web, y la clase controladora interfaz solo hace empleo de estas operaciones. Si existe algún cambio en la clase XML, no implica cambios significativos en la controladora.

Controlador: asigna la responsabilidad del manejo de una serie de eventos generados por un actor externo, es un evento de entrada externa (Larman, 1999). En la aplicación la clase controladora interfaz, es la encargada de operar con los eventos externos que el actor puede generar, y darle tratamiento. Un caso particular es cuando el usuario selecciona en la ventana principal de la aplicación, la dirección del servicio Web al que desea conectarse; la clase interfaz captura esa información y la sede a la clase WSDL que es la encargada de establecer la conexión con el servicio seleccionado.

Conclusiones del capítulo

De los objetos presentados en el modelo del dominio y sus relaciones, fueron reutilizados para modelar la solución propuesta, el actor planificador, el sistema GeoQ-Guardián y los servicios Web disponibles en la Universidad. El rendimiento de la aplicación depende en gran medida de los recursos de hardware y la infraestructura de red que exista en el entorno operacional del sistema GeoQ-Guardián. La selección del patrón arquitectónico en capas y el uso de los patrones GRAPS, ayudaron a modelar diferentes vistas del sistema y a agrupar por niveles las clases según sus funcionalidades, logrando una mejor comprensión del funcionamiento del software.

Capítulo 3. Implementación y pruebas a la solución propuesta

Introducción

En el presente capítulo se describe cómo los elementos identificados en el diseño se implementan en términos de componentes, especificando las clases y objetos utilizados en la implementación y cómo dependen unos de otros. Seguido se valida la aplicación mediante pruebas de software de caja negra y se especifican los casos de prueba realizados al sistema.

3.1. Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes, como ficheros de código fuente, ejecutables y otros. El modelo de implementación describe además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizados (Jacobson y otros, 2000).

3.1.1. Diagrama de Componentes

Un componente es una parte física de un sistema: modulo, BD, programa ejecutable, etc. Este diseño consiste en convertir el diseño de datos, interfaces y arquitectura en un software operacional. Con este diseño se proporciona un medio para evaluar el funcionamiento de las estructuras de datos, interfaces y algoritmos (Pressman, 2001).

Los diagramas de componentes expuestos a continuación han sido definidos por casos de uso y sobre la base del patrón arquitectónico en capas. La figura 7 muestra el diagrama de componentes del caso de uso Listar operaciones, los demás diagramas se encuentran en el anexo 4.

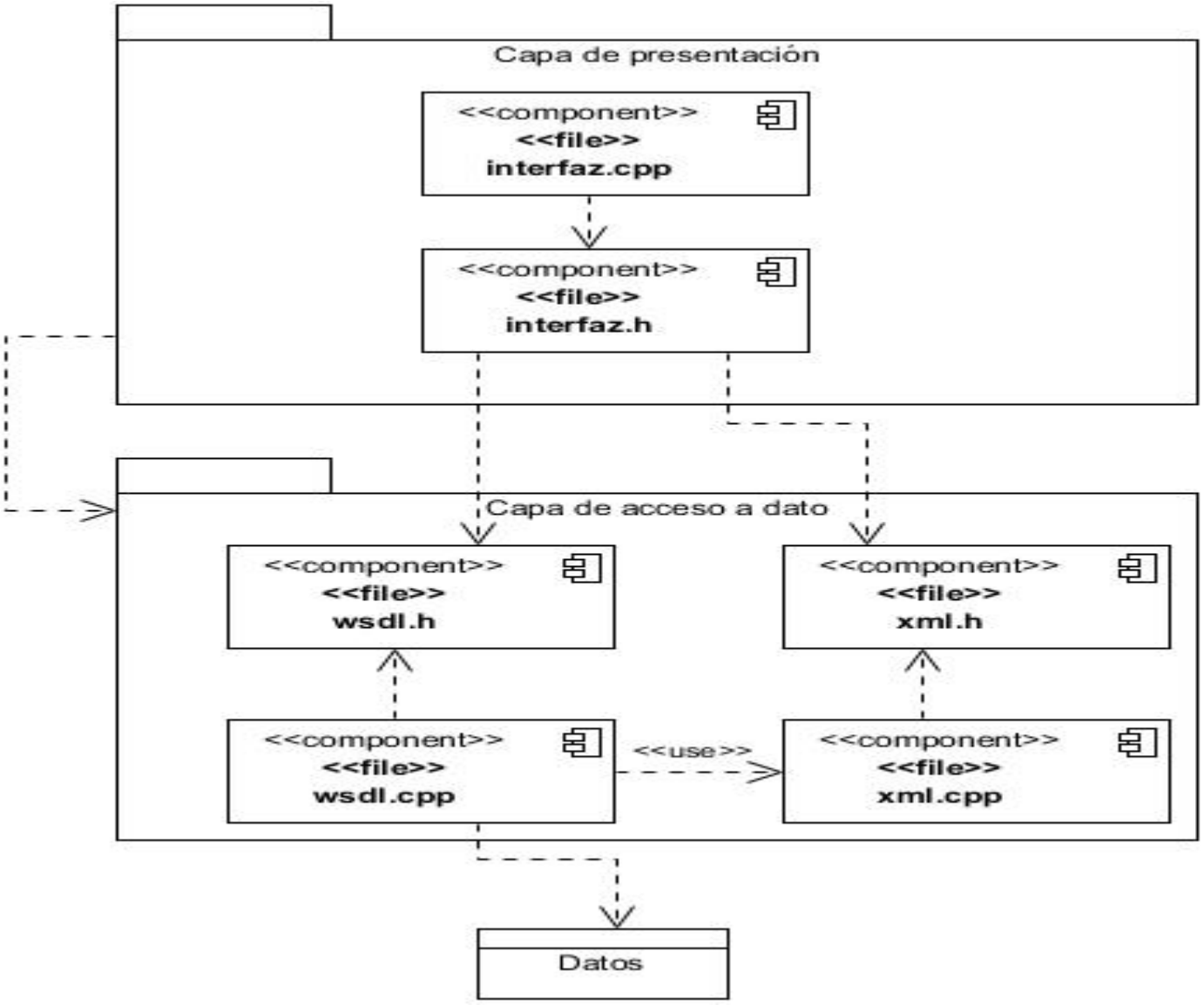


Fig 7. Diagrama de componentes del caso de uso Listar operaciones.

3.2. Modelo de Despliegue

El modelo de despliegue describe la distribución física del sistema, muestra cómo están distribuidos los componentes de software entre los nodos de cómputo (Jacobson y otros, 2000). La figura 7 representa la distribución física de la solución propuesta.

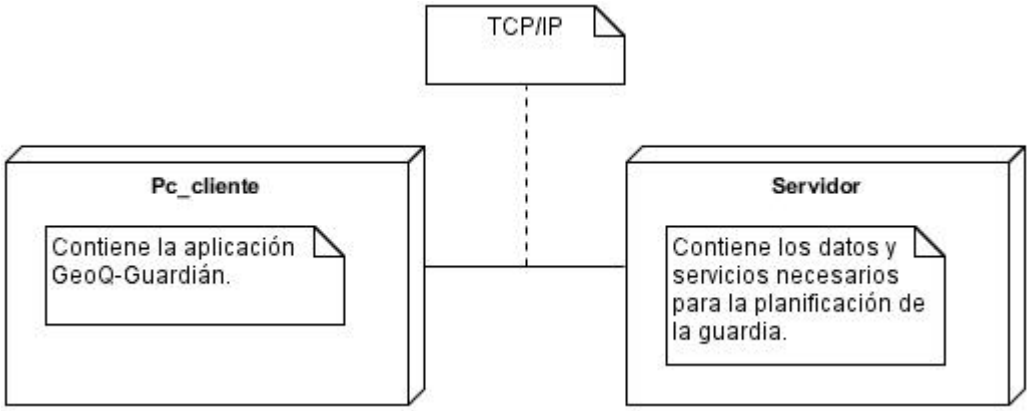


Fig 8. Modelo de despliegue de la solución propuesta.

3.3. Resultados de la implementación

Después del total cumplimiento de las tareas trazadas durante el diseño e implementación del software, se obtuvo un conjunto de resultados que a continuación se mencionan:

- ✓ Para el envío y recibo de mensajes se utilizó SOAP y HTTP, teniendo en cuenta que son estos los protocolos utilizados para la publicación de los servicios Web en la UCI.
- ✓ Para des-serializar los mensajes recibidos en XML, se empleó una función basada en el parseo de objetos del tipo QDomDocument y QDomElement para interpretar las etiquetas del archivo XML como nodos.
- ✓ Para poder acceder a las operaciones disponibles en los servicios Web, se obtuvo del proceso de parseo del WSDL, 3 parámetros necesarios para poder consultar estas operaciones, ellos son: la dirección URL del servicio Web, un ejemplo: http://akademos2.uci.cu/servicios/v4/ws_akademos.php, el Namespace para distinguir entre los servicios con el mismo nombre pero con diferentes propósitos u orígenes, un ejemplo: urn:akademos, y finalmente el nombre de la operación a consultar.
- ✓ Se definió una clase controladora WSDL encargada de establecer la conexión y otra clase XML encargada de contener la información de los servicios Web, estas 2 clases se relacionan de la siguiente manera; cuando la clase WSDL se conecta al servicio Web, esta envía a XML el mensaje obtenido, y en XML se realiza el parseo y obtención de la información necesaria para el resto del sistema.

- ✓ Teniendo en cuenta que actualmente no existe un servicio Web publicado en la Universidad que contenga todos los datos necesarios para la planificación de la guardia en GeoQ-Guardían, se dio la posibilidad de poder consultar varias operaciones disponibles en varios servicios Web, mostrar en tablas los datos de estas operaciones y finalmente conformar la información necesaria para la planificación de la guardia.
- ✓ Las conexiones con WSDL se realizaron con objetos del tipo QNetworkAccessManager, QNetworkReply y QNetworkRequest.
    - QNetworkAccessManager es la clase que permitió el envío de solicitudes y recibo de respuestas por la red. Esta clase contiene la configuración de un proxy, así como un grupo de señales de respuestas que se utilizaron para controlar el progreso de una operación realizada por la red. Con la creación de un solo objeto de este tipo en el sistema, se garantizó la realización de las peticiones a los servicios. Cada operación asociada al objeto QNetworkAccessManager retorna un objeto del tipo QNetworkReply, este objeto devuelto se utilizó para obtener los datos enviados en la respuesta a la solicitud correspondiente.
    - La clase QNetworkReply como ya se mencionó contiene los datos relacionados con la solicitud realizada con QNetworkAccessManager. Además posibilitó el acceso secuencial a la información consultada, esto significa que una vez recibidos los datos por la red se emite la señal readyRead() indicando que hay datos listos para leer, cuando el proceso ha terminado se emite la señal finished(), estos datos son destruidos pero almacenados en una variable temporal y cedidos a la clase XML.
    - QNetworkReques es la clase que contiene la información necesaria para enviar una petición por la red con QNetworkAccessManager. En este caso la información especificada fue, el encabezado de la petición (por ejemplo: *<?xml version=\"1.0\" encoding=\"UTF-8\"?>*) y la dirección URL (por ejemplo: http://akademos2.uci.cu/servicios/v4/AkademosWS.wsdl).
- ✓ La creación de las capas vectoriales se realizó de acuerdo a las operaciones consultadas por el proveedor de datos, cada capa vectorial se asoció a una operación.

3.4. Pruebas del sistema propuesto

El proceso de pruebas permite determinar el estado de la calidad de un producto software. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante técnicas de prueba (PRUEBASDESOFTWARE, 2005). Un caso específico son las pruebas de caja negra que se mencionan a continuación.

3.4.1. Prueba de caja negra

Las pruebas de caja negra son realizadas sobre la base de la interfaz del software y apoyándose en los casos de prueba pretenden demostrar que las funciones definidas previamente como requisitos son operativas, que las entradas se aceptan de forma adecuada y que se produce el resultado correcto, estas pruebas (Juristo y otros, 2005):

- ✓ Verifican las especificaciones funcionales y no consideran la estructura interna del programa.
- ✓ Son hechas sin el conocimiento interno del producto.
- ✓ No validan funciones ocultas, por tanto los errores asociados a ellas no serán encontrados.

Para la validación del sistema se seleccionó la técnica partición equivalente. Este método de prueba de caja negra divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada (Gil, 2007). A continuación en la tabla 2 se muestra la técnica partición equivalente aplicada al caso de uso Listar operaciones y en el anexo 5 se encuentra la aplicada al caso de uso Consultar operación.

Tabla 2. Caso de prueba del caso de uso Listar operaciones

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**
SC 1: "Visualizar descripción de un servicio Web"	EC 1.1: Visualizar descripción del servicio Web.	El planificador selecciona en la tabla "Dirección WSDL" la dirección del servicio Web a consumir. El sistema ejecuta la solicitud y visualiza en el	Proveedor: 1. Ventana principal. 2. El usuario hace clic izquierdo en la tabla

		campo “Descripción del servicio Web”, la descripción del servicio.	“Dirección WSDL” sobre la dirección del servicio Web a consumir. 3. El sistema visualiza en el campo “Descripción del servicio Web”, la descripción del servicio.
	EC 1.2: No se visualiza la descripción del servicio Web.	El planificador selecciona en la tabla “Dirección WSDL” la dirección del servicio Web a consumir. Si el servicio Web tiene errores, el sistema muestra en el campo “Descripción del servicio Web”, un mensaje informativo en función del error generado.	Proveedor: 1. Ventana principal. 2. El usuario hace clic izquierdo en la tabla “Dirección WSDL” sobre la dirección del servicio Web a consumir. 3. El sistema visualiza en el campo “Descripción del servicio Web”, el error ocurrido.
SC 2: “Listar operaciones de un servicio Web”	EC 2.1: Listar las operaciones del servicio Web.	El planificador selecciona en la tabla “Dirección WSDL” la dirección del servicio Web a consumir. El sistema ejecuta la solicitud y visualiza en la tabla de “Operaciones” el listado de operaciones del servicio Web.	Proveedor: 1. Ventana principal. 2. El usuario hace clic izquierdo en la tabla “Dirección WSDL” sobre la dirección del servicio Web a consumir. 3. El sistema visualiza en la tabla “Operaciones”, el listado de operaciones del servicio Web.
	EC 2.2: El servicio	El planificador selecciona en la tabla	Proveedor:

	Web no contiene operaciones.	"Dirección WSDL" la dirección del servicio Web a consumir. Si el servicio Web no contiene operaciones, el sistema muestra un mensaje "No existen operaciones".	1. Ventana principal. 2. El usuario hace clic izquierdo en la tabla "Dirección WSDL" sobre la dirección del servicio Web a consumir. 3. El sistema muestra un mensaje.

Tabla 3. Descripción de las variables del caso de prueba Listar operaciones

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Dirección WSDL	QTableWidget	No	El usuario selecciona en la tabla "Dirección WSDL", el servicio a consumir.

Tabla 4. Matriz de datos del caso de prueba Listar operaciones

Id del escenario	Dirección WSDL	Respuesta del sistema	Resultado de la prueba
EC 1.1	V "http://akademos2.uci.cu/servicios/v4/AkademosWS.wsdl"	El sistema visualiza la descripción del servicio Web consumido.	Satisfactorio
EC 1.2	IN "http://personal.uci.cu/servicios/v4/PersonalpWS.wsdl"	El sistema muestra un mensaje informando que no encuentra el servicio especificado.	Satisfactorio

Id del escenario	Dirección WSDL	Respuesta del sistema	Resultado de la prueba
EC 2.1	V "http://akademos2.uci.cu/servicios/v4/AkademosWS.wsdl"	El sistema lista las operaciones del servicio Web.	Satisfactorio
EC 2.2	IN "http://personal.uci.cu/servicios/v4/PersonalpWS.wsdl"	El sistema muestra un mensaje informando que no existen operaciones en el servicio Web.	Satisfactorio

Después de haber sido aplicados los casos de prueba al sistema, se detectaron en una primera iteración 5 no conformidades asociadas a errores de interfaz y de validación, las cuales fueron resueltas en un período de tiempo de 2 días. Se realizó una segunda iteración y se detectaron 4 no conformidades, asociadas a errores ortográficos y de validación, siendo solucionadas en 1 día. En una tercera iteración no se detectaron no conformidades siendo eliminadas en su totalidad en las dos iteraciones anteriores.

Conclusiones del capítulo

En este capítulo se desarrollaron los principales artefactos para llevar a cabo el proceso de implementación del sistema. La realización del Modelo de Implementación, permitió detallar los componentes utilizados y la relación entre ellos en el desarrollo de la aplicación. En el proceso de implementación el uso de protocolos estandarizados como SOAP, HTTP, el uso de las clases QNetworkAccessManager, QNetworkReply y QNetworkRequest; permitió obtener un software reutilizable. Se llevó a cabo el diseño de pruebas de caja negra, con el que se pudo probar el funcionamiento de cada uno de los elementos que componen la interfaz de la aplicación, permitiendo así validar la solución propuesta.

Conclusiones generales

La interoperabilidad de los servicios Web y el uso de protocolos estandarizados como SOAP y HTTP, permitieron mediante el proveedor de datos el acceso a la información disponible en estos servicios desde la plataforma GeoQ-Guardián. El estudio de la estructura del proxy generado por las herramientas analizadas en el epígrafe 1.4, permitió implementar una solución similar y adaptada a la necesidad de GeoQ-Guardián. El desarrollo de la aplicación, dirigido por las actividades establecidas en el marco de trabajo de la metodología AUP, logró una organización en el proceso de diseño e implementación del sistema. El uso del patrón arquitectónico en capas unido a los patrones GRASP, permitió en la implementación del sistema, agrupar las responsabilidades similares para favorecer la comprensibilidad y cohesión del software. El diseño de las pruebas de caja negra permitió la validación del sistema propuesto y verificar el cumplimiento de los requisitos funcionales capturados.

Recomendaciones

- ✓ Realizar otros tipos de pruebas al software construido.
- ✓ Mejorar el funcionamiento de las operaciones que garantizan la conexión a los servicios Web en el sistema.

Bibliografía citada

1. **Confederación Hidrográfica del Júcar. 2010.** *http://www.chj.es/es-es/Organismo/Paginas/Organismo.aspx.* [En línea] 2010. [Citado el: 9 de diciembre de 2013]. http://www.chj.es/es-es/ciudadano/Reutilizacion/Documents/GUIA%20USO%20DE%20SERVICIO%20WMS.pdf.
2. **Consortium, World Wide Web. 2004.** [En línea] 2004.
3. **del Río San, José Jorge. 2010.** *Introducción al tratamiento de datos espaciales en hidrografía .* s.l. : Bubok, 2010. ISBN: 8490093865.
4. **Didia. 2013.** *SOFTPEDIA.* [En línea] SoftNews NET SRL, 12 de diciembre de 2013. [Citado el: 11 de febrero de 2014]. http://www.softpedia.es/programa-Qt-Creator-167814.html.
5. **Dirección de Informatización. UCI. 2007.** Directorio de Servicios Web UCI. *http://uddi.uci.cu/.* [En línea] mayo de 2007. [Citado el: 6 de diciembre de 2013]. http://uddi.uci.cu/files/arquitectura.2007.5.9.pdf.
6. **Falgueras Campderrich, Benet. 2003.** *Ingeniería de Software.* Barcelona : UOC, 2003. ISNB: 84-8318-997-6.
7. **García Peñalvo, Francisco José y García Carrasco, Joaquín. 2009.** *http://www.usal.es/webusal/.* [En línea] Universidad de Salamanga, 4 de diciembre de 2009. [Citado el: 2 de mayo de 2014]. http://campus.usal.es/~teoriaeducacion/rev_numero_03/n3_art_garcia-garcia.htm.
8. **Gil, Manuel Torres. 2007.** *http://indalog.ual.es.* [En línea] 2007. [Citado el: 5 de abril de 2014]. http://indalog.ual.es/mtorres/LP/Prueba.pdf.
9. **IBM. 2011**. IBM Knowledge Center. [En línea] 13 de julio de 2011. [Citado el: 18 de mayo de 2014]. http://www-01.ibm.com/support/knowledgecenter/SS7K4U_7.0.0/com.ibm.websphere.zseries.doc/info/zseries/ae/cwbs_jaxws.html?lang=es.
10. **Jacobson, Ivan; Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson educación, S.A, 2000. ISBN: 84-7829-036-2.
11. **Jaramillo González, Víctor Hugo. 2007.** Academia.edu. *http://www.academia.edu/.* [En línea] Universidad Técnica Particular de Loja. Ecuador. 2007. [Citado el: 2 de mayo de 2014].

http://www.academia.edu/6440848/INFRAESTRUCTURA_DE_DATOS_ESPACIALES_IDE_PARA_EL_ESTUDIO_Y_ANALISIS_AMBIENTAL_UNA_EXPERIENCIA_EN_EL_SUR_DEL_ECUADOR.

12. **Junta de Andalucía. 2013**. Marco de Desarrollo de la Junta de Andalucía. [En línea] 1de marzo de 2013. [Citado el: 18 de mayo de 2014]. http://www.juntadeandalucia.es/servicios/madeja/contenido.
13. **Larman, Craig. 1999.** *UML Y PATRONES.* México : Dawn Speth While , 1999. ISBN: 970-17-0261-1.
14. **López Padilla, Felicidad. 2008.** *Departamento de Sistemas Informáticos.* [En línea] junio de 2008. [Citado el: 11 de febrero de 2014]. http://www.dsi.uclm.es/personal/franciscomsimarro/pfc/pfc07/doc/feli08.pdf.
15. **Machuca Morales, Carlos Andrés. 2008.** *https://sites.google.com/site/camoralesma/.* [En línea] 2008. [Citado el: 6 de marzo de 2014].
16. **Marco de Desarrollo de la Funta de Andalucía . s/f.** *http://www.juntadeandalucia.es.* [En línea] s/f. [Citado el: 9 de diciembre de 2013].
17. **Mora García, Nelly. 2009.** *Diseño e implementación de un Servidor de Mapas Web para una Red Bioclimática en Montaña.* Bolívia : s.n., 2009.
18. **Noguez M, Julieta y Valdivia B, Roberto. 2003.** *Lenguaje Unificado de Desarrollo. México*: Tecnológico de Monterrey, 2003.
19. **OSGeoLive.** *http://live.osgeo.org/es/index.html.* [En línea] Disclaimer. [Citado el: 2 de mayo de 2014]. http://live.osgeo.org/es/overview/qgis_mapserver_overview.html.
20. **Pressman, Roger S. 2001.** *Ingeniería de Software. Un enfoque práctico.* Madrid y Carchelejo : s.n., 2001.
21. **RAE. 2001.** *http://www.rae.es/.* [En línea] 2001. [Citado el: 6 de marzo de 2014]. http://buscon.rae.es/drae/?type=3&val=inform%C3%A1tica&val_aux=&origen=REDRAE.
22. **RedLatinGeo. 2001.** Colaboración de Tecnologías de la Información Geográfica . *http://redgeomatica.rediris.es/.* [En línea] 9 de mayo de 2001. [Citado el: 11 de diciembre de 2014]. http://redgeomatica.rediris.es/metadatos/publica/recetario/html/capitulo06.html.
23. **Reyes, Barzaga. 2013.** *renia.cujae.edu.cu.* [En línea] 2013. [Citado el: 14 de diciembre de 2013]. http://renia.cujae.edu.cu/index.php/revistacientifica/article/view/182/119.
24. **Reynoso, Carlos y Kiccillof, Nicolás. 2004.** *Estilos y Patrones en la Estratergia de Arquitecturas de Microsoft.* Buenos Aires: Universidad de Buenos Aires.

25. **Sistema de Información Geográfica de la Universidad de Alicante. s/f.** SIGUA. *http://www.sigua.ua.es.* [En línea] s/f. [Citado el: 9 de diciembre de 2013]. http://www.sigua.ua.es/web/utils/ogc/wfs.php?idioma=va.
26. **Sistemas GeoInformáticos SGI . 2007.** Sistemas GeoInformáticos. [En línea] 2007. [Citado el: 9 de diciembre de 2013]. http://sginicaragua.com/index.php?id=5.
27. **Sommerville, Ian. 2005.** *Ingeniería de software.* Madrid : PEARSON EDUCATION, 2005. ISNB: 84-7829-074-5.
28. **Summerfield, J.B.M. 2008.** *C++ GUI Programming with Qt 4.* 2008.
29. **Taborga, Oscar. 2012.** Proyecto de los hermanos Carrero. *Programación en Castellano.*http://www.programacion.com/articulo/arquitectura_y_funcionalidad_de_ado_net_312/2. [En línea] 2012. [Citado el: 25 de octubre de 2013].
30. **Universidad Tecnica del Norte. 2004.** Repositorio Digital. *http://repositorio.utn.edu.ec/.* [En línea] 2004. [Citado el: 19 de octubre de 2013]. http://repositorio.utn.edu.ec/bitstream/123456789/587/1/CAPITULOS.pdf.
31. **Universidad Union Bolivariana. 2014.** mex.tl. *http://ingenieriadesoftware.mex.tl/.* [En línea] Ingenieria de Software, 2014. [Citado el: 28 de enero de 2014]. http://ingenieriadesoftware.mex.tl/52788_Rup-Agil.html.
32. **Zucchi, Lara; Navarro, Magali; Vazquez, Florencia; Martin, Lucia y Cabral, Micaela. 2013.** Educación Tecnológica. *SlideShare.* [En línea] 6 de mayo de 2013. [Citado el: 19 de octubre de 2013]. http://www.slideshare.net/valcalde/la-brecha-digital-teoria.

Bibliografía

1. **Algesa. 2013.** ALGESA.com.ra. *ALEGSA.* [En línea] Santa Fe, Argentina, 2013. [Citado el: 25 de 1 de 2014]. http://www.alegsa.com.ar/Dic/lenguaje%20de%20programacion.php.
2. **Anglada Martínez, Ramón Alexander y Garófalo Hernández, Alain Abel. 2013.** *Tecnologías de la información y las telecomunicaciones.*2, La Habana : Ediciones Futuro, 2013, Vol. 7. ISSN: 2227-1899.
3. **Benedí Pérez, Jennier. s/f.** [En línea] s/f. [Citado el: 6 de diciembre de 2013]. http://oa.upm.es/1387/1/PFC_PABLO_ORTIZ_LOURDES_RIESTRA.pdf.
4. **Cali, Santiago. 2006.** Base de Datos: Conceptualización y sistemas de administración. [En línea] 25 noviembre de 2011. http://www.buenastareas.com/ensayos/Bases-De-Datos-Conceptos-Basico/1536159.html.
5. **Camacho Ortiz, Maria Fernanda. 2012.** Prezi. [En línea] 13 de septiembre de 2012. [Citado el: 6 de diciembre de 2013]. http://prezi.com/yhkgillpm6xk/datos/.
6. **Carrillo Pérez, Isaías, Pérez Gonzáles, Rodrigo y Rodríguez Martín, Aureliano David. 2008.** [En línea] 15 de octubre de 2008. [Citado el: 19 de enero de 2014].
7. **Chirino, Alfredo. 2011.** *http://www.ocw.uned.ac.cr.* [En línea] 2011. [Citado el: 9 de noviembre de 2013]. http://www.ocw.uned.ac.cr/eduCommons/s.e.p/tecnologia-y-trabajo/tutorias/tutoria-segunda/presentacion-etapas-del-procesamiento-de-datos.
8. **Codd, E. F. 2011.** The Relational Model for Database Management: Version 2. Reading, Mass. Los Altos, CA: Addison-Wesley Publishing.
9. **Date, Christopher. 2003.** Introducción a los Sistemas de Bases de Datos. Primera Parte. La Habana: Felix Varela, 2003.
10. **Eeles, Peter. 2001.** Layering Strategies. *Rational Software White Paper.* s.l. : Rational the software development company, 2001.
11. **Fletes Gudiño, Pedro y otros. 2007.** Departamento de Sistemas y Computación. *http://labredes.itcolima.edu.mx/fundamentosbd/index2.html.* [En línea] Instituto Tecnológico de Colima, 2007. [Citado el: 9 de Marzo de 2014]. http://labredes.itcolima.edu.mx/fundamentosbd/sd_u1_6.htm.

12. **Gómez, Joaquin Andreu. 2011.** *Redes Locales .* s.l. : Editex , 2011. ISBN: 8497719727.
13. **González Morales, Julio Heberto. 2004.** Web Services mediante WebServiceConnector de flash. *Cristalab.* [En línea] 2004.
14. **Gutierrez, Demián. 2010.** *Frameworks y Componentes.* [En línea] abril de 2010. [Citado el: 11 de febrero de 2014]. http://www.codecompiling.net/files/slides/IS_clase_10_frameworks_componentes.pdf.
15. **Gutierrez Saavedra, Jorge A. 2007.** *El Mundo Informático.* [En línea] WordPress.com, 2007. [Citado el: 21 de 1 de 2014]. http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion.
16. **Heber, Romero. 2009.** Gestión del conocimiento y la información en el Polo de Gestión Universitaria. Universidad de las Ciencias Informáticas. Ciudad de la Habana: s.n., 2009.
17. **Honorato Ponce, Jose Miguel. 2011.** *http://openaccess.uoc.edu.* [En línea] 2011. [Citado el: 2 de mayo de 2014]. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/8145/1/jponcehPFC0611.pdf.
18. **IDEPA.** *Infraestructura de Datos Espaciales (LÍNEA-IDE).* [En línea] [Citado el: 28 de noviembre de 2013]. http://www2.idepa.es/idesipa/servicioWFS.asp.
19. **Juristo, Natalia; Moreno, Ana M. y Vegas, Sira. 2005.** *Técnicas De Evaluación de Software.*
20. **Joseph, L., H. and M. Carl Charles. 2004.** [aut. libro]
21. **López Nájera, Alberto. 2009.** *Fundamentos de Informática para profesionales de la Salud.* La Mancha : s.n., 2009. ISBN: 978-1-4092-6918-2.
22. **Mato García, Rosa María.** Sistemas de Bases de Datos. La Habana: Pueblo y Educación, 2005.
23. **Mendoza Sánchez, María A. 2004.** Metodologías de desarrollo de Software. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html. [En línea] 2004. Citado el: [9 de diciembre de 2013].
24. **Murray, Pablo.** 2003. GESTIÓN-INFORMACIÓN-CONOCIMIENTO. Escuela de Bibliotecarios de la Biblioteca Nacional de Buenos Aires. Buenos Aires, Argentina: s.n., 2003.
    Nistal Freire, José Luis. 2001. [En línea] julio de 2001. [Citado el: 6 de diciembre de 2013]. http://teleformacion.edu.aytolacoruna.es/PASCAL/document/conten.htm.
25. **Peris Soler, Pablo y Belenguer Navarro, Noemí. s/f.** [En línea] s/f. [Citado el: 2 de mayo de 2014]. *http://*ccia.ei.uvigo.es/docencia/SCS/0910/transparencias/Tema4.pdf.
26. **PRUEBAS DE SOFTWARE. 2005.** *Gestión de Calidad y Pruebas de Software.* [En línea]

27. **Reynoso, Billy. 2005.** Introducción a la Arquitectura de Software. [Digital] Buenos Aires : Universidad de Buenos Aire, 2005.
28. **Sandoval, Iris N. 2011.** *http://irissandoval.lacoctelera.net.* [En línea] UNIVERSIDAD NACIONAL EXPERIMENTAL SIMON RODRIGUEZ, 16 de abril de 2011. [Citado el: 9 de noviembre de 2013]. http://irissandoval.lacoctelera.net/post/2011/05/15/introduccion-al-procesamiento-datos.
29. **Schmuller, Joseph. 2000.** *Aprendiendo UML en 24 hr.* México : PEARSON EDUCATION , 2000. ISBN: 968-444-463-X.
30. **Sequeira, Msc. Tania. 2010.** LinkedIn Corporation . *http://www.slideshare.net.* [En línea] 16 de Noviembre de 2010. [Citado el: 9 de Marzo de 2014]. http://www.slideshare.net/NoeGonzalezMendoza/arquitectura-cliente-servidor.
31. **Sevenen Corp. 2012.** *http://sevenencorp.com/.* [En línea] Android Panama, 2012. [Citado el: 7 de marzo de 2014]. http://www.sevenencorp.com/servicios/desarrollo/aplicaciones-de-escritorio.
32. **Teniente López, Ernest; Olivé, Antoni Ramon; Zarroca Mayol, Enric y Gómez Seone, Cristina. 2003.** *Diseño de sistemas software en UML.* s.l. : UPC 2003.
33. **Thelin, Johan. 2007.** *Foundations of Qt Development.* New York : worldwide by Springer- Verlag, 2007. ISBN-13: 978-1-59059-831-3, ISBN-10: 1-59059-831-8.
34. **Troya, José M y Vallecillo, Antonio. 2000.** *Ingeniería del software y bases de datos: tendencias actuales.* 2000.
35. **Vieito Román, Alberto Ramón. 2005.** *Manual de Programación. Lenguaje C++.* España : MAD, S.L, 2005. ISBN: 84-665-4456-9.
36. **W3C. Guía Breve de Servicios Web. 2008.** *[Cited 25 october 2009].* Available from world wide Web. http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

ANEXOS

Anexo 1

Entrevista al líder de proyecto SIG-DESKTOP

Fecha: _____

Nombre del entrevistado: _____

Cargo: _____

Área a la que pertenece: _____

Realizada por: _____

Siendo el día ___del mes __en el año 20 ____se realiza la siguiente entrevista al líder de proyecto SIG-DESKTOP con el objetivo de identificar los requerimientos del sistema, los criterios de estos expertos serán utilizados para el desarrollo del trabajo de diploma Proveedor de datos de los servicios Web UCI para el sistema GeoQ-Guardián.

Preguntas

¿Cuáles son las deficiencias que presenta el sistema GeoQ-Guardián que generan errores en la planificación de la guardia?

¿Por qué es necesario consultar versiones de datos actualizados?

¿Quiénes son los implicados en el proceso de planificación de la guardia?

¿Cómo es el proceso de actualización de datos que está implementado en el sistema?

Desde su punto de vista, explique que considera más importante:

- Obtener versiones de datos actualizadas.
- Minimizar el tiempo de obtención de los datos.

¿Qué rol se le asigna a los implicados en el proceso de planificación de la guardia?

Sobre la base de las deficiencias que presenta el sistema, diga:

- ¿Qué considera usted necesario para la solución que se propone?

Firma del entrevistado _____

Entrevista a los administradores de los servicios Web en la Universidad

Fecha: _____

Nombre del entrevistado: _____

Cargo: _____

Área a la que pertenece: _____

Realizada por: _____

Siendo el día ___ del mes ___en el año 20____se realiza la siguiente entrevista a los administradores de los servicios Web en la Universidad con el objetivo de identificar los requerimientos del sistema, los criterios de estos expertos serán utilizados para el desarrollo del trabajo de diploma Proveedor de datos de los servicios Web UCI para el sistema GeoQ-Guardián.

Preguntas

¿Qué tipo de información están disponible en los servicios Web que se le ofrece a la Universidad?

¿Cómo están distribuidos los servicios Web?

¿Cada qué tiempo se actualizan las versiones de datos?

¿Qué estructura o estándar se tiene en cuenta para publicar los servicios?

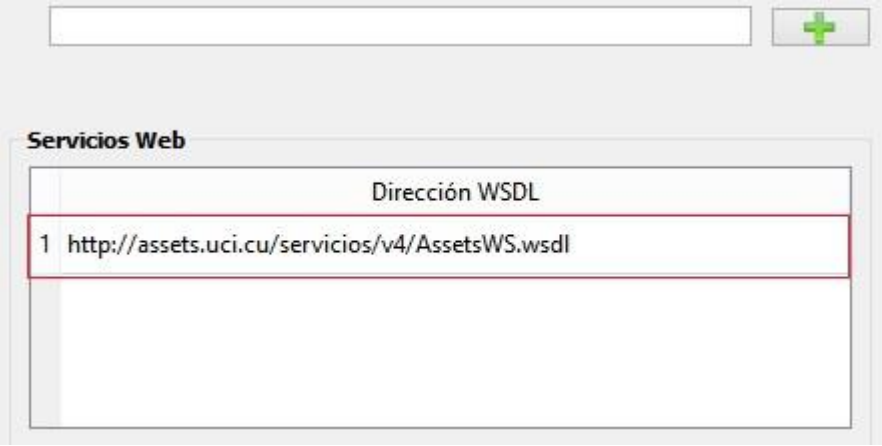
Firma del entrevistado _____

Anexo 2

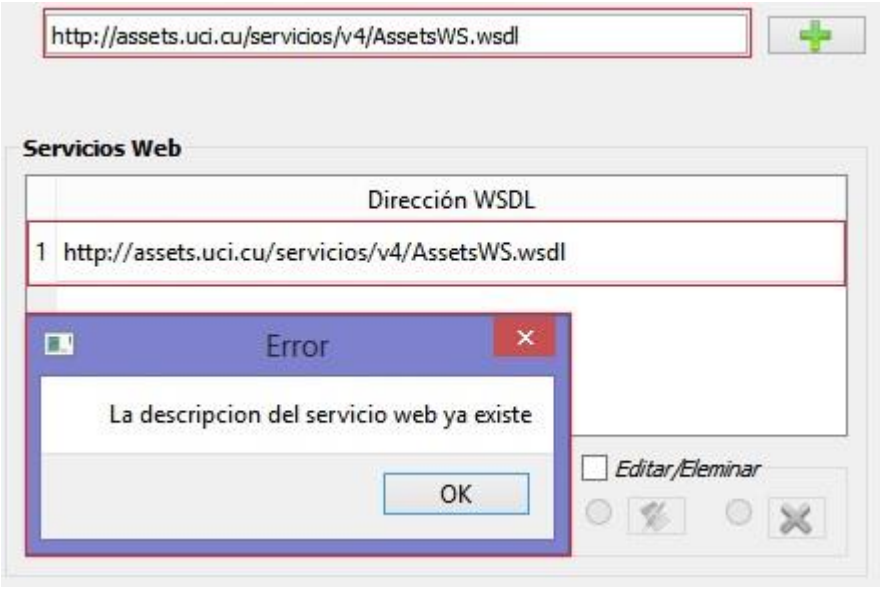
Descripción textual de los casos de uso del sistema

Caso de Uso:	Gestionar dirección
Actores:	Planificador
Resumen:	El caso de uso inicia cuando el planificador va a realizar algunas de las operaciones siguientes: - **Registrar dirección de un servicio Web:** el planificador ingresa la dirección del servicio Web en el formulario y la nueva dirección es registrada. Finalizando el caso de uso. - **Modificar dirección de un servicio Web:** el planificador selecciona la dirección que desea modificar y la modifica. Finalizando así el caso de uso. - **Eliminar dirección de un servicio Web:** el planificador selecciona la dirección que desea eliminar y la elimina. Finalizando así el caso de uso.
Precondiciones:	• Para eliminar o modificar debe existir al menos una dirección de servicio Web registrada en el sistema.
Referencias:	RF5, RF6, RF7
Prioridad:	Crítico

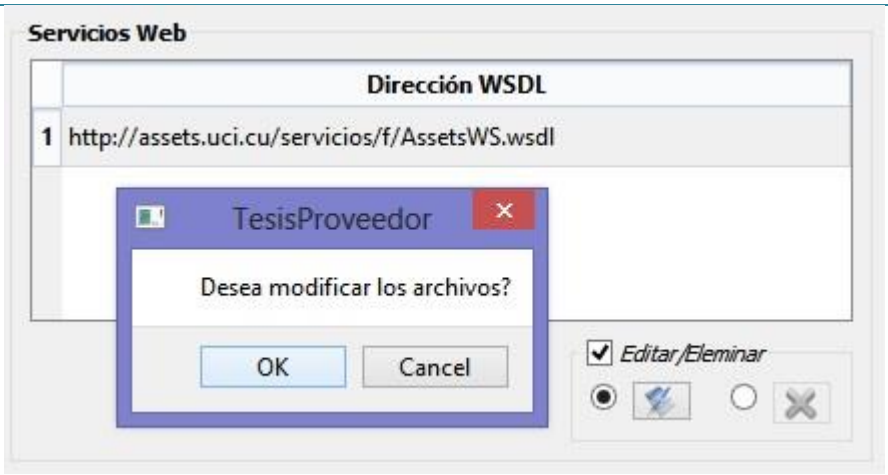
Flujo Normal de Eventos **Sección "Registrar dirección de un servicio Web"**	
Acción del actor	**Respuesta del Sistema**
1.1 El planificador ingresa en el formulario la dirección del servicio Web. 1.2 El planificador acciona el botón adicionar.	1.3 El sistema verifica que la dirección no haya sido adicionada anteriormente. En caso contrario ver flujos alternos 1.3. 1.4 El sistema adiciona la dirección y la muestra en la tabla de direcciones. Finalizando así el caso de uso.
Prototipo de Interfaz	

Prototipo de Interfaz


Flujos Alternos	
Acción del actor	**Respuesta del sistema**
1.5 El actor acciona el botón OK del mensaje mostrado y regresa al paso 1.1.	1.3 Si la dirección ya existe, el sistema muestra un mensaje informándolo.

Prototipo de Interfaz


Flujo Normal de Eventos Sección “Modificar dirección de un servicio Web”	
Acción del actor	**Respuesta del Sistema**
2.1 El planificador selecciona en el sistema la opción *“Editar/Eliminar”.* 2.3 El planificador selecciona la opción editar. 2.4 El planificador selecciona la dirección del servicio Web que desea editar y la edita. 2.5 El planificador acciona el botón asociado a la operación editar. 2.7 El planificador acciona el botón OK del mensaje mostrado. En caso contrario ver flujo alterno 2.7.	2.2 El sistema permite el acceso a las operaciones editar y eliminar. 2.6 El sistema muestra un mensaje para confirmar si desea realizar la operación. 2.8 El sistema verifica que los cambios realizados sobre la dirección sean correctos, en caso contrario ver flujo alterno 2.8, y realiza el cambio. Finalizando así el caso de uso.
Prototipo de Interfaz	
	
Prototipo de Interfaz	



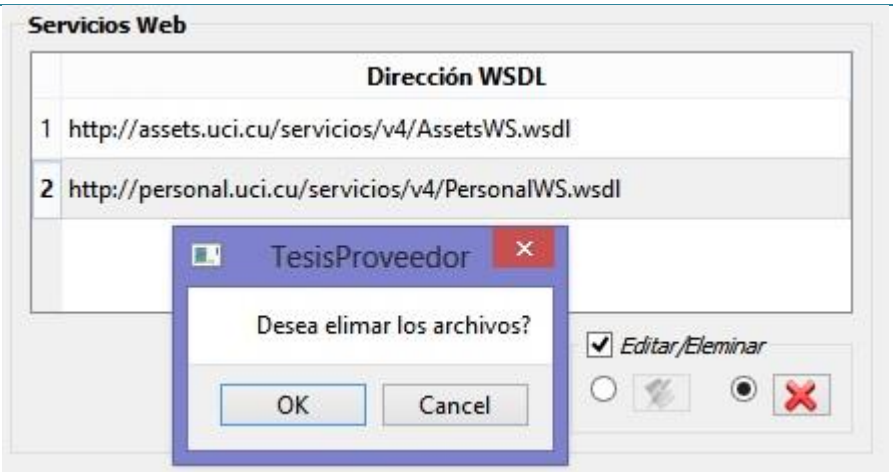
Flujos Alternos	
Acción del actor	**Respuesta del sistema**
2.7 El planificador acciona el botón Cancelar del mensaje mostrado.	2.7.1 El sistema no realiza ninguna operación.
Prototipo de Interfaz	
2.10 El planificador acciona el botón OK del mensaje informativo.	2.8 El sistema detecta que los cambios realizados en la dirección del servicio Web no son correctos. 2.9 El sistema muestra un mensaje informativo. 2.11 El sistema no modifica la dirección seleccionada y regresa al paso 2.4.
Prototipo de Interfaz	

Flujo Normal de Eventos

Sección "Eliminar dirección de un servicio Web"

Acción del actor	Respuesta del Sistema
3.1 El planificador selecciona en el sistema la opción "Editar/Eliminar". 3.3 El planificador selecciona la opción eliminar. 3.4 El planificador selecciona la dirección del servicio Web que desea eliminar. 3.5 El planificador acciona el botón asociado a la operación eliminar. 3.7 El planificador acciona el botón OK del mensaje mostrado. En caso contrario ver flujo alterno 3.7.	3.2 El sistema permite el acceso a las operaciones editar y eliminar. 3.6 El sistema muestra un mensaje para confirmar si desea realizar la operación. 3.8 El sistema elimina la dirección seleccionada. Finalizando así el caso de uso.

Prototipo de Interfaz



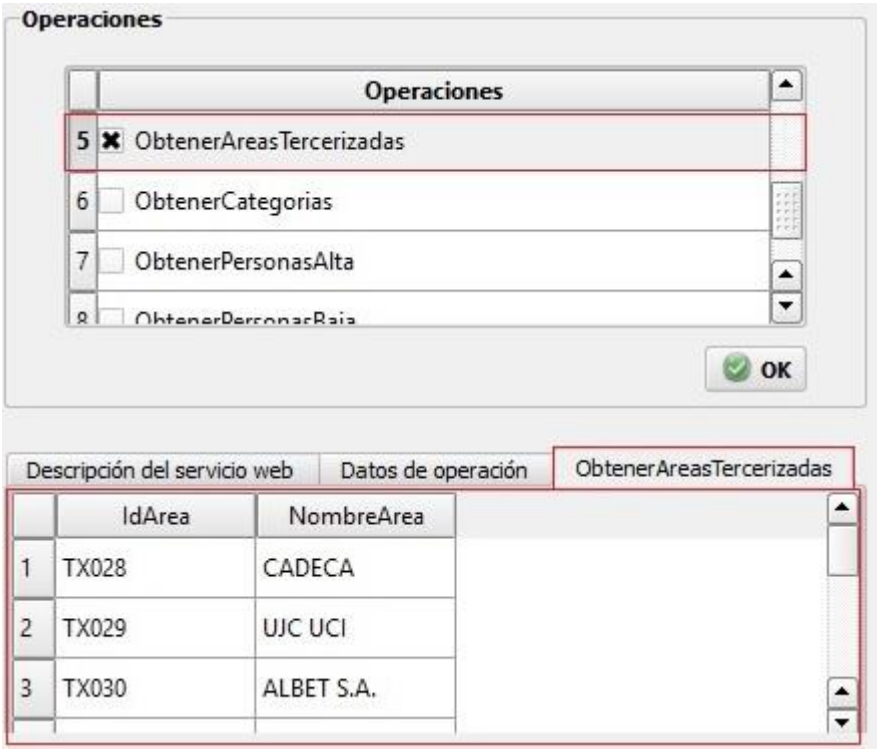
Flujos Alternos	
Acción del actor	**Respuesta del sistema**
3.7 El planificador acciona el botón Cancelar del mensaje mostrado.	3.7.1 El sistema no realiza ninguna operación.

Caso de Uso:	Consultar operación
Actores:	Planificador
Resumen:	El caso de uso comienza cuando el planificador selecciona la operación a consultar y acciona el botón OK. El caso de uso termina cuando el sistema crea capas vectoriales y visualiza en las capas, los datos referentes a la operación seleccionada.
Precondiciones:	• Debe existir al menos una operación registrada en el sistema. • Las operaciones registradas en el sistema deben contener datos.
Referencias:	RF3, RF4
Prioridad:	Crítico

Flujo Normal de Eventos **Sección "Consultar operación de un servicio Web"**	
Acción del actor	**Respuesta del Sistema**
1.1 El actor selecciona la operación del servicio Web que desea	1.3 El sistema ejecuta la petición y visualiza en una tabla los datos correspondientes a las operaciones.

consultar. 1.2 El actor acciona el botón OK.	En caso contrario ver flujos alternos 1.3.

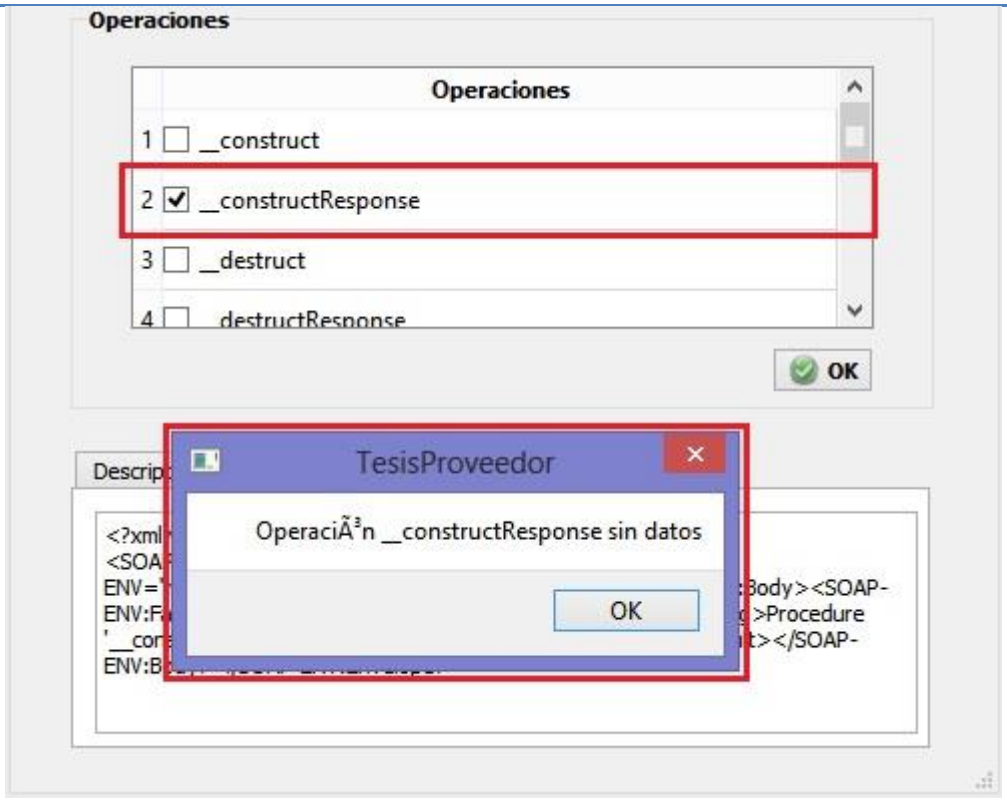
Prototipo de Interfaz



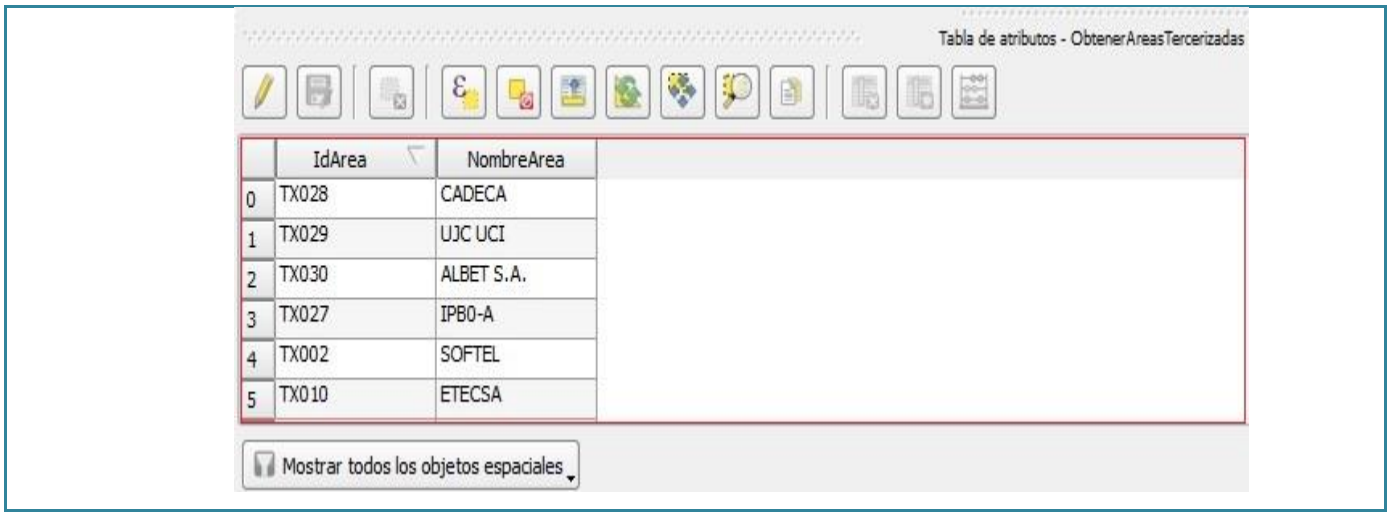
Flujos Alternos

Acción del actor	Respuesta del sistema
1.3.1 El actor acciona el botón OK del mensaje mostrado, finalizando así el caso de uso.	1.3 Si la operación no contiene información, el sistema muestra un mensaje informativo, *"Operación sin datos".*

Prototipo de Interfaz



Flujo Normal de Eventos Sección “Crear capa vectorial”	
Acción del actor	**Respuesta del Sistema**
2.1 Se realizan los pasos 1.1 y 1.2 de la Sección “Consultar operación de un servicio Web”.	2.2 El sistema crea capas vectoriales con los datos asociados a las operaciones consultadas, si no ver flujos alternos. 1.3. Finalizando así el caso de uso.
Prototipo de Interfaz	



Anexo 3

Diagrama de clases del diseño del caso de uso Gestionar dirección

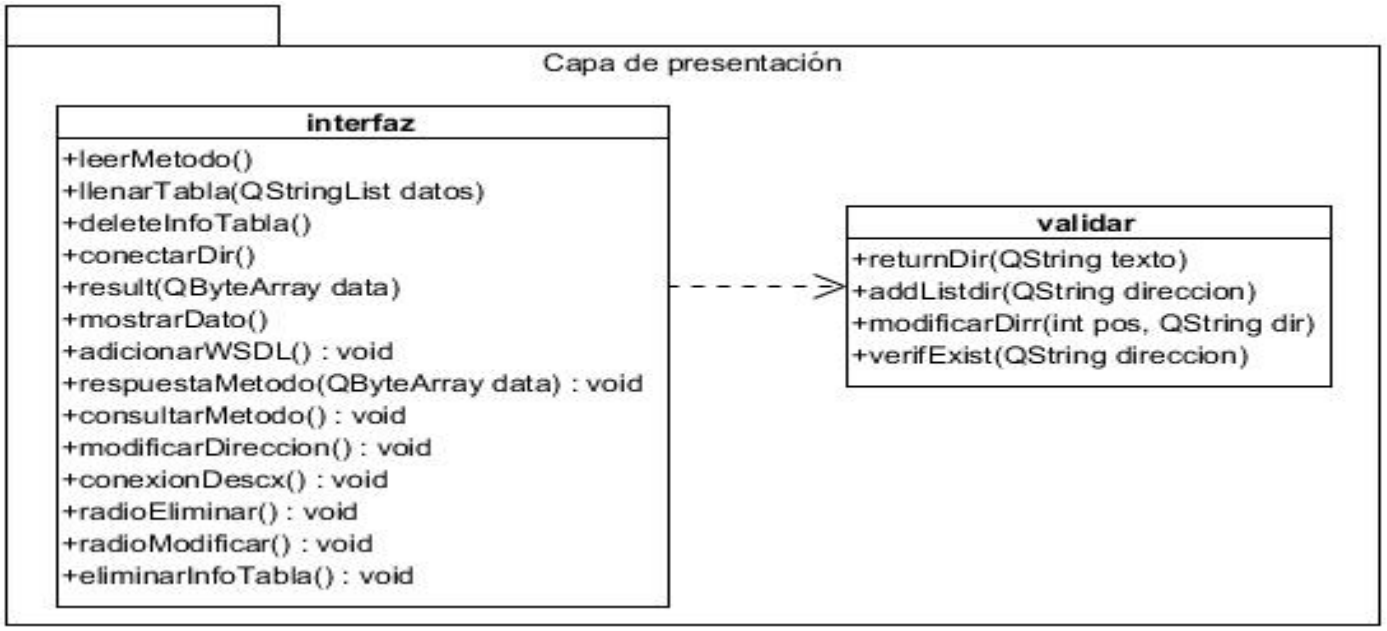
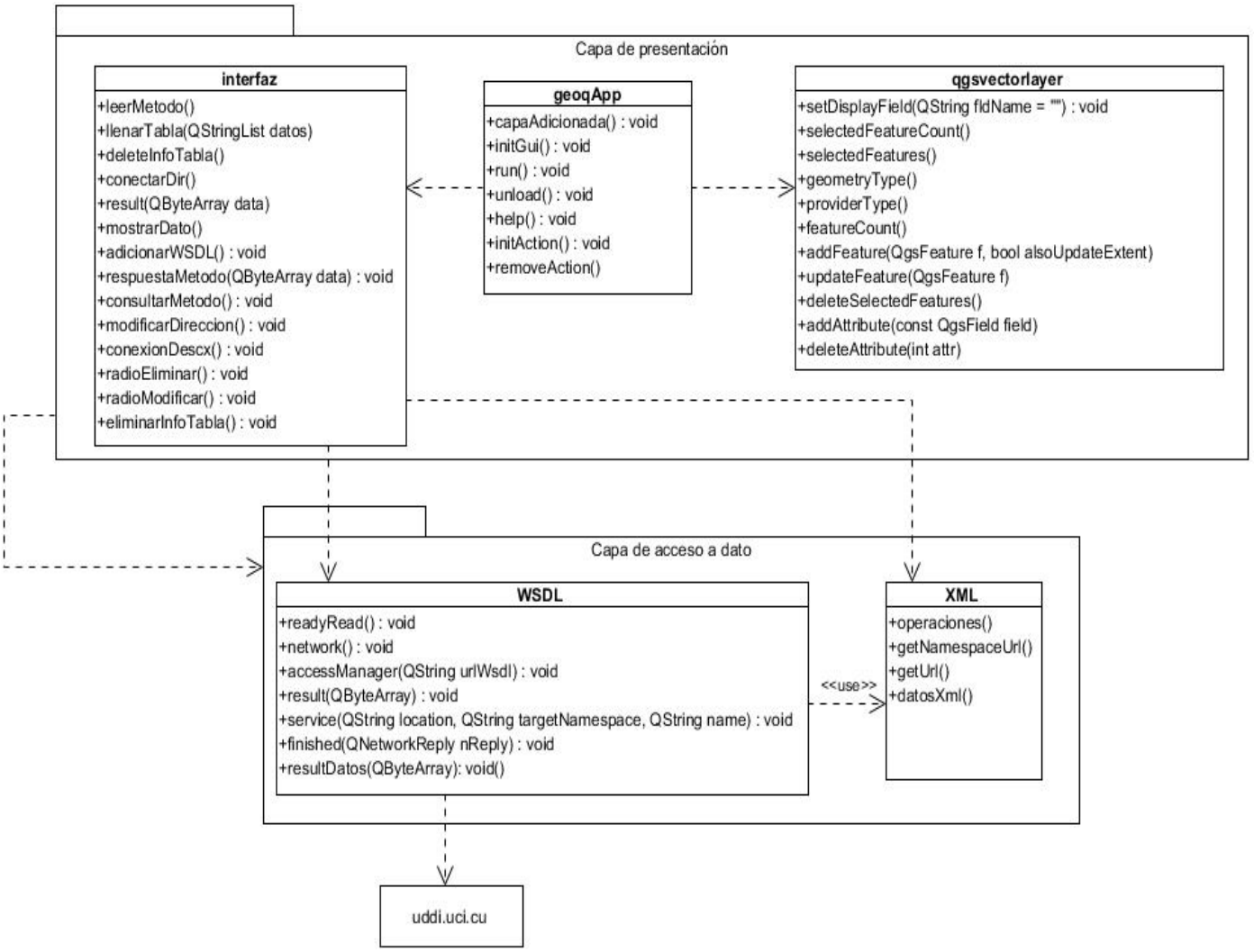


Diagrama de clases del diseño del caso de uso Consultar operación



Anexo 4

Diagrama de componentes del caso de uso Consultar operación

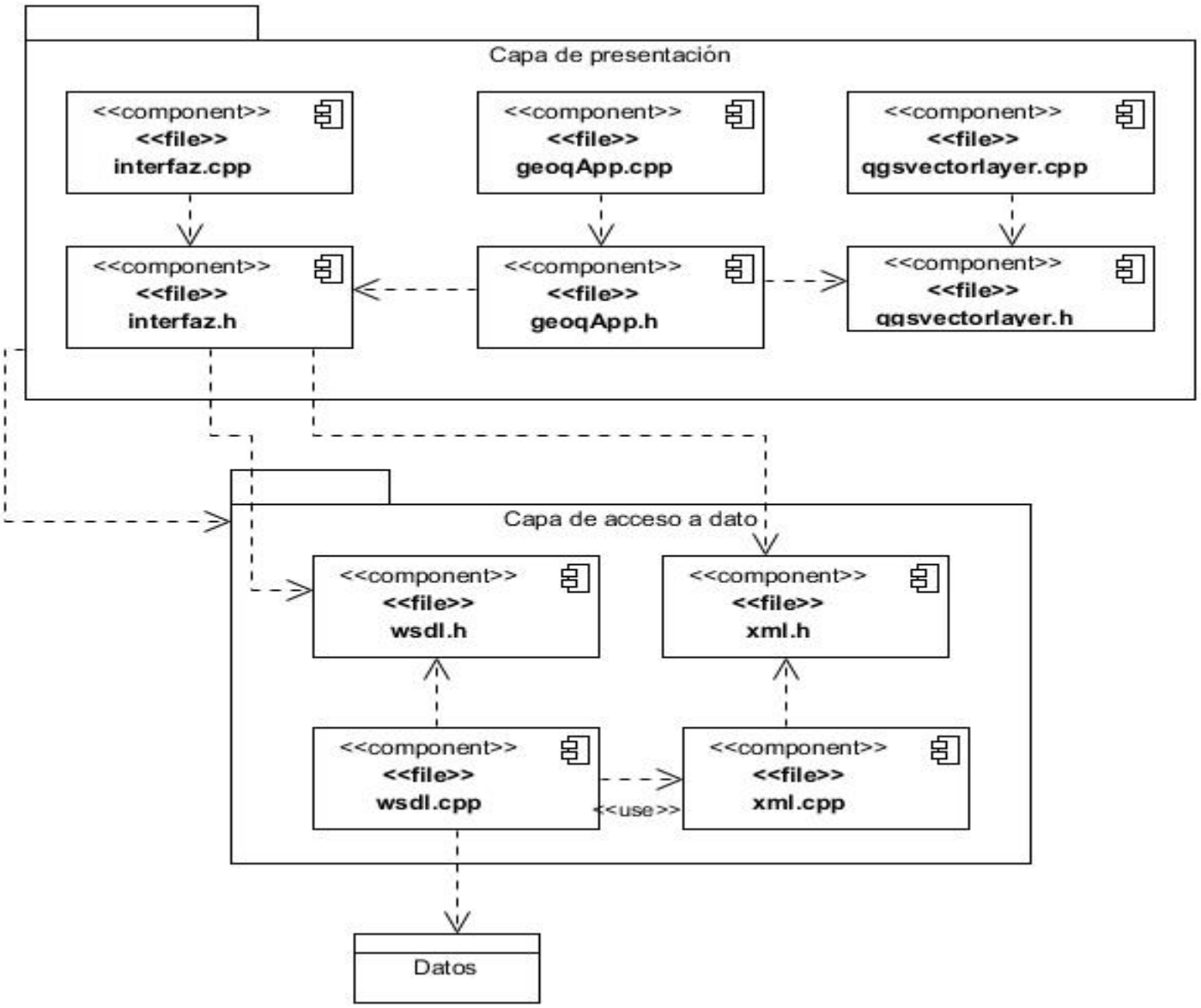
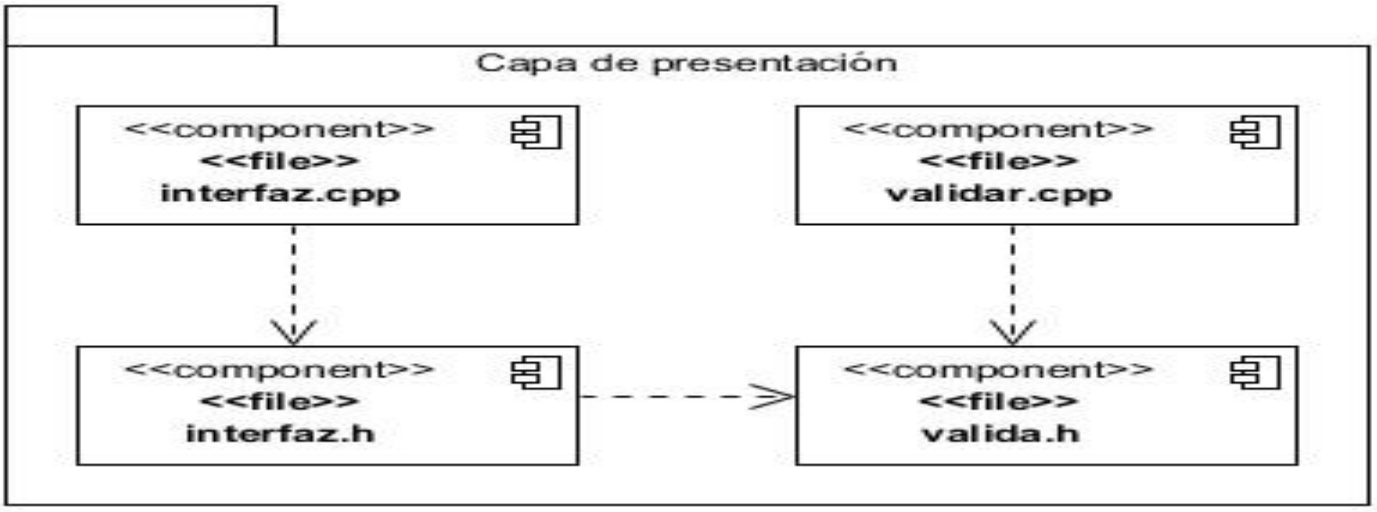


Diagrama de componentes del caso de uso Gestionar dirección



Anexo 5

Secciones a probar en el caso de uso Consultar operación

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: "Consultar operación de un servicio Web"	EC 1.1: El sistema muestra en una tabla los datos de la operación consultada.	El planificador selecciona en la tabla "Operaciones" la operación del servicio web a consultar. El sistema ejecuta la solicitud y visualiza en una tabla los datos asociados a la operación consultada.	Proveedor 4. Ventana principal. 5. El usuario selecciona el checkbox de la operación que desea consultar. 6. El usuario acciona el botón OK. 7. El sistema muestra en una tabla los datos de la operación consultada.
	EC 1.2: La operación consultada no contiene datos.	El planificador selecciona en la tabla "Operaciones" la operación del servicio web a consultar. Si la operación no contiene datos el sistema muestra un mensaje, "Operación sin datos".	Proveedor. 1. Ventana principal. 2. El usuario selecciona el checkbox de la operación que desea consultar. 3. El usuario acciona el botón OK. 4. El sistema muestra un mensaje

			"Operación sin datos". 5. El usuario acciona el botón OK del mensaje.
SC 2: "Crear capa vectorial"	EC 2.1: Se crea la capa vectorial.	El planificador selecciona en la tabla "Operaciones" la operación del servicio web a consultar. El sistema ejecuta la solicitud, si la operación contiene datos, crea una capa vectorial con los datos asociados a la operación consultada.	Proveedor 1. Ventana principal. 2. El usuario selecciona el checkbox de la operación que desea consultar. 3. El usuario acciona el botón OK. 4. El sistema crea una capa vectorial con los datos asociados a la operación consultada.
	EC 2.2: No se crea la capa vectorial.	El planificador selecciona en la tabla "Operaciones" la operación del servicio web a consultar. El sistema ejecuta la solicitud, si la operación no contiene datos, el sistema muestra un mensaje "Operación datos".	Proveedor 1. Ventana principal. 2. El usuario selecciona el checkbox de la operación que desea consultar. 3. El usuario acciona el botón OK. 4. El sistema muestra un mensaje.

			5. El usuario acciona el botón del mensaje.

Descripción de variable del caso de uso consultar operación

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Operaciones	QTableWidget	No	El usuario selecciona en la tabla "Operaciones", la operación a consultar.

Matriz de datos del caso de uso consultar operación

SC 1: "Visualizar descripción de un servicio web"

Id del escenario	Operaciones	Respuesta del sistema	Resultado de la prueba
EC 1.1	V "*ObtenerAreas*"	El sistema visualiza en una tabla los datos asociados a la operación consultada.	Satisfactorio
EC 1.2	V "C*onstruct*"	El sistema muestra un mensaje informando, "Operación sin datos".	Satisfactorio
EC 2.1	V "*ObtenerAreas*"	El sistema crea una capa vectorial con los datos de la operación consultada.	Satisfactorio
EC 2.2	V "C*onstruct*"	El sistema muestra un mensaje informando, "Operación sin datos".	Satisfactorio