Universidad de las Ciencias Informáticas

Facultad 6

“Capa de Servicios Web para el Sistema Integrado de Gestión Estadística (SIGE)”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

José Luis García Companioni

Yasel Noda García

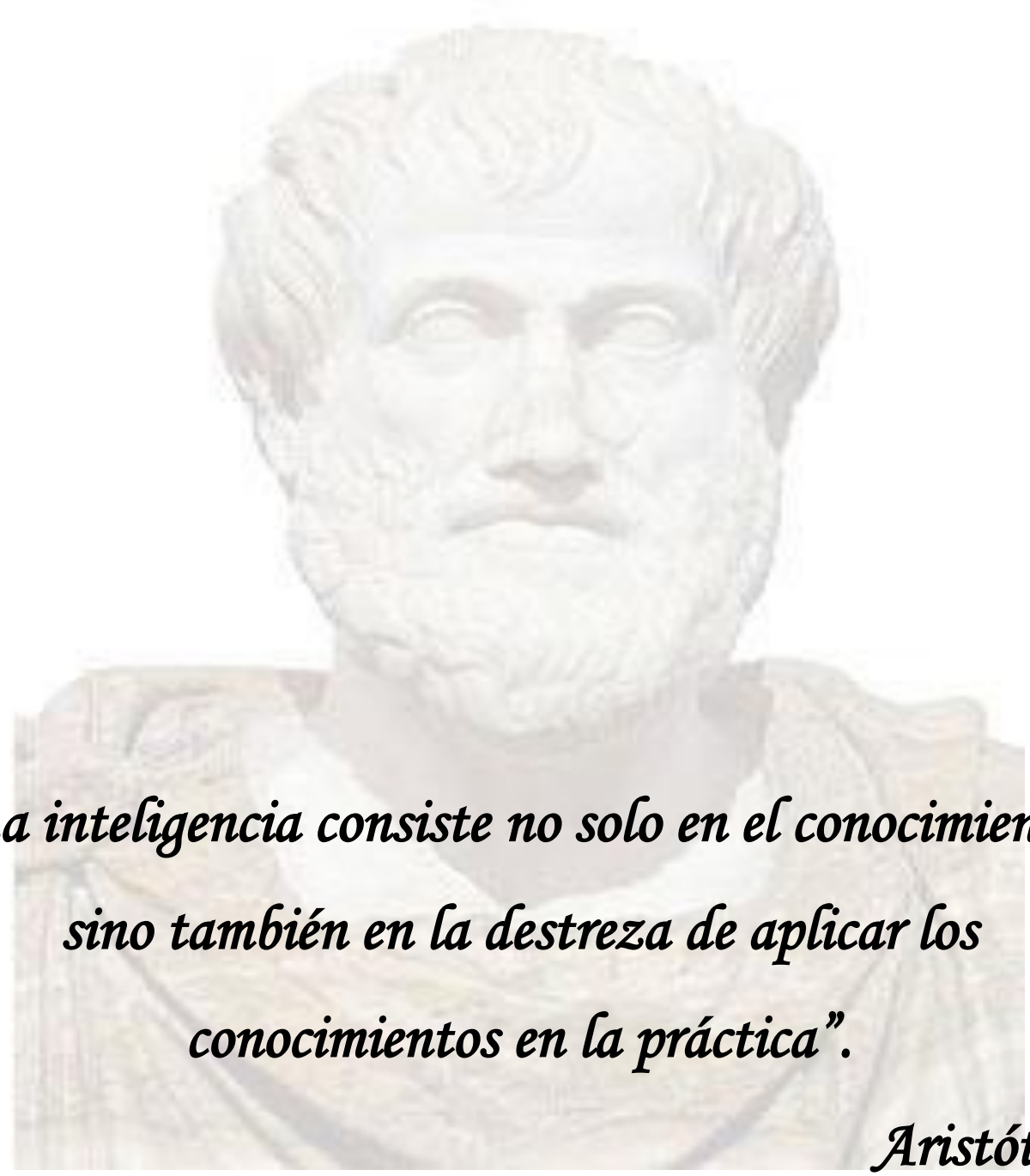
Tutores:

Ing. Yuned Rivero Guerra

Ing. Alejandro González Sánchez

La Habana, junio de 2015

“Año 57 de la Revolución”



"La inteligencia consiste no solo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica".

Aristóteles

Declaración de autoría

Declaramos ser autores del Trabajo de Diploma: “Capa de Servicios Web para el Sistema Integrado de Gestión Estadística” y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que conste firmamos la presente a los ____ días del mes de _____ de 2015.

_____
José Luis García Companioni
Autor

_____
Yasel Noda García
Autor

_____
Ing. Yuned Rivero Guerra
Tutora

_____
Ing. Alejandro González Sánchez
Tutor

Datos de Contacto

Ing. Yuned Rivero Guerra

Centro de trabajo: Centro de Tecnologías de Gestión de Datos. Universidad de las Ciencias Informáticas.

Especialidad de graduación: Ingeniera en Ciencias Informáticas.

Correo electrónico: yrivero@uci.cu

Ing. Alejandro González Sánchez

Centro de trabajo: Centro de Tecnologías de Gestión de Datos. Universidad de las Ciencias Informáticas.

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Correo electrónico: alejandrogs@uci.cu

Agradecimientos de José Luis

Quisiera empezar agradeciendo a las dos personas más importantes para mí, que son mi mamá y mi abuela, las cuales han hecho el papel de madre y padre a la vez y me han sabido formar con grandes valores humanos y una buena educación, pues para mí no cabe la menor duda de que sin su ayuda y sus consejos no hubiera llegado hasta aquí, ya que son el significado de mi vida. Mi mamá siempre me dice que le agradece a Dios por darle un hijo tan bueno como yo, pero yo le agradezco mucho más por haberme dado esa mamá y esa abuela, pues yo creo que me dio más de lo que merezco. También quisiera agradecer a mi bisabuela Teresa y a mi abuela Blanca, por siempre haber tenido paciencia para educarme y cuidarme cada día que necesité de ellas. No puede faltar el agradecimiento a mi tío Rosendo, que es como un hermano para mí. Agradecer a mi tía Turni por brindarme tanto cariño, a Zonia y Julio, a los cuales les tengo mucho aprecio. Le agradezco a mi novia Alina por soportarme estos últimos 3 años y haberme dado todo el cariño y el apoyo que he necesitado en los buenos y malos momentos. La realidad es que existen muchos familiares, amigos y compañeros a los que quisiera agradecer pero no puedo mencionarlos a todos, sin embargo les agradezco de corazón por todo su apoyo en los momentos que hemos podido compartir. Además agradecer a todos los profesores que han formado parte de mi aprendizaje como estudiante a lo largo de mi vida, muchos de ellos grandes educadores con los cuales he tenido el privilegio de nutrir mis conocimientos y desarrollarme para cumplir mi objetivo y el de mi mamá, que ha sido graduarme de una carrera universitaria. Agradecer a mi dúo de tesis, el cual tuve la oportunidad de elegir y como ya lo conocía desde el politécnico, sabía la calidad humana que posee y no me hubiera imaginado a otra persona con la cual trabajar en mi tesis de diploma que no fuera él. No por ser los últimos son los menos importantes, pues ellos se merecen un agradecimiento especial, me refiero a nuestros tutores Yuned y Alejandro, que nos han sabido guiar de manera excepcional, a pesar de los problemas que se han presentado a lo largo del desarrollo de este trabajo de diploma.

Dedicatoria de José Luis

Este trabajo de diploma está dedicado a:

- Maritza Companioni Perera.
- Mireya Perera Pérez.

Agradecimientos de Yasel

Agradezco de manera especial a Dios, porque nunca me abandonó cuando más lo necesitaba, gracias a mi mamá Yaneisy por las fuerzas y el ánimo que día a día me daba, gracias mamá por ser mi madre y mi padre, por el esfuerzo y el sacrificio. Gracias a mi tío Héctor, a mis abuelos, a mis hermanos, gracias a todos mis tíos y primos que me ayudaron y me apoyaron. Agradezco a todas y todos los del grupo 6505 que son especiales y amigos y a mis profesores que me formaron, enseñaron y educaron y a Alina y a mis hermanos de la fe. Gracias a Pepe por el esfuerzo, la dedicación y la confianza, también a mis tutores Yuned y Alejandro.

Dedicatoria de Yasel

Dedico el Trabajo de Diploma a:

- Yaneisy García Pacheco, mi mamá.
- Héctor García Pacheco, mi tío.
- Casilda y a José, mis abuelos.
- Yireidi, Yariel y Héctor Luis, mis hermanos.

Resumen

El Centro de Tecnologías de Gestión de Datos (DATEC) enmarcado en la Universidad de las Ciencias Informáticas (UCI), desarrolló el Sistema Integrado de Gestión Estadística (SIGE) el cual realiza un conjunto de procedimientos y acciones, tales como: guiar el proceso de diseño y captura de formularios en un organismo, controlar la seguridad de la información, permitir la visualización de datos a directivos a través de reportes y permitir la gestión de encuestas. SIGE tiene como objetivo llevar a cabo el proceso de automatización de los procesos de gestión estadística en la ONEI y en algunos de los organismos asociados a la misma. Para estos organismos utilizar algún proceso de SIGE tienen que contar con todo el sistema aunque no lo necesiten en su totalidad, además, la solicitud y envío de la información es un proceso que puede demorar hasta varios días. El objetivo del presente trabajo de diploma consiste en desarrollar una Capa de Servicios Web para SIGE que garantice a los organismos asociados a la ONEI utilizar algunos de sus procesos sin requerir la totalidad del sistema, agilizándose también el proceso de solicitud y envío de la información. Para guiar el proceso de desarrollo de la Capa de Servicios se caracteriza la metodología, herramientas y tecnologías empleadas. Además, se diseñan e implementan las clases de la Capa de Servicios Web aplicando buenas prácticas de los patrones de diseño. Se realizaron pruebas funcionales, de seguridad y de aceptación para probar el correcto funcionamiento de la Capa de Servicios Web.

Palabras claves: estadística, proceso, servicio web, Sistema Integrado de Gestión Estadística.

Abstract

The Data Management Technology Centre (DATEC) from the University of Informatics Sciences (UCI), developed the Statistics Management Integrated System (SIGE) which performs a set of procedures and actions such as guiding the design process and capturing forms in an organism, it also controls the information security, allows data visualization to executives through reports as well as managing surveys. SIGE aims to carry out the process of automation of statistics management processes in the ONEI and some of the organisms associated with it. For these organizations to use some SIGE process they must have all the system even if they do not need it complete, also the application and submission of information is a process that can take up to several days. The objective of this paper is to develop a Web services layer for SIGE that guarantees to the agencies associated to ONEI use some of their processes without requiring the entire system, speeding up at the same time the process of application and submission of information. To guide the development process of the service layer, the methodology, tools and technologies used are characterized. In addition, the classes of the Service Layer are designed and implemented applying best practices of design patterns. Functional, security and acceptance tests were performed to test the proper functioning of the Web Services Layer.

Keywords: process, statistics, Statistics Management Integrated System, web service.

Índice

Índice de figuras

Índice de tablas

Introducción

Actualmente las Tecnologías de la Información y las Comunicaciones (TICs) constituyen un factor decisivo en el desarrollo económico y productivo de los países. El desarrollo del potencial de las nuevas generaciones depende del aprovechamiento que puedan hacer del uso de las TICs. La gestión estadística es una tarea que las empresas y los gobiernos a nivel mundial tienen en consideración por la importancia y valor que representa para minimizar los gastos y maximizar las ganancias. Han sido numerosos los sistemas informáticos que se han realizado con el objetivo del análisis estadístico, los cuales están encaminados a registrar datos y devolver reportes que los directivos pueden utilizar para tomar decisiones ya sea a nivel de gobierno o de empresa.

Las empresas cubanas no están ajenas de la gestión estadística, esta es una tarea que se atiende con suma importancia, pues constituye uno de los pilares del sistema económico cubano. En Cuba, la Oficina Nacional de Estadística e Información (ONEI) constituye el órgano rector en lo que a gestión estadística se refiere, pues es la encargada de analizar estadísticamente toda la información relevante que los otros organismos le proporcionan para así conformar reportes y enviar informes al gobierno. El Centro de Tecnologías de Gestión de Datos (DATEC), enmarcado en la Universidad de las Ciencias Informáticas (UCI), desarrolló el Sistema Integrado de Gestión Estadística (SIGE) con el objetivo de gestionar la información estadística de entidades y organismos. Junto a las exigencias de la ONEI el sistema se fue perfeccionando hasta el punto que no solo es capaz de guiar el proceso de diseño y captura de formularios en un organismo sino que también controla la seguridad de la información, permite la visualización de datos a directivos a través de reportes así como la gestión de encuestas. Hoy en día el número de clientes ha aumentado desde su inicio y continúa creciendo en la actualidad, ejemplo de ellos son los siguientes organismos: Fiscalía General de la República, Tribunal Supremo Popular, Unión de Empresas Recuperadoras de Materias Primas e incluso la propia universidad.

Actualmente cuando la ONEI solicita una información a un organismo se realiza a través de una plantilla específica atendiendo al tipo de información solicitada. Para poder enviarse y recibirse la plantilla con la información solicitada los organismos tardan hasta varios días por lo que se hace lenta la solicitud y envío de la información. Además, cuando un organismo quiere hacer uso de algún proceso de SIGE tiene necesariamente que tener el sistema en su totalidad, no siendo factible porque si dicha entidad cuenta con

desarrolladores y no desea o no necesita todo el sistema no tiene forma de acceder a los procesos que gestiona.

De la problemática anterior surge como **problema de investigación**: ¿cómo extender procesos de SIGE a terceros sin requerir la totalidad del sistema?

Objetivo General: desarrollar una Capa de Servicios Web para el Sistema Integrado de Gestión Estadística que permita hacer uso de algunos procesos por terceros.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- Caracterizar los enfoques teórico-conceptuales referentes a los Servicios Web.
- Diseñar los Servicios Web necesarios para la creación de la Capa de Servicios para el Sistema Integrado de Gestión Estadística.
- Implementar los Servicios Web.
- Probar los Servicios Web implementados.

Por esta razón el **objeto de estudio** es el proceso de desarrollo de Servicios Web enmarcado en el **campo de acción:** proceso de desarrollo de Servicios Web para el Sistema Integrado de Gestión Estadística.

Preguntas científicas:

- ¿Cuáles son los fundamentos teóricos que se necesitan para lograr que la Capa de Servicios Web sobre SIGE cumpla los objetivos propuestos?
- ¿Cuáles son los procesos de SIGE que deben ser seleccionados para la confección de la Capa de Servicios?
- ¿De qué manera se implementará la Capa de Servicios que permita extender los procesos de SIGE a terceros?
- ¿Permite realmente la Capa de Servicios que los procesos de SIGE puedan ser usados por terceros sin que estos requieran la totalidad del sistema?

Con el fin de resolver el problema de la investigación y darle cumplimiento a los objetivos planteados se trazaron las siguientes **tareas de investigación:**

1. Análisis de los conceptos básicos y técnicos implicados en el desarrollo de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.
2. Identificación de los requisitos funcionales y no funcionales de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.
3. Confección del diagrama de casos de uso del sistema de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.
4. Elaboración del modelo de diseño de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.
5. Elaboración del modelo de implementación de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.
6. Implementación de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.
7. Diseño de los casos de prueba a partir de los requisitos del sistema para llevar a cabo la realización de las pruebas.
8. Realización de pruebas funcionales, de seguridad y de aceptación para probar el correcto funcionamiento de la Capa de Servicios Web.

Tipo de estudio

Descriptivo: su principal objetivo es describir el fenómeno y reflejar lo esencial y más significativo del mismo, sin tener en cuenta las causas que lo originan, para lo que es necesario captar sus relaciones internas y regularidades, así como aquellos aspectos donde se revela lo general. En este tipo de investigación es de principal importancia la profundidad teórica del planteamiento investigativo, pues ayuda a comprender el valor científico de los resultados obtenidos. La descripción a realizar en estas investigaciones puede asumir el objeto en estado de reposo o en movimiento y la información que se quiere obtener tiene que ser revelada por el investigador, pues se encuentra implícita en el objeto de investigación.

Este tipo de estudio se utiliza porque se realiza una exhaustiva investigación del problema en cuestión, reflejando lo más importante, lo esencial del mismo y se describen de acuerdo a la investigación los conceptos asociados.

Métodos para la búsqueda y procesamiento de la información

Métodos teóricos

Analítico sintético: se hace uso de este método para la identificación de conceptos empleados dentro del campo de los Servicios Web, analizando documentos para la extracción de los elementos más importantes sobre el tema en cuestión.

Modelación: se utiliza este método para la realización de los modelos que intervienen en el desarrollo de la Capa de Servicios Web para SIGE de acuerdo a las características específicas de la aplicación.

Métodos empíricos

Análisis documental: este método es usado para analizar todos los documentos y conceptos que sustentan la investigación y así seleccionar los más adecuados para la misma.

Entrevista: este método se utiliza para recoger la información tanto primaria como secundaria que tributa al desarrollo de la Capa de Servicios para el Sistema Integrado de Gestión Estadística. Es una entrevista planificada y estructurada. Se le realiza al ingeniero: Héctor Luis Reyes Zaldívar, jefe de proyecto de SIGE y se persiguen los siguientes objetivos:

- Especificar el problema de investigación y alcance de la Capa de Servicios Web para SIGE.
- Definir el objetivo general de la Capa de Servicios web para SIGE.
- Identificar los procesos de SIGE que van a ser llevados a la Capa de Servicios Web.
- Determinar los requisitos funcionales.
- Identificar los Casos de Uso.

El presente documento está estructurado en tres capítulos:

Capítulo 1: Fundamentación Teórica: en este capítulo se abordan los principales conceptos asociados a la investigación, se describen las principales herramientas empleadas. Se realiza un estudio de las metodologías y se selecciona una para guiar el desarrollo de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.

Capítulo 2: Análisis y diseño de la solución: en este capítulo se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr que el sistema funcione correctamente. Se describe el análisis y diseño de las clases del sistema. Se construyen los artefactos correspondientes al análisis y diseño, modelando la estructura de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.

Capítulo 3: Implementación y prueba: en este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado. Se describen las pruebas a realizar, con el objetivo de probar el correcto funcionamiento de los objetivos de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.

Capítulo 1: Fundamentación teórica

Introducción

En el presente capítulo se abordarán, los elementos principales que justifican y soportan teóricamente la solución propuesta. Además se presentarán las tecnologías, herramientas y metodología que serán empleadas en el desarrollo de la Capa de Servicios Web.

1.1. Conceptos asociados a la capa de Servicios Web

Para el desarrollo de la solución se tuvieron en cuenta una serie de conceptos importantes tales como: Servicio Web, orientación a servicios, servicio web basado en SOAP y en REST, de los cuales las definiciones y acrónimos se muestra a continuación.

1.1.1. Servicio Web

La World Wide Web Consortium lo define como “un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red” (W3C Consortium, 2004).

La w3c.es dice: “Existen múltiples definiciones sobre lo que son los Servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una posible sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar” (w3c.es, 2014).

IBM define servicio web como el término que designa una tecnología que permite que las aplicaciones se comuniquen en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder

por la red a través de mensajería XML estandarizada. Usa protocolos basados en el lenguaje XML con el objetivo de describir una operación para ejecutar o datos para intercambiar con otro servicio web. Un grupo de servicios web que interactúa de esa forma define la aplicación de un servicio web específico en una arquitectura orientada a servicios (SOA) (IBM, 2014).

A partir del análisis de los conceptos de Servicio Web anteriormente expuestos se decide utilizar el brindado por IBM siendo uno de los más completos porque además de brindar las ventajas explica de manera entendible al usuario que un servicio web se caracteriza por ser una tecnología y una interfaz de software que describe un conjunto de operaciones.

1.1.2. Orientación a servicios

Según Thomas Erl los principios que rigen la orientación a servicios son (Erl, 2004):

- Los Servicios deben ser reusables: todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación.
- Los Servicios deben tener bajo acoplamiento: los Servicios tienen que ser independientes los unos de los otros.
- Los Servicios deben permitir la composición: todo Servicio debe ser construido de tal manera que pueda ser utilizado para construir Servicios genéricos de más alto nivel, el cual estará compuesto de Servicios de más bajo nivel.
- Los Servicios deben de ser autónomos: todo Servicio debe tener su propio entorno de ejecución.
- Los Servicios deben poder ser descubiertos: todo Servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de Servicios que proporcionen las mismas funcionalidades.

1.1.3.Comparación entre los estilos REST y SOAP para el desarrollo de la Capa de Servicios

SOAP (por sus siglas en inglés Simple Object Access Protocol): proporciona el marco por el cual la información específica de la aplicación puede ser transmitida de una manera extensible. Proporciona una descripción completa de las acciones necesarias tomadas por un nodo SOAP al recibir un mensaje SOAP. Es un protocolo basado en XML que consta de:

- Un marco para describir el contenido del mensaje y las instrucciones de proceso.
- Un conjunto opcional de reglas de codificación para representar tipos de datos definidos.
- Una convención para representar llamadas a procedimientos remotos y respuestas (Putte et al. 2004).

REST (por sus siglas en inglés Representational State Transfer) Exequiel Catalani define a REST de la siguiente forma: REST o lo que es lo mismo "Transferencia de estado representacional" es una técnica de arquitectura de software y tiene sus inicios por el año 2000, basado en una tesis doctoral escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP. REST es un estilo de arquitectura para el diseño de aplicaciones en red. La idea es que, en lugar de utilizar los mecanismos complejos, tales como RPC o SOAP para la conexión entre máquinas, se utiliza HTTP para hacer llamadas entre las máquinas (Catalani, 2012).

A continuación se muestra una tabla con una comparación entre los estilos REST y SOAP teniendo en cuenta algunos elementos de interés para escoger el estilo que se utilizará para el desarrollo de la Capa de Servicios Web (Navarro, 2010).

Tabla 1: comparación entre REST y SOAP

	REST	**SOAP**
Características principales	Las operaciones se definen en los mensajes.	Las operaciones son definidas como puertos WSDL.
	Una dirección única para cada instancia del proceso.	Dirección única para todas las operaciones.
	Cada objeto soporta las operaciones estándares definidas.	Múltiples instancias del proceso comparten la misma operación.
	Componentes débilmente acoplados.	Componentes fuertemente acoplados.
Tecnología	Pocas operaciones con muchos recursos.	Muchas operaciones con pocos recursos.
	Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia.	Se centra en el diseño de aplicaciones distribuidas.
	Confía en documentos orientados al usuario	WSDL (Web Services Description

servicio	que define las direcciones de petición y las respuestas.	Language).
	Interactuar con el servicio supone horas de testeado y depuración de URIs.	Se pueden construir automáticamente stubs (clientes) por medio del WSDL.
	WADL propuesto en noviembre de 2006.	WSDL 2.0.
Seguridad	HTTPS.	WS-Security.
Metodología de diseño	Identificar recursos a ser expuestos como servicios.	Listar las operaciones del servicio en el documento WSDL.
	Definir URLs para direccionar los servicios.	Definir un modelo de datos para el contenido de los mensajes.
	Distinguir los recursos de solo lectura (GET) de los modificables (POST, PUT, DELETE).	Elegir un protocolo de transporte apropiado y definir las correspondientes políticas de seguridad.
	Implementar e implantar el servidor Web.	Implementar e implantar el contenedor del Servicio Web.

De acuerdo a la bibliografía consultada se escoge para realizar la Capa de Servicios Web para SIGE a REST debido a las siguientes características expuestas por Rafael Navarro Marset:

- Escalabilidad de la interacción con los componentes: la Web ha crecido exponencialmente sin degradar su rendimiento. Una prueba de ellos es la variedad de clientes que pueden acceder a través de la Web.
- Generalidad de interfaces: gracias al protocolo HTTP, cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial. Esto no es del todo cierto para otras alternativas, como SOAP para los Servicios Web.

A continuación se muestra una imagen con la estructura general del funcionamiento de un servicio REST:

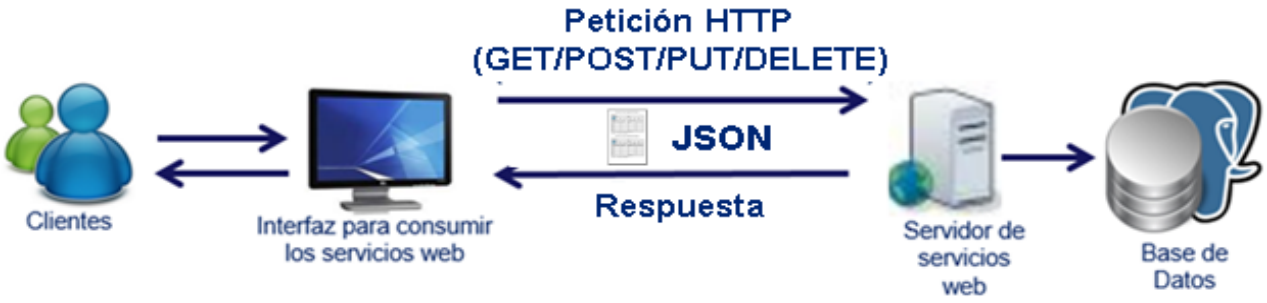


Figura 1: estructura de los Servicios Web basados en REST.

En la figura anterior se observa el funcionamiento una vez implementado y desplegado un servicio REST, el cliente se conecta a una interfaz visual para consumir los servicios web, esta a su vez se encarga de las respuestas y peticiones realizadas por el cliente. La petición se realiza usando HTTP y los métodos GET para obtener, DELETE para eliminar, PUT para editar y POST para insertar. Las respuestas son dadas mediante un JSON por cada petición que se realice. La interfaz recibe y envía las peticiones al servidor donde se encuentran los servicios web y este a su vez obtiene y actualiza información de la base de datos.

1.2. Metodología de desarrollo de software

Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales (Medina, 2015). En el desarrollo de software es importante la selección de la metodología adecuada para el éxito del producto. Para ello se destacan dos enfoques principales: las metodologías tradicionales y metodologías ágiles.

1.2.1. Metodologías tradicionales

Existen las metodologías tradicionales que se modificaron para poder aplicarlas al desarrollo de software, aunque durante mucho tiempo fueron la única solución al desarrollo. Son poco flexibles y muy cuadriculadas. Consistían en una serie de fundamentos y conceptos aplicados al desarrollo de software, documentación, planificación y procesos (Darío, 2014).

1.2.2. Metodologías ágiles

Ante las dificultades de las metodologías tradicionales referentes al tiempo y flexibilidad, aparecen las metodologías ágiles como una respuesta metodológica, especialmente porque están orientadas a proyectos pequeños, constituyen una solución a la medida del entorno, simplificando las prácticas y asegurando la calidad del producto (Darío, 2014).

1.2.3. Comparación entre las metodologías tradicionales y ágiles

A continuación se muestra una comparación entre las metodologías tradicionales y ágiles (Letelier y Penadés, 2006).

Tabla 2: tabla comparativa entre las metodologías tradicionales y las ágiles.

Metodologías tradicionales	Metodologías ágiles
Más Roles y más especificaciones.	Pocos Roles, más genéricos y flexibles.
Más Artefactos.	Pocos Artefactos.
La arquitectura es esencial: se promueve que la arquitectura se defina tempranamente en el proyecto.	Menos énfasis en la arquitectura: la arquitectura se va definiendo y mejorando a lo largo del proyecto.
Cierta resistencia a los cambios (estos pueden provocar grandes costos).	Sujeta a cambios en cualquier etapa del proyecto.
Dirigidas al proceso: roles, actividades y artefactos.	Dirigidas a las personas: el individuo y el trabajo en equipo.
Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas en proyectos grandes y con equipos posiblemente dispersos.	Pueden ser aplicables a proyectos de cualquier tamaño, aunque generalmente su enfoque va dirigido a proyectos pequeños, corta duración, equipos pequeños donde todo el equipo de desarrollo trabaja en el mismo lugar.
Proceso mucho más controlado, con numerosas políticas o normas.	Proceso menos controlado, con pocos principios.

Teniendo en cuenta las particularidades presentes en cada una de estas metodologías y atendiendo a las necesidades y condiciones de la Capa de Servicios Web para SIGE que se desea implementar tales como la inclinación de la misma a obtener un resultado funcional y no una documentación detallada, por el período de tiempo con que se dispone y por la cantidad de miembros en el equipo se decide utilizar una

metodología ágil para guiar el proceso de desarrollo de la Capa de Servicios Web. A continuación se muestran las características de algunas metodológicas ágiles con el objetivo de seleccionar la que se adapte a las características de la Capa de Servicios Web.

OpenUp (por sus siglas en inglés Open Unified Process)

OpenUP es un proceso unificado que aplica iterativos e incrementales planteamientos dentro de un ciclo de vida estructurado. OpenUP abraza una filosofía pragmática y ágil que se centra en la naturaleza colaborativa de desarrollo de software. Es un proceso independiente de herramientas que puede ser extendido para abordar una amplia variedad de tipos de proyecto.

OpenUP estructura el ciclo de vida del proyecto en cuatro fases: concepción, elaboración, construcción y transición. El ciclo de vida del proyecto ofrece a las partes interesadas y a los miembros del equipo visibilidad y puntos decisivos durante todo el proyecto. Esto permite una supervisión eficaz y tomar decisiones en el momento oportuno (eclipse.org, 2012).

A continuación se muestran las fases de la metodología ágil OpenUp:

- Fase de concepción: el proyecto parte de la premisa de que el modelo de negocio ha sido creado, el director del proyecto ha sido identificado, el equipo (al menos para la primera iteración) se ha definido, el entorno de desarrollo (incluyendo las herramientas y la infraestructura) en su sitio.
- Fase de elaboración: la mayoría de las actividades durante una iteración en la fase de elaboración suceden en paralelo. En esencia, los principales objetivos de la elaboración están relacionados con una mejor comprensión de las necesidades, crear y establecer una línea de base para la arquitectura, para el sistema, y mitigar los riesgos de alta prioridad.
- Fase de construcción: las iteraciones en la fase de construcción tienen una estructura de desglose del trabajo similar a las iteraciones en la fase de elaboración, con actividades ocurriendo en paralelo. Hay un énfasis diferente en la forma en que las actividades abordan los objetivos de la fase, sin embargo, se espera que la arquitectura sea estable cuando esta fase se inicia, el resto de los requisitos deben aplicarse en la parte superior de la misma. Otra ventaja de la validación de la arquitectura y la eliminación de la mayor cantidad de riesgos posibles durante la elaboración es proporcionar una mayor previsibilidad en la construcción, lo que permite que el director del

proyecto pueda centrarse en la eficacia del equipo y reducción de costos. La funcionalidad se implementa continuamente, probado e integrado.

- Fase de transición: en la fase de transición, la mayoría de las actividades suceden en paralelo. Los principales objetivos son para afinar la funcionalidad, el rendimiento y la calidad global del producto beta desde el final de la fase de construcción.

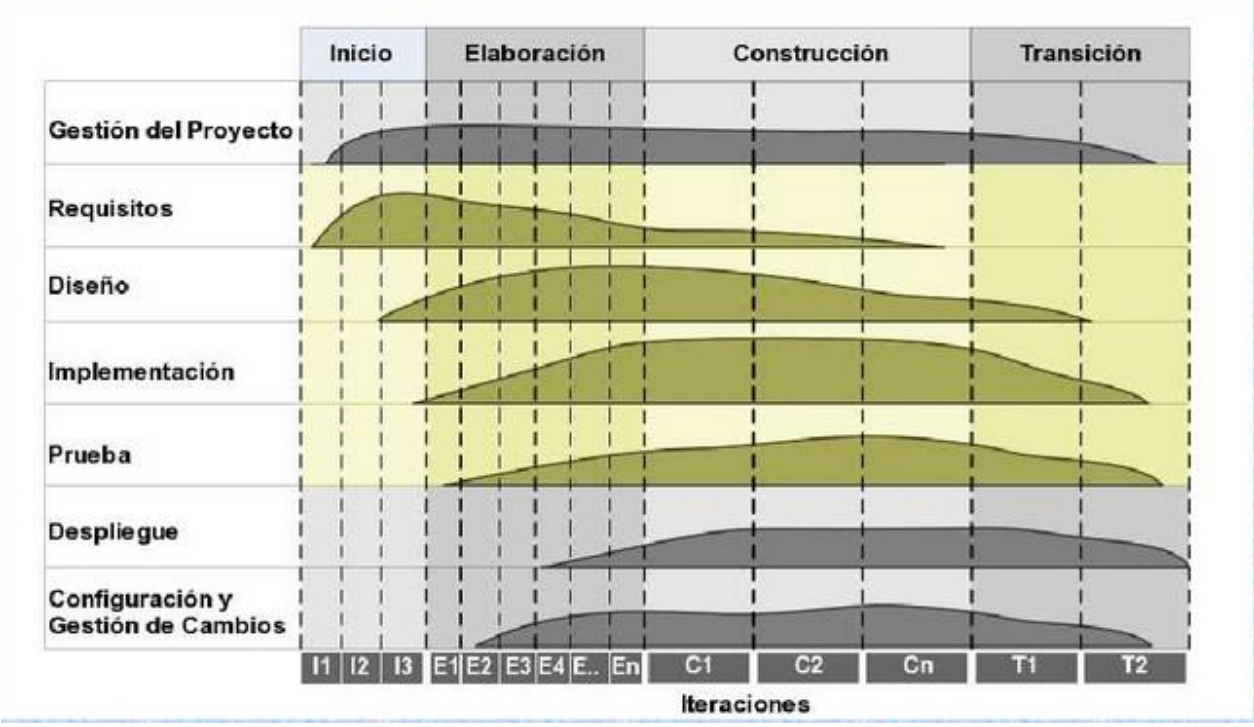


Figura 2: ciclo de vida de OpenUp.

OpenUP es un proceso de desarrollo de software mínimamente suficiente, esto quiere decir que incluye solo el contenido fundamental, esto es que no provee orientación sobre temas en los que el proyecto tiene que lidiar, como son: el tamaño del equipo, el cumplimiento, seguridad, orientación tecnológica entre otras. Sin embargo OpenUP es completa en el sentido de que manifiesta por completo el proceso de construir un sistema. Para atender las necesidades que no están cubiertas en su contenido OpenUp es extensible a ser utilizado como base sobre la cual se pueden añadir o adaptarse a contenido de otro proceso que sea necesario (Juarez, 2013).

XP (por sus siglas en inglés Extreme Programming)

Es uno de varios procesos ágiles. Hace hincapié en la satisfacción del cliente. Faculta a sus desarrolladores para responder con seguridad a las necesidades cambiantes de los clientes. Enfatiza el trabajo en equipo. Los gerentes, clientes y desarrolladores son socios iguales en un equipo de colaboración. El equipo se auto organiza alrededor del problema para resolverlo lo más eficientemente posible.

Extreme Programming mejora un proyecto de software en cinco formas esenciales; la comunicación, la sencillez, la retroalimentación, el respeto y el valor. Los programadores extremos se comunican constantemente con sus clientes, mantienen su diseño simple y limpio, entregan el sistema a los clientes tan pronto como sea posible e implementan cambios como se sugiere (extremeprogramming.org, 2013).

A continuación se muestran las fases de la metodología ágil XP:

- Fase de planificación del proyecto: el primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente.
- Fase de diseño: sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente y entendible.
- Fase de codificación: el cliente es una parte más del equipo de desarrollo, su presencia es indispensable en las distintas fases de XP.
- Fase de prueba: uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos que se vayan implementando.

SCRUM

Scrum en sí es un marco sencillo para el desarrollo de software. Ken Schwaber y Jeff Sutherland han escrito La Guía Scrum. O sea, es el nombre con el que se denomina a los marcos de desarrollo ágiles caracterizados por (Scrum.org, 2015):

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.

- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o de cascada.

A continuación se muestran las fases de la metodología ágil SCRUM:

- Concepto: desarrollo de la visión y alcances del producto. Se determina el grupo de trabajo.
- Especulación: se parte de la visión del producto. Se realizan hipótesis (especulaciones acerca de lo que se entiende o se desea producir).
- Exploración: desarrollo de las funcionalidades. Genera un incremento del producto.
- Revisión: se verifica el cumplimiento de la visión y los objetivos por parte del subproducto generado.
- Cierre: entrega pactada de un subproducto: listo, revisado y probado.

AUP (por sus siglas en inglés Agil Unified Process)

La Metodología AUP es una versión simplificada de RUP que aplica técnicas ágiles. Esta describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software, realizando iteraciones dentro del proyecto y descomponiendo a este en mini proyectos con el objetivo de tener un mayor control sobre cada una de las iteraciones.

Se enfoca en la determinación de riesgos en etapas tempranas, para que el aplicativo sea adaptable a la gestión de cambios. Bajo este parámetro se propone la determinación de requerimientos, basada en el modelado de casos de uso, en donde se determina la funcionalidad total del sistema (Alpízar y Argüello, 2006).

A continuación se muestran las fases de la metodología ágil AUP:

- Iniciación: el objetivo principal de la fase de iniciación es archivar el consenso de los interesados del proyecto en relación a los objetivos del proyecto.
- Elaboración: el principal objetivo de la fase de elaboración es probar la arquitectura del sistema a desarrollar. El punto es asegurar que el equipo puede desarrollar un sistema que satisfaga los requisitos, y la mejor manera de hacerlo que es la construcción de extremo a extremo del esqueleto de trabajo del sistema conocido como "prototipo de la arquitectura".

- Construcción: el objetivo de la fase de construcción consiste en desarrollar el sistema hasta el punto en que está listo para la pre-producción de pruebas. En las etapas anteriores, la mayoría de los requisitos han sido identificados y la arquitectura del sistema se ha establecido.
- Transición: la fase de transición se enfoca en liberar el sistema a producción. Deben hacerse pruebas extensivas a lo largo de esta fase.

Analizadas las características de las anteriores metodologías de desarrollo de software, es seleccionada OpenUp como la candidata para guiar el desarrollo de la Capa de Servicios, ya que se ajusta al ambiente que se presenta, además, se centra en grupos pequeños de trabajo, tiempo limitado y genera una cantidad mínima de artefactos.

1.3. Herramientas y tecnologías

Las herramientas son programas, aplicaciones o simplemente instrucciones que ofrecen la posibilidad de realizar varias funcionalidades con diferentes propósitos. Por otra parte, las tecnologías son el conjunto de conocimientos técnicos, ordenados científicamente que permiten diseñar y crear bienes y servicios. A continuación se muestran todas las herramientas y tecnologías utilizadas para el desarrollo de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.

1.3.1. Lenguaje de programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que se pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes (Marin, 2008).

A continuación se muestra el lenguaje de programación utilizado para el desarrollo de la Capa de Servicios Web:

Java

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana, también posee otras características importantes que a continuación se muestran (EXES, 2015):

- Es un lenguaje que es compilado, generando ficheros de clases compilados, pero estas clases compiladas son en realidad interpretadas por la máquina virtual java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma: el mismo código java que funciona en un sistema operativo funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.

Para el desarrollo de la Capa de Servicios Web se utiliza la máquina virtual de Java JDK 7, la cual permite crear aplicaciones y componentes usando el lenguaje Java, JDK 7 según Oracle esta nueva versión actualiza algunos algoritmos de seguridad y los algoritmos criptográficos, además se añaden nuevas herramientas en función de controlar los estilos de las páginas generadas y hacer un tratamiento diferente de los errores (Oracle, 2015b)

1.3.2. Servidor web

Apache Tomcat

Es una aplicación de código abierto de las tecnologías Java Servlet y JavaServer Pages. Las especificaciones Java Servlet y JavaServer Pages se desarrollan bajo la Comunidad Java. Apache Tomcat se desarrolla en un entorno abierto y participativo y liberado bajo la licencia Apache versión 2 (tomcat.apache.org 2015b).

Para llevar a cabo el desarrollo de la Capa de Servicios Web se hará uso de Apache Tomcat en su versión 8.0.3 porque implementa el Servlet 3.1 y JavaServer Pages 2.3 especificaciones del Java Community Process, e incluye una plataforma útil para el desarrollo y despliegue de aplicaciones y servicios web (tomcat.apache.org, 2015a).

1.3.3. Entorno de desarrollo

Un entorno de desarrollo informático IDE (Integrated Development Environment) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios (Galiano, 2015)

NetBeans IDE

NetBeans IDE es una aplicación de código abierto ("open source") diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. NetBeans IDE dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco sus funcionalidades son ampliables mediante la instalación de paquetes (Ramírez, 2015).

Para llevar a cabo la implementación de la Capa de Servicios Web se hará uso de NetBeans en su versión 8.0 por las siguientes características (Oracle 2015d):

- NetBeans IDE 8.0 proporciona analizadores de código y editores para trabajar con las últimas tecnologías Java 8 - Java SE 8, Java SE Embedded 8 y Java ME Embedded 8.
- Cuenta con nuevas herramientas para HTML5.
- Tiene mejoras en la compatibilidad con PHP y C / C ++.

1.3.4. Sistema Gestor de Bases de Datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: DataBase Management System) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejarlos adecuadamente (Bertino y Martino, 1995).

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es un sistema de gestión de bases de datos de código abierto y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (postgresql.org.es, 2010).

Para el desarrollo de la Capa de Servicios Web se utiliza como Sistema Gestor de Bases de Datos la versión 9.3 de PostgreSQL. Las principales características de la versión se muestran a continuación (postgresql.org, 2013):

Esta versión amplía la fiabilidad, disponibilidad y capacidad de integración de PostgreSQL con otras bases de datos. Los usuarios ya están descubriendo que al utilizar la versión 9.3 pueden crear aplicaciones que antes no habrían sido posibles. PostgreSQL 9.3 permite:

- Hacer visitas sencilla auto-actualizable.
- Añadir funciones para el tipo de datos JSON, incluidos los operadores y funciones para extraer elementos de valores JSON.
- Añadir soporte para activadores de eventos.
- Añadir capacidad opcional de páginas de datos de suma de comprobación y denunciar la corrupción.

Para administrar la base de datos se utiliza pgAdmin III por las siguientes características:

pgAdmin III es una aplicación de diseño y administración de bases de datos para su uso con PostgreSQL. Está escrito en C ++ usando las wxWidgets, marco multiplataforma para permitir que se ejecute en múltiples plataformas (postgresql.org, 2003).

1.3.5. Lenguaje de modelado

Un lenguaje de modelado es un lenguaje artificial diseñado para expresar modelos. Ya que los modelos habitualmente se muestran en forma de diagramas por comodidad, los lenguajes de modelado suelen incorporar notaciones gráficas. Igual que los lenguajes naturales, los lenguajes de modelado poseen un

léxico, es decir, un conjunto de palabras que existen en el lenguaje; y una sintaxis, es decir, un conjunto de reglas que nos dice cómo se pueden combinar dichas palabras para componer "frases" que tengan sentido. Las "palabras" de los lenguajes de modelado no se transmiten mediante texto o sonido como en el caso de los lenguajes naturales, sino que, habitualmente, lo hacen en forma de iconos o dibujos para facilitar la visualización formal de los conceptos, que son abstractos (ConML, 2015).

UML

UML (Unified Modeling Language), Lenguaje Unificado de Modelado, es una notación estándar para el modelado de objetos del mundo real, como primer paso en el desarrollo de una metodología de diseño orientado a objetos (Rouse, 2015).

Para llevar a cabo el modelado de la Capa de Servicios Web se utilizará UML en su versión 2.0 por las siguientes características:

- **Visualizar:** permite expresar de una forma gráfica un sistema de forma tal que otro lo pueda entender.
- **Especificar**: permite especificar cuáles son 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.

1.3.6. Herramienta CASE (por sus siglas en inglés Computer-Aided Software Engineering)

El diseño de software normalmente se efectuará con un poco de ayuda de las herramientas de ingeniería de software asistida por ordenador, o herramientas CASE. Es básicamente el uso de apoyo basado en la computadora por los desarrolladores para desarrollar y mantener el software, especialmente en la escala más grande, o para los proyectos más complejos. Las herramientas CASE permiten a los ingenieros de software dar un paso atrás de las complejidades reales de código al mirar el cuadro más grande y el diseño de sus proyectos más grandes. Desde el desarrollo, el diseño del sistema, la codificación, a través de la prueba y mantenimiento, estas herramientas informáticas pueden utilizarse durante todo el ciclo de

vida de software para asegurar que el producto final es de alta calidad, con defectos mínimos, y en la mayor parte de tiempo eficiente y de manera rentable posible (SoftwareEngineerInsider.com, 2014).

Como herramienta CASE para el modelado de la Capa de Servicios se selecciona **Visual Paradigm 8.0**, el cual cuenta con las siguientes características:

Es una plataforma de modelado diseñado para apoyar a los arquitectos de sistemas, desarrolladores y diseñadores UML para acelerar el proceso de análisis y diseño de aplicaciones empresariales, facilita la visualización UML en la última notación de UML 2.1 (Soft112, 2014).

1.3.7.Marcos de trabajo

Visión general amplia, esquema, o el esqueleto de elementos interconectados que soporta un enfoque particular de un objetivo específico, y sirve como una guía que se puede modificar según sea necesario mediante la adición o eliminación de elementos (BusinessDictionary, 2015). A continuación se muestran las características de los marcos de trabajo seleccionados para el desarrollo de la Capa de Servicios Web.

RESTful Web Services framework Jersey 2.0

Es de código abierto, es un marco para el desarrollo de servicios web RESTful en Java (Oracle, 2015c).

Jersey 2.0 cuenta con una API de cliente para la comunicación con un servicio REST o cualquier otro servicio Web expuesto en el protocolo HTTP. Además cuenta con las siguientes características:

- Cuenta con filtros e interceptores que hacen posible observar y modificar los mensajes entrantes y salientes en el cliente, así como en el lado del servidor.
- Clientes y Servicios asíncronos. En el servidor, se puede incrementar el rendimiento mediante el uso de menos recursos (Stenberg, 2013).

JPA 2.1 (por sus siglas en inglés Java Persistence API)

Proporciona a los desarrolladores de Java con un centro de mapeo para la gestión de datos relacionales en aplicaciones Java (Oracle, 2015a).

A continuación se muestran algunas de las características de JPA 2.1:

- Permite conversiones de código entre los tipos de bases de datos y objetos.
- Permite actualizaciones masivas.
- Permite la generación de tablas automáticas, índices y la generación de esquemas.
- Permite la fusión de objetos.

  (wikibooks.org, 2013).

Conclusiones del capítulo

Para llevar a cabo el desarrollo de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística se realizó un estudio de los conceptos principales asociados a Servicios Web los cuales conformaron el punto de partida para sentar las bases tanto para el posterior diseño como para la implementación de la Capa de Servicios Web. Para guiar el proceso de desarrollo de la Capa de Servicios Web se seleccionó como metodología de desarrollo de software a OpenUp debido a que es una metodología ágil, centrada en equipos pequeños de trabajo, enfocada a un período corto de tiempo y genera una cantidad mínima de artefactos. El lenguaje de modelado previsto es UML 2.0 y como herramienta CASE Visual Paradigm en su versión 8.0. Por otra parte Java es el lenguaje de programación escogido mientras que NetBeans 8.0 es seleccionado como IDE de desarrollo. Se utiliza a PostgreSql 9.3 como Sistema Gestor de Bases de Datos y al administrador de bases de datos pgadmin III. Para llevar a cabo la implementación de la Capa de Servicios Web se utilizaron los frameworks: Jersey en su versión 2.0 y JPA en su versión 2.1.

Capítulo 2: Análisis y diseño de la solución

Introducción

En el presente capítulo se muestra el modelo de dominio, se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr que la Capa de Servicios funcione correctamente. Se describe el análisis y diseño de las clases de la Capa de Servicios Web. Se construyen los artefactos correspondientes al análisis y diseño según la metodología seleccionada en el capítulo anterior, modelando la estructura de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.

2.1. Modelo de dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales (término utilizado en la primera edición del libro de Larman), modelo de objetos del dominio y modelos de objetos de análisis. Utilizando la notación UML, un modelo del dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Pueden mostrar: objetos del dominio o clases conceptuales, asociaciones entre las clases conceptuales y atributos de las clases conceptuales (Larman, 2003). A continuación se muestra el modelo de dominio para lograr una mejor comprensión de los conceptos de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.

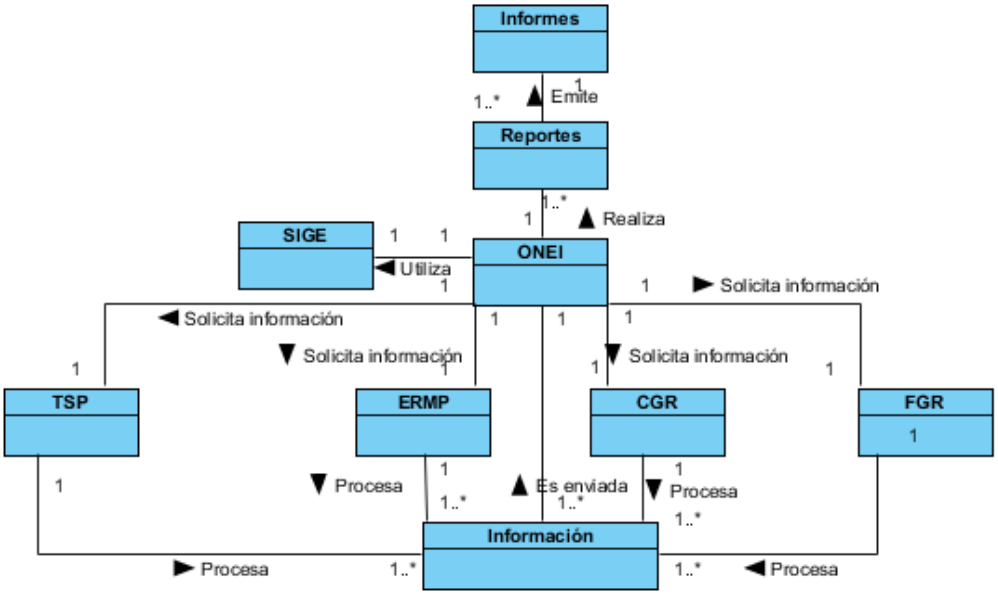


Figura 3: modelo de dominio.

Descripción de los conceptos del dominio

ONEI: entidad que representa la Oficina Nacional de Estadística e Información, es el órgano encargado de manejar toda la información estadística proveniente de los organismos asociados.

SIGE: entidad que representa al Sistema Integrado de Gestión Estadística, sistema desarrollado con el objetivo de analizar la información estadística.

TSP: entidad que representa al Tribunal Supremo Popular, organismo asociado a la ONEI y que cuenta con SIGE.

CGR: entidad que representa a la Contraloría General de la República, organismo asociado a la ONEI y que cuenta con SIGE.

ERMP: entidad que representa a la Empresa Recuperadora de Materias Primas, organismo asociado a la ONEI y que cuenta con SIGE.

FGR: entidad que representa a la Fiscalía General de la República, organismo asociado a la ONEI y que cuenta con SIGE.

Información: entidad que representa la información que los organismos procesan y que es solicitada por la ONEI, realizándose a través de plantillas que ya están predefinidas.

Reportes: entidad que representa a los reportes que son realizados una vez que la ONEI obtiene de los organismos la información solicitada mediante las plantillas.

Informes: entidad que representa a los informes que son realizados por la ONEI una vez visualizados los reportes y enviados a la entidad del gobierno que solicitó la información.

2.2. Requisitos del sistema

En la ingeniería del software, los requisitos se utilizan como datos de entrada en la etapa de diseño del producto y establecen qué debe hacer el sistema, pero no cómo hacerlo. Son una condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. En fin son algo que el sistema debe hacer o una cualidad que el sistema debe poseer (Somerville, 2007).

2.2.1. Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno de un software, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software (Somerville, 2007). Para un mejor entendimiento de la descripción de los

requisitos de la Capa de Servicios Web se hace necesario un análisis de los términos que a continuación se exponen:

- **Plantillas de encuestas o formularios**: son plantillas predefinidas por SIGE, solamente contienen una estructura sin datos.
- **Formularios o encuestas**: son las plantillas de formularios o encuestas con datos ya introducidos.
- **JSON:** por sus siglas en inglés JavaScript Object Notation, es un formato ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La Capa de Servicios Web debe cumplir con los requisitos funcionales que a continuación se describen:

RF_1: autenticar usuario.
Descripción: permite al usuario acceder a la Capa de Servicios Web.
Entrada: usuario y contraseña.
Salida: usuario autenticado.

RF_2: obtener plantillas de encuesta.
Descripción: permite al usuario ver todas las plantillas de encuesta publicadas hasta el momento.
Entrada: URL específica para obtener las plantillas de encuesta introduciendo como parámetros un rango en el que el inicio o el fin o ambos pueden ser opcional.
Salida: JSON.

RF_3: obtener plantillas de formulario.
Descripción: permite al usuario ver todas las plantillas de formulario publicadas hasta el momento.
Entrada: URL específica para obtener las plantillas de formulario introduciendo como parámetros un rango en el que el inicio o el fin o ambos pueden ser opcional.
Salida: JSON.

RF_4: obtener plantilla específica de encuesta.
Descripción: permite obtener la plantilla de la encuesta especificada por el usuario en la petición realizada.

Entrada: URL con parámetros de la encuesta (id del número y subnúmero de modelo).
Salida: JSON.

RF_5: obtener plantilla específica de formulario.
Descripción: permite obtener una plantilla específica de formulario seleccionada por el usuario.
Entrada: URL con parámetros del formulario (id del número y subnúmero de modelo).
Salida: JSON.

RF_6: captar datos de formulario.
Descripción: permite introducirle datos a la plantilla de formulario.
Entrada: URL con parámetros del formulario (id del número de modelo, id de subnúmero de modelo) y JSON.
Salida: JSON.

RF_7: validar formulario.
Descripción: permite verificar que todos los datos en el formulario sean correctos de acuerdo con las normas establecidas.
Entrada: URL con parámetros del formulario (id del número de modelo, id de subnúmero de modelo, id de fecha de informe acumulado, id de código de variante, código del centro informante).
Salida: JSON.

RF_8: archivar formulario.
Descripción: permite cambiar a estado archivado uno o varios formularios que se encuentran en estado pendiente a archivar.
Entrada: URL específica para archivar uno o varios formularios y JSON
Salida: JSON.

RF_9: visualizar formulario.
Descripción: permite al usuario poder ver los datos de un formulario seleccionado.
Entrada: URL con parámetros del formulario URL con parámetros del formulario (número de modelo, subnúmero de modelo, id de fecha del informe acumulado, id de código de variante y código del centro informante de formulario).

Salida: JSON.

RF_10: eliminar formulario.
Descripción: permite eliminar un formulario existente.
Entrada: URL con parámetros del formulario (número de modelo, subnúmero de modelo, id de fecha del informe acumulado, id de código de variante y código del centro informante de formulario).
Salida: JSON.

RF_11: editar formulario.
Descripción: permite modificar algún dato de los existentes a una plantilla de formulario.
Entrada: URL con parámetros del formulario (número de modelo, subnúmero de modelo, id de fecha del informe acumulado, id de código de variante y código del centro informante de formulario) y JSON.
Salida: JSON.

RF_12: obtener formularios en estado de “digitación”.
Descripción: permite ver un listado con todos los formularios en estado de “digitación”.
Entrada: URL con parámetros del formulario (id de estado y un rango que puede ser opcional).
Salida: JSON.

RF_13: obtener formularios en estado de “validación”.
Descripción: permite ver un listado con todos los formularios en estado de “validación”.
Entrada: URL con parámetros del formulario (id de estado y un rango que puede ser opcional).
Salida: JSON.

RF_14: obtener formularios en estado de “archivado”.
Descripción: permite ver un listado con todos los formularios en estado de archivado.
Entrada: URL con parámetros del formulario (id de estado y un rango que puede ser opcional).
Salida: JSON.

RF_15: obtener formularios en estado de “validación pendientes a archivar”.
Descripción: permite ver un listado de todos los formularios en estado de “validación pendientes a archivar.
Entrada: URL con parámetros del formulario (id de estado y un rango que puede ser opcional).

Salida: JSON.

RF_16: obtener formularios en estado de “validado con errores”.
Descripción: permite ver un listado con todos los formularios en estado de “validado con errores”.
Entrada: URL con parámetros del formulario (id de estado y un rango que puede ser opcional).
Salida: JSON.

RF_17: obtener formularios en estado de validación “pendientes a validar”.
Descripción: permite ver un listado con todos los formularios en estado de “validación” pendientes a validar.
Entrada: URL con parámetros del formulario (id de estado y un rango que puede ser opcional).
Salida: JSON.

RF_18: enviar formulario a digitación.
Descripción: permite enviar uno o varios formularios del estado “archivado” al estado de “digitación”.
Entrada: URL específica para enviar el formulario a digitación y JSON.
Salida: JSON.

RF_19: obtener formulario específico.
Descripción: permite obtener un formulario de los existentes.
Entrada: URL con parámetros del formulario (id del número de modelo, id de subnúmero de modelo, id de fecha de informe acumulado, id de código de variante, código del centro informante).
Salida: JSON.

RF_20: obtener detalles de validación de un formulario.
Descripción: permite obtener las alertas y los errores de un formulario que se encuentra en estado de “validación”.
Entrada: URL con parámetros del formulario (id del número de modelo, id de subnúmero de modelo, id de fecha de informe acumulado, id de código de variante, código del centro informante).
Salida: JSON.

RF_21: captar datos de la encuesta.
Descripción: permite introducirle datos a la plantilla de encuesta.

Entrada: URL con parámetros de la encuesta (id de número de modelo, id de subnúmero de modelo) y JSON.
Salida: JSON.

RF_22: eliminar encuesta.
Descripción: permite eliminar una encuesta existente.
Entrada: URL con los parámetros de la encuesta (id de la encuesta).
Salida: JSON.

RF_23: editar encuesta.
Descripción: permite modificar algún dato de los existentes a una plantilla de encuesta.
Entrada: URL con parámetros de la encuesta (id de la encuesta) y JSON.
Salida: Base de Datos actualizada.

RF_24: obtener encuestas en estado de “encuesta cerrada”.
Descripción: permite obtener un listado con todas las encuestas en estado de “encuesta cerrada”.
Entrada: URL con parámetros de la encuesta (id de estado y un rango que puede ser opcional).
Salida: JSON.

RF_25: obtener encuestas en estado de “digitación”.
Descripción: permite ver un listado con todas las encuestas en estado de “digitación”.
Entrada: URL con parámetros de la encuesta (id de estado y un rango que puede ser opcional).
Salida: JSON.

RF_26: enviar encuesta a “digitación”.
Descripción: permite enviar una o varias encuestas de “archivado” a “digitación”.
Entrada: URL específica para enviar la encuesta a digitación y JSON.
Salida: JSON.

RF_27: obtener encuesta específica.
Descripción: permite obtener una encuesta de las existentes en la Base de Datos.
Entrada: URL con parámetros de la encuesta (id de la encuesta).
Salida: JSON.

RF_28: cerrar encuesta.
Descripción: permite cerrar una encuesta una vez que el usuario la capta.
Entrada: URL con parámetros de la encuesta (id de encuesta).
Salida: JSON.

RF_29: obtener imagen de la encuesta.
Descripción: permite obtener una imagen que posee la encuesta para una pregunta de tipo imagen.
Entrada: URL específica para obtener una imagen y JSON.
Salida: JSON.

RF_30: subir archivo de la encuesta.
Descripción: permite subir un archivo a una encuesta para la pregunta de tipo archivo.
Entrada: URL con parámetros de la encuesta (id de modelo, id de subnúmero de modelo y el tamaño del archivo permitido en megabytes) y la dirección del archivo.
Salida: JSON.

A continuación se muestran ejemplos de una URL que se utiliza como entrada y de un JSON para una mejor comprensión de la Capa de Servicios Web.

Figura 4: ejemplo de URL de la Capa de Servicios Web.


```
{
  "success": true,
  "total": 0,
  "templatesurvey": {
    "idnummodelo": "110",
    "idsubnummodelo": "20",
    "idestado": 2,
    "tipodemodelo": 8,
    "idclasificaciondeseguridad": 1,
    "idpiedefirma": 5,
    "periodicidad": 0,
```


Figura 5: ejemplo de fragmento de JSON de la Capa de Servicios Web.

2.2.2. Requisitos no funcionales

Los requisitos no funcionales son propiedades que hacen al producto atractivo, usable, rápido o confiable. Se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar. Además se conocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software (Somerville, 2007).

Requisitos de Seguridad

RnF_1: la información manejada por la Capa de Servicios deberá estar protegida de acceso no autorizado.

Requisitos de Software

RnF_2: el servidor donde se instalará la Capa de Servicios debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 14.10.

RnF_3: el servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 14.10.
- PostgreSQL versión 9.3.
- PgAdmin III como administrador de Bases de Datos para PostgreSQL.

Requisitos de Hardware

RnF_4: la PC Servidor debe cumplir con los siguientes requisitos de hardware:

- Ordenador Core 2 Duo V, con 2.2 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 4 Gb.

Requisitos de Restricciones de diseño e implementación

RnF_5: lenguaje y marco de trabajo para el desarrollo de la Capa de Servicios Web.

La capa de Servicios Web deberá ser implementada en el lenguaje de programación Java utilizando la JDK 7 y utilizar los frameworks Jersey en su versión 2.0 y JPA en su versión 2.1.

2.3. Diagrama de Casos de uso del sistema

El diagrama de casos de uso representa la forma en cómo un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso) (users.dcc, 2015).

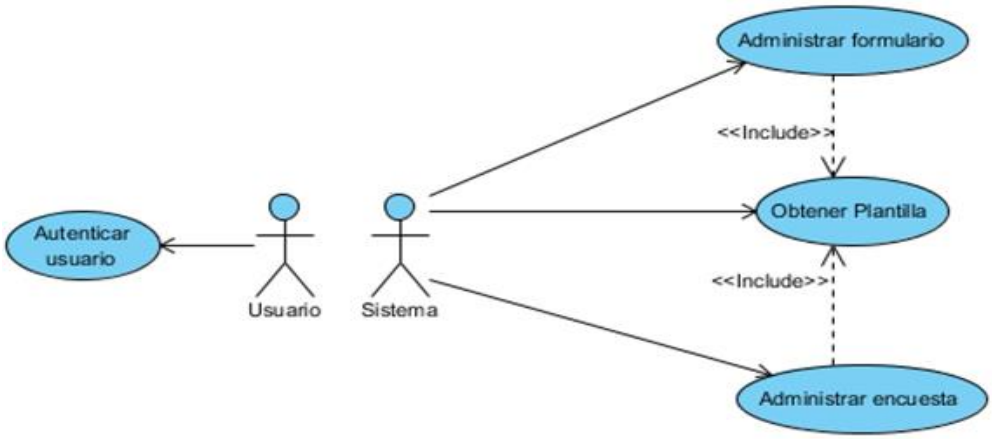


Figura 6: diagrama de casos de uso del sistema.

2.3.1. Descripción de los casos de uso

CUS_1: autenticar usuario: caso de uso que permite que un usuario mediante su usuario y contraseña acceda a la Capa de Servicios con los privilegios correspondientes.

CUS_2: obtener plantilla: caso de uso que permite obtener todas las plantillas de encuesta, las plantillas de formulario u obtener una plantilla específica de encuesta o formulario mediante una URL con parámetros como entrada y JSON de salida.

CUS_3: administrar formulario: caso de uso que permite realizar varias operaciones sobre los formularios entre las que se encuentran: captar, visualizar, eliminar, editar, validar, archivar, obtener formularios de acuerdo a los estados en que se pueden encontrar, enviar formulario a digitación, obtener un formulario específico y obtener los detalles de validación de formulario, esos detalles son los errores y las alertas de validación.

CUS_4: administrar encuesta: permite realizar varias operaciones sobre las encuestas como: captar, eliminar, obtener encuestas en estado de digitación o en estado de archivado, editar, enviar encuesta a digitación, cerrar, obtener encuesta específica, obtener imagen y subir archivo de la encuesta.

2.3.2. Patrones de casos de uso

Contrario a lo que se pudiera pensar, un patrón de casos de uso no describe un uso particular de un sistema. Más bien, captura técnicas para que el modelo sea mantenible, reusable, y entendible. Entonces los patrones de casos de uso capturan mejores prácticas para modelar casos de uso (Buzz, 2015).

CRUD (Create Read Update Delete) Parcial: en el diseño de la Capa de Servicios Web se utiliza el patrón de casos de uso CRUD parcial. Consiste en agrupar un caso de uso faltando algunas de las operaciones básicas de gestión (buscar, eliminar, insertar, actualizar). Ejemplo de ello se evidencia en los casos de uso: administrar formulario y Administrar encuesta.

Inclusión concreta: se incluye una relación del caso de uso base al caso de uso de inclusión. El último puede ser instalado en sí mismo. El caso de uso base puede ser concreto o abstracto. Un ejemplo del patrón en la Capa de Servicios Web se evidencia en los casos de uso: administrar formulario y administrar encuesta que incluyen al caso de uso obtener plantilla.

2.3.3. Descripción del caso de uso arquitectónicamente significativo del sistema

Tabla 3: descripción del Caso de Uso "Administrar formulario".

Objetivo	Administrar los formularios de la Capa de Servicios Web.
Actores	Sistema.
Resumen	Caso de uso que permite realizar varias operaciones sobre los formularios entre las que se encuentran: captar, visualizar, eliminar, actualizar, validar, archivar, obtener formularios de acuerdo a los estados en que se pueden encontrar, enviar formulario a digitación, obtener un formulario específico y obtener los detalles de validación de formulario.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Usuario autenticado correctamente en la Capa de Servicios
Postcondiciones	En dependencia de la opción que el usuario escoja realizar: - Se capta un formulario. - Se visualiza un formulario. - Se elimina un formulario.

	- Se edita un formulario. - Se valida un formulario. - Se archiva un formulario. - Se obtienen los formularios en estado de digitación. - Se obtienen los formularios en estado de validación. - Se obtienen los formularios en estado de archivado. - Se obtienen los formularios en estado de validación pendientes a archivar. - Se obtienen los formularios en estado de validación pendientes a validar. - Se obtienen los formularios en estado de validado con errores. - Se envía un formulario a digitación. - Se obtiene un formulario específico. - Se obtienen los detalles de validación de formulario.

Flujo de eventos

Actor	Capa de Servicios Web
El usuario escoge cuál acción realizar: - Captar un formulario. - Visualizar un formulario. - Eliminar un formulario. - Editar un formulario. - Validar un formulario. - Archivar un formulario. - Obtener formularios en estado de digitación. - Obtener formularios en estado de validación. - Obtener formularios en estado de archivado. - Obtener formularios en estado de validación pendientes a archivar. - Obtener formularios en estado de validación pendientes a validar. - Obtener formularios en estado de validado con errores. - Enviar formulario a digitación. - Obtener formulario específico. - Obtener detalles de validación de formulario.	De acuerdo a la acción a realizar: - Captar formulario: ir a la sección 1. - Visualizar formulario: ir a la sección 2. - Validar formulario: ir a la sección 3. - Eliminar formulario: ir a la sección 4. - Editar formulario: ir a la sección 5. - Archivar un formulario: ir a la sección 6. - Obtener formularios en estado de digitación: ir a la sección 7. - Obtener formularios en estado de validación: ir a la sección 7. - Obtener formularios en estado de archivado: ir a la sección 7. - Obtener formularios en estado de validación pendientes a archivar: ir a la sección 7. - Obtener formularios en estado de validación pendientes a validar: ir a la sección 7. - Obtener formularios en estado de validado con errores: ir a la sección 7. - Enviar formulario a digitación: ir a la sección 8. - Obtener formulario específico: ir a la sección 9. - Obtener detalles de validación de formulario: ir a la sección 10.

Sección 1: “Captar formulario”

Flujo básico

Actor	Capa de Servicios Web
1. El usuario selecciona la plantilla (ver descripción del caso de uso obtener plantilla). 2. URL con parámetros del formulario (id del número de modelo, id de subnúmero de modelo) y JSON.	
	1. La Capa de Servicios devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Sección 2: "Visualizar formulario"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Se introduce URL con los parámetros del formulario (número, subnúmero, id de fecha del informe acumulado, id de código de variante y código del centro informante)	
	2. La Capa de Servicios devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Sección 3: "Validar formulario"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Se introduce URL con parámetros del formulario (id del número de modelo, id de subnúmero de modelo, id de fecha de informe acumulado, id de código de variante, código del centro informante).	
	2. La Capa de Servicios devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Sección 4: "Eliminar formulario"	
Flujo básico	
Actor	**Capa de Servicios Web**

1. Se introduce URL con los parámetros del formulario (número, subnúmero, id de fecha del informe acumulado, id de código de variante y código del centro informante).	
	2. Formulario eliminado de la Base de Datos. Base de Datos actualizada. Terminando así el caso de uso.
Sección 5: "Editar formulario"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Se introduce URL con los parámetros del formulario (número, subnúmero, id de fecha del informe acumulado, id de código de variante y código del centro informante de formulario) y JSON.	
	2. Formulario actualizado en la Base de datos. Terminando así el caso de uso.
Sección 6: "Archivar un formulario"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Se introduce URL con parámetros del formulario (id del número de modelo, id de subnúmero de modelo, id de fecha de informe acumulado, id de código de variante, código del centro informante).	
	2. Devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Sección 7: "Obtener un formularios por estado"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Introduce URL los parámetros del formulario (id de estado y un rango que puede ser opcional). El estado puede ser: digitación, validación, archivado, validado con errores, pendiente a archivar y pendiente a validar.	

	2. Devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Sección 8: "Enviar formulario a digitación"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Introduce JSON.	
	2. Devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Sección 9: "Obtener formulario específico"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Introduce URL con los parámetros del formulario (id del número de modelo, id de subnúmero de modelo, id de fecha de informe acumulado, id de código de variante y código del centro informante).	
	2. Devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Sección 10: "Obtener detalles de validación de formulario"	
Flujo básico	
Actor	**Capa de Servicios Web**
1. Introduce URL con los parámetros del formulario (id del número de modelo, id de subnúmero de modelo, id de fecha de informe acumulado, id de código de variante y código del centro informante).	
	2. Devuelve un JSON perteneciente al formulario correspondiente a la petición. Terminando así el caso de uso.
Relaciones	**CU Incluidos** CU_2: obtener plantilla.

2.4. Modelo de diseño

Expande y detalla el modelo de análisis tomando en cuenta todas las implicaciones y restricciones técnicas, donde se distinguen tres capas que son el diseño del sistema, el diseño de objetos y el diseño de la persistencia. Permite experimentar y visualizar el sistema que se construirá. Donde se puede razonar acerca de los requerimientos del sistema con un cliente o usuario final, también a los detalles y la vista que presenta un modelo al usuario (Cana, 2014).

Diagrama de clases del diseño

Un diagrama de clases de diseño representa las especificaciones de las clases e interfaces de software en una aplicación (Larman, 2003). Entre la información general se encuentra:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información acerca del tipo de los atributos. .
- Navegabilidad.
- Dependencias.

2.4.1. Diagrama de clases del diseño del caso de uso: "Administrar Formulario"

La figura 7 representa el diagrama de clases del caso de uso "Administrar Formulario", el diagrama está compuesto de manera general por 5 paquetes. El paquete útil contiene la clase que se encarga de gestionar las fechas. El paquete validator contiene las clases que realizan las validaciones, ValidationProgram constituye la clase más importante de este paquete. El paquete model contiene todas las entidades referentes a los formularios. El paquete domain se encarga de gestionar todas las peticiones y respuestas de los servicios web, Model constituye una de las clases más importante de este paquete y por último el paquete service que realiza todas las operaciones sobre los servicios web, dentro de este paquete se encuentra la clase ModelService la cual usa a los demás paquetes y al framework Jersey 2.0 y JPA 2.1. El paquete model y validator usan también al framework JPA 2.1. Para una mejor comprensión y para más detalles del diagrama de clases ver el artefacto "Modelo de diseño".

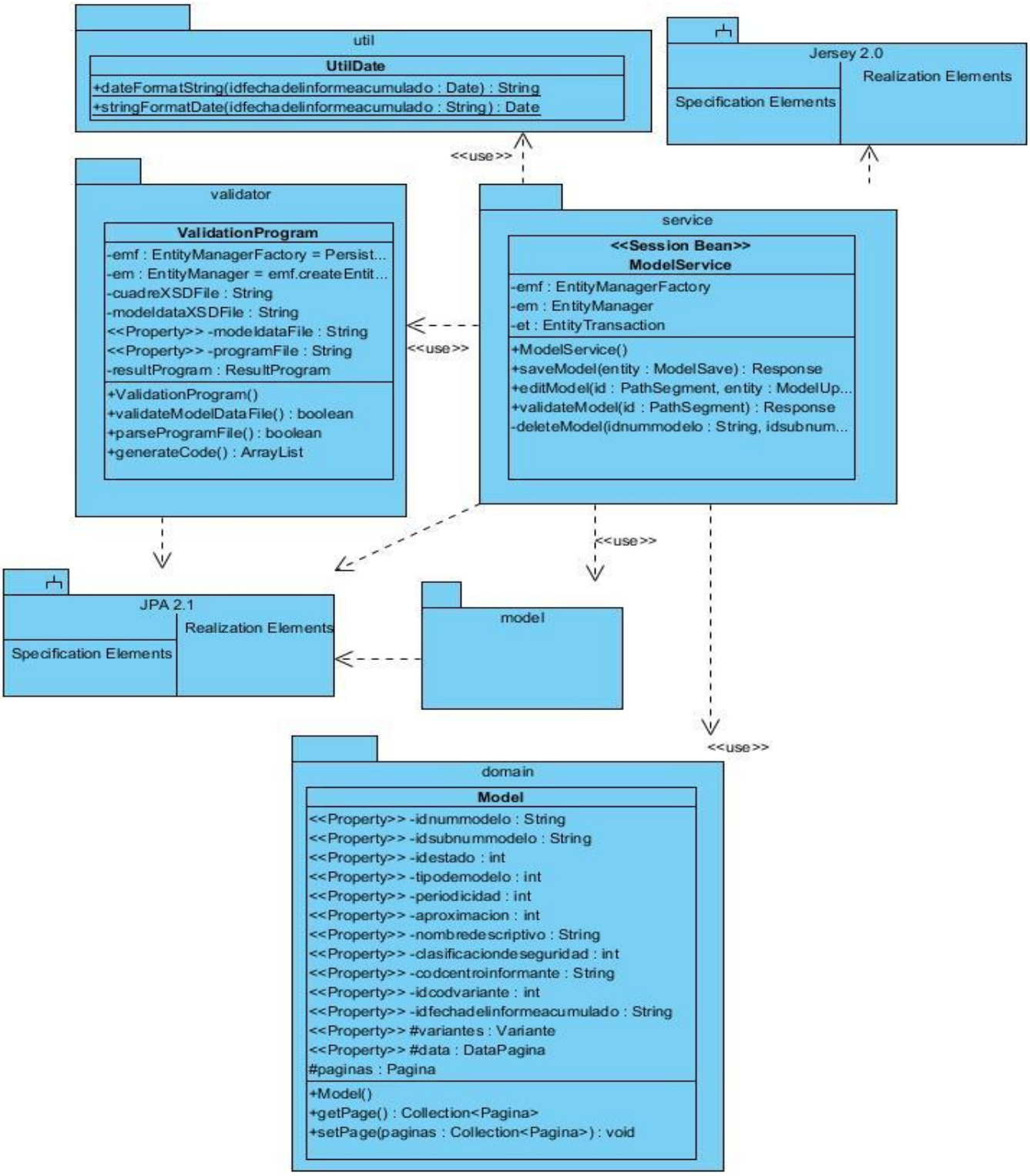


Figura 7: diagrama de clases del caso de uso: "Administrar formulario"

2.5. Patrones utilizados en la solución

Los patrones ayudan a construir sobre la experiencia colectiva de ingenieros de software experimentados. Estos capturan la experiencia existente y que ha demostrado ser exitosa en el desarrollo de software, y ayudan a promover las buenas prácticas de diseño. Cada patrón aborda un problema específico y recurrente en el diseño o implementación de un sistema de software (Buschmann et al., 2008).

2.5.1. Patrón arquitectónico

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos (Welicki, 2015).

Patrón arquitectónico empleado en la Capa de Servicios Web:
Arquitectura basada en Recursos

Una de las más elocuentes presentaciones de arquitecturas peer-to-peer ha sido la disertación doctoral de Roy Fielding, elaborada con anterioridad pero expuesta con mayor impacto en el año 2000. Es en ella donde se encuentra la caracterización más detallada del estilo denominado Representational State Transfer o REST. Aunque la literatura especializada tiende a considerar a REST una variante menor de las arquitecturas basadas en servicios, Fielding considera que REST resulta de la composición de varios estilos más básicos, incluyendo repositorio replicado, cache, cliente-servidor, sistema en capas, sistema sin estado, máquina virtual, código a demanda e interfaz uniforme.

En síntesis muy apretada, podría decirse que REST define recursos identificables y métodos para acceder y manipular el estado de esos recursos. El caso de referencia es nada menos que la World Wide Web, donde los URIs identifican los recursos y HTTP es el protocolo de acceso. El argumento central de Fielding es que HTTP mismo, con su conjunto mínimo de métodos y su semántica simplísima, es suficientemente general para modelar cualquier dominio de aplicación. De esta manera, el modelado tradicional orientado a objetos deviene innecesario y es reemplazado por el modelado de entidades tales como familias jerárquicas de recursos abstractos con una interfaz común y una semántica definida por el propio HTTP. REST es en parte una reseña de una arquitectura existente y en parte un proyecto para un

estilo nuevo. La caracterización de REST constituye una lectura creativa de la lógica dinámica que rige el funcionamiento de la Web (una especie de ingeniería inversa de muy alto nivel), al lado de una propuesta de nuevos rasgos y optimizaciones, o restricciones adicionales (Reynoso y Kiccillof, 2004).

2.5.2. Patrones de diseño

Un patrón de diseño describe una estructura recurrente de componentes que se comunican para resolver un problema general de diseño en un contexto particular. Nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. Identifica las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades (Erich Gamma et al., 1994).

2.5.3. Patrones Grasp

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Grosso, 2011).

En la Capa de Servicios Web se evidencian los siguientes patrones GRASP:

Alta cohesión: en cuanto al diseño de objetos, la cohesión (o de manera más específica, la cohesión funcional) es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas y que no hace una gran cantidad de trabajo, tiene alta cohesión. En la solución se observa el uso de este patrón en la clase Tbdescriptordelmodelo del paquete model y ModelService del paquete service, a las que se le asigna responsabilidades de manera que la cohesión permanezca alta.

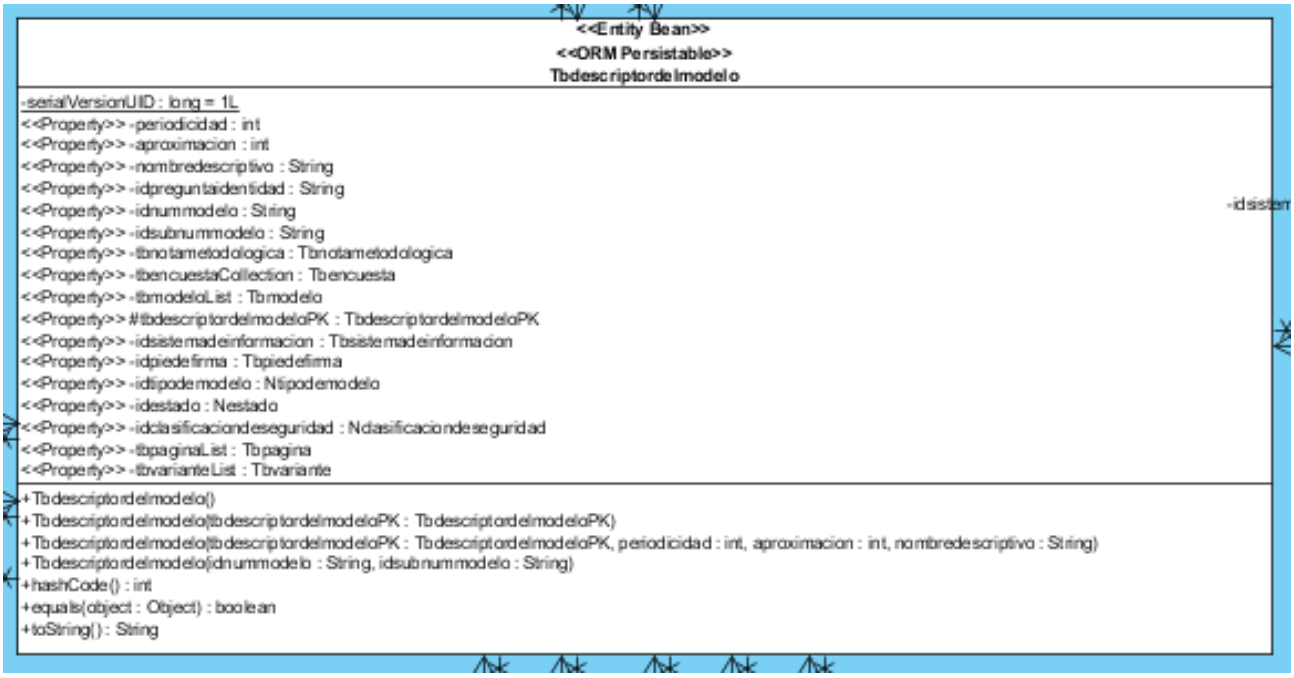


Figura 8: clase Tbdescriptordelmodelo del paquete model.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que un elemento está conectado a otros elementos. Un elemento con bajo (o débil) acoplamiento no depende de demasiados elementos, los cuales pueden ser clases, subsistemas, sistemas. En la solución se observa el uso de este patrón en la clase Napp del paquete model e IndicadoresPagina del paquete domain, a las que se le asigna responsabilidades de manera que el acoplamiento permanezca bajo.

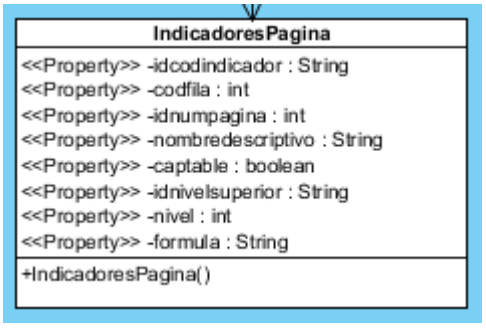


Figura 9: clase IndicadoresPagina del paquete domain.

Experto: asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad. En la solución se observa el uso de este patrón en la clase Tbdescriptordelmodelo del paquete model y Model del paquete domain.

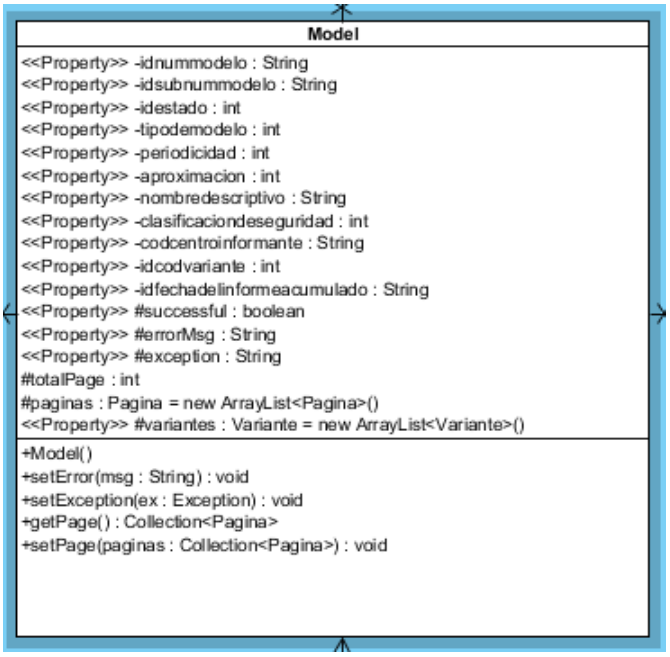


Figura 10: clase Model del paquete domain.

Creador: responde a la pregunta ¿Quién debería ser responsable de crear una nueva instancia de alguna clase? En la solución se observa el uso de este patrón en las clases Model y SModel del paquete domain.

2.5.4. Patrones GoF

Patrones publicados por Gamma, Helm, Johnson y Vlossodes en 1995: patrones del grupo de los cuatro (gang of four). Esta serie de patrones permiten ampliar el lenguaje y aprender nuevos estilos de diseño. Los patrones GoF se clasifican en 3 grupos (eseida, 2015):

- **Creacionales**: conciernen al proceso de creación de objetos.
- **Estructurales**: tratan la composición de clases y/u objetos.
- **Comportamiento**: caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

En la Capa de Servicios Web se evidencia el siguiente patrón GoF:

Fachada / Facade: simplifica el acceso a un conjunto de clases proporcionando una única clase que todos utilizan para comunicarse con dicho conjunto de clases. Este patrón se utiliza en la clase ModelService que es la que proporciona los servicios asociados al recurso formulario.

2.6. Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos (sparxsystems, 2013).

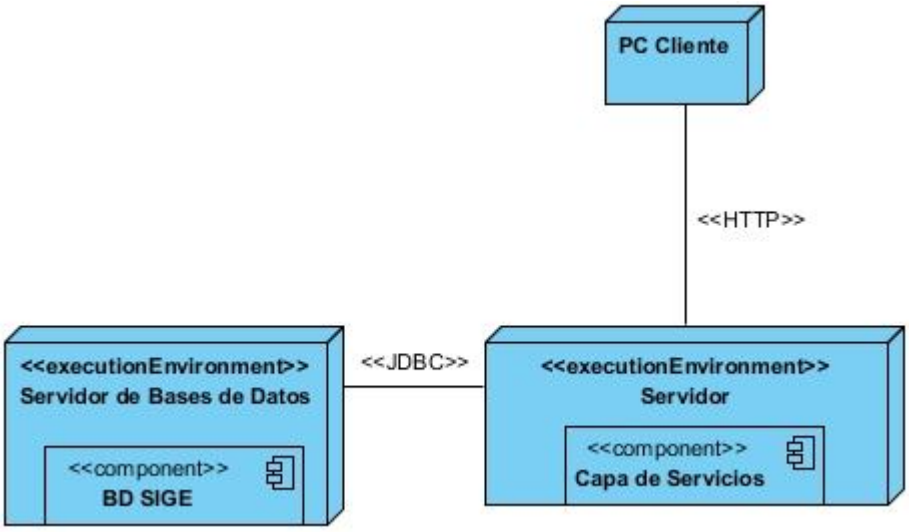


Figura 11: diagrama de despliegue.

El diagrama representa 3 nodos principales. El nodo PC Cliente que requiere de un navegador web, el nodo Servidor Web donde debe estar instalada la Capa de Servicios Web y el nodo Servidor de Bases de Datos en el cual debe estar instalado el Sistema Gestor de Bases de Datos PostgreSQL con la Base de Datos de SIGE. El nodo PC Cliente estará conectado al nodo Servidor Web mediante el protocolo HTTP mientras que el nodo Servidor Web estará conectado al nodo Servidor de Bases de Datos mediante JDBC (Java Database Connectivity).

Conclusiones del capítulo

En el desarrollo del presente capítulo se realizó una representación visual de clases conceptuales del entorno real de los objetos del proyecto a través del modelo de dominio. Se describieron los 30 requisitos

funcionales y 5 no funcionales que la Capa de Servicios debe cumplir para su correcto funcionamiento, agrupándose los RF en 4 casos de uso. Se aplicaron patrones de casos de uso para un mejor entendimiento y comprensión del diagrama de casos de uso así como patrones de diseño como son los GRASP y los GoF dirigidos a diseñar eficazmente la Capa de Servicios Web. Se realizó el diagrama de clases del diseño para la representación de las clases y las relaciones entre ellas. Por último se confeccionó el diagrama de despliegue para representar las relaciones físicas entre los componentes de software y hardware de la Capa de Servicios Web.

Capítulo 3: Implementación y prueba

Introducción

En el presente capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, así como el diagrama de componentes de la Capa de Servicios Web. Se describen las pruebas a realizar, con el objetivo de probar las funcionalidades de la Capa de Servicios en los diferentes escenarios, verificando que los resultados sean los esperados y que la Capa de Servicios funcione correctamente.

3.1. Modelo de Implementación

La implementación brinda la posibilidad de probar y desarrollar componentes como unidades, que finalmente serán integrados como un sistema ejecutable basándose en las especificaciones de diseño.

3.1.1. Diagrama de componentes

El propósito principal del diagrama de componentes es mostrar las relaciones estructurales entre los componentes de un sistema. Los componentes se consideran, unidades autónomas encapsuladas dentro de un sistema o subsistema que proporcionan una o más interfaces. Los diagramas de componentes son generalmente dirigidos al personal de aplicación de un sistema, presenta una comprensión temprana del sistema global que se está construyendo (Bell, 2004).

Seguidamente se muestra el diagrama de componentes del CU “Administrar Formulario” el cual cuenta con 3 paquetes de implementación básicos. Estos paquetes son:

- El paquete de clases Domain, que agrupa las clases que permiten la interacción directa con el paquete Service.
- El paquete de clases Model, que agrupa las clases que permiten la interacción directa con el paquete Service y además con la base de datos de SIGE.
- El paquete de clases Service, que agrupa las clases que interactúan con el paquete de clases Domain y con el paquete de clases Model.
- El paquete de clases útil, que agrupa las clases que permiten trabajar con algunas utilidades sobre los formularios como por ejemplo la fecha y además interactúa con el paquete service.

- El paquete de clases validator, que agrupa las clases que permiten la interacción directa con el paquete Service.

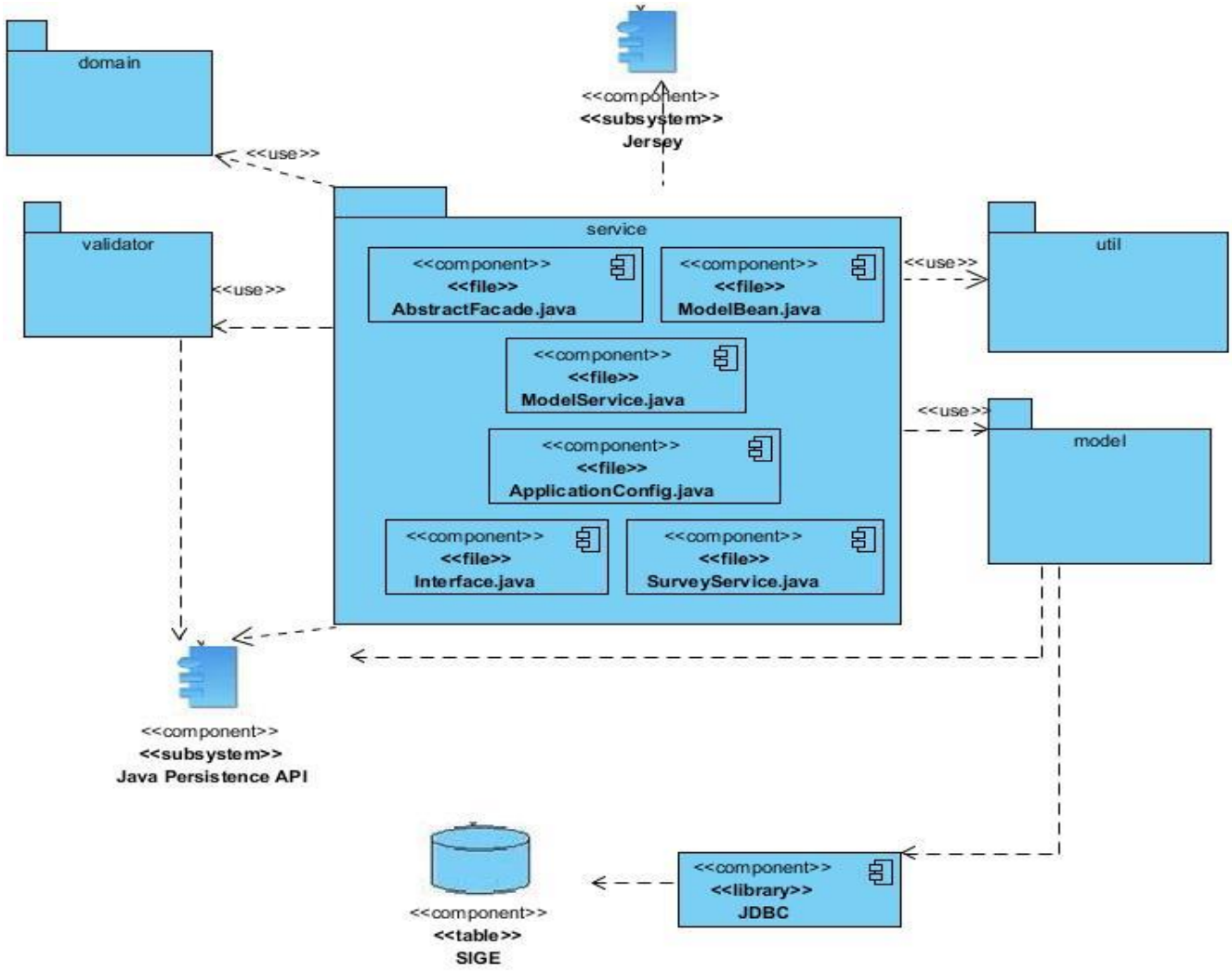


Figura 12: diagrama de Componentes del caso de uso: "Administrar Formulario". Vista General.

3.2. Código Fuente

Se puede definir como código fuente de un sistema informático al conjunto de líneas de texto que constituyen las instrucciones que debe seguir la computadora y que deben ser ejecutadas. El código fuente varía según el lenguaje de programación, y debe ser traducido al lenguaje máquina para que sea ejecutado.

3.2.1. Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez, el estándar de codificación debería establecer cómo operar con la base de código existente (Microsoft, 2015).

Para el desarrollo de la Capa de Servicios Web se tuvieron en cuenta los estándares que a continuación se explican:

- CamelCase: es un estándar en el que un nombre se forma de varias palabras que se unen como una sola palabra con la primera letra de cada una de las varias palabras en mayúsculas para que cada palabra que compone el nombre sea de fácil lectura (TechTarget, 2015).
- UpperCamelCase: la primera letra de la nueva palabra es mayúscula, permitiendo que sea fácil de distinguir de un nombre lowerCamelCase. En la Capa de Servicios Web se utiliza para nombrar las clases (Figura 13)

```
public class ModelService extends AbstractFacade<Tbmodelo> {
    @PersistenceContext(unitName = "csw_sigePU")
  private EntityManagerFactory emf = Persistence.createEntityManagerFactory("csw_sigePU");
    private EntityManager em = emf.createEntityManager();
    private EntityTransaction et = em.getTransaction();
```

Figura 13: fragmento de código fuente de la clase: "MedelService".

- lowerCamelCase: la primera letra del primer nombre es minúscula. La ventaja de CamelCase es que en cualquier sistema informático donde las letras en un nombre tienen que ser sin espacios. En la Capa de Servicios se utiliza este estilo para nombrar las variables y los métodos (Figura 14).

```
public int hashCode() {
    int hash = 0;
    hash += (tbmodeloPK != null ? tbmodeloPK.hashCode() : 0);
    return hash;
}
```

Figura 14: fragmento de código del método: "hashCode"

- Número de declaraciones por línea: en la Capa de Servicios Web se declara cada variable en una línea distinta (Figura 15).

```
String[] fecha = fechadeconfeccion.split("-");
int y = Integer.parseInt(fecha[0]);
int m = Integer.parseInt(fecha[1]);
int d = Integer.parseInt(fecha[2]);
```

Figura 15: ejemplo de declaración de variables.

- Sentencias de importación: no existen líneas en blanco entre ellas (Figura 16).

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collection;
```

Figura 16: ejemplo de sentencias de importación.

- Las sentencias if, for y try-catch, tienen en la Capa de Servicios la estructura que se muestra en las siguientes figuras.

```
for (Tbpaginaaspecto item : entity.getTbpaginaaspectoList()) {
    PaginaAspectos t = entity2domainPage(item);
    page.getPaginaaspectos().add(t);

}
```

Figura 17: ejemplo con la estructura de la sentencia For.

```
if (descriptor.getIdtipodemodelo().getIdtipodemodelo() == 4) {
    Tbcentroinformante center = null;
    try {
        center = (Tbcentroinformante);

    } catch (Exception e) {
        response.setException(e);
        if (e.getClass().equals(javax.persistence.NoResultException.class)) {
            response.setError("La unidad de observación especificada no existe en el sistema.");
            return response;
        }
    }
```

Figura 18: ejemplo con la estructura de las sentencias If y Try-Catch.

3.3. Pruebas del software

El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las prueba para la detección de errores. Además son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa (Pressman, 2010).

3.3.1. Niveles de prueba

Los niveles de prueba son diferentes ángulos de verificar y validar en determinados momentos el ciclo de vida del software. En el desarrollo de la fase de pruebas de la Capa de Servicios Web se aplicarán las pruebas que abarcan los siguientes niveles:

- **Desarrollador**: se realiza con el objetivo de detectar errores en la implementación de requerimientos.
- **Aceptación**: se realiza con el objetivo de detectar fallas en la implementación del sistema.
- **Unitario**: se realiza con el objetivo de detectar errores en los datos, lógica, algoritmos.
- **Sistema:** se realiza con el objetivo de verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas.

3.3.2. Tipos de Prueba

A partir de los niveles de pruebas anteriormente explicados se seleccionan los siguientes tipos de prueba para asegurar el correcto funcionamiento de la Capa de Servicios Web.

Pruebas funcionales: se asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Pruebas de aceptación de tipo alfa: determinación por parte del cliente de la aceptación o rechazo del sistema desarrollado. Es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción. Detecta errores en el sistema bajo un ambiente controlado. Se llevan a cabo en el lugar en donde fue desarrollado el software.

Pruebas de seguridad: intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán. La seguridad del sistema debe ser probada en su invulnerabilidad frente a un ataque frontal.

3.3.3. Métodos de Prueba utilizados en la solución

Pruebas de Caja Blanca: la prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba. Se garantiza que se ejercite por lo menos una vez todos los caminos independientes de cada módulo, además que se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa y que se ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2010).

Para llevar a cabo el método de caja blanca se utiliza la técnica del camino básico, la cual permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

Pruebas de Caja Negra: se centran en los requisitos funcionales del software, es decir, permiten obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2010).

Para llevar a cabo el método caja negra se utiliza la técnica partición equivalente la cual divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Además se utiliza el complemento RESTClient del Firefox, el cual permite probar de una manera rápida y fácil si un servicio REST funciona apropiadamente.

3.3.4. Diseño de Caso de Prueba

Los casos de pruebas se realizan con el objetivo de determinar que una funcionalidad ha sido implementada cubriendo las necesidades del cliente. Para cada caso de uso debe estar asociado un caso de prueba que recoja la especificación de ese caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas y describiendo cada variable que recoge el caso de uso en cuestión.

En la siguiente tabla se detallan las variables que se encuentran asociadas al caso de uso “Administrar formulario”:

Tabla 4: tabla de variables para el caso de prueba.

No	Nombre del campo	Clasificación	Valor nulo	Descripción
1	URL	N/A	No.	Entrada que contiene los parámetros para acceder al servicio web solicitado.
2	JSON	N/A	No.	Entrada que tiene una estructura determinada según el servicio web
3	start	Valor entero positivo	No.	Parámetro de la URL. Valor que indica el comienzo de un rango que puede ser opcional.
4	limit	Valor entero positivo	No.	Parámetro de la URL. Valor que indica el final de un rango que puede ser opcional
5	idmodeloestado	Valor entero positivo	No.	Parámetro de la URL. Especifica el estado del modelo que se desea seleccionar.
6	idnummodelo	Valor entero positivo	No.	Parámetro de la URL. Especifica el número del modelo que se desea seleccionar.
7	idsubnummodelo	Valor entero positivo	No.	Parámetro de la URL. Especifica el subnúmero del modelo que se desea seleccionar.
8	idfechadelinformeacumulado	Tipo fecha, tiene el formato: 2015-03-27	No.	Parámetro de la URL. Especifica la fecha del informe acumulado que se desea seleccionar.
9	idcodvariante	Valor entero positivo	No.	Parámetro de la URL. Especifica el código de variante que se desea seleccionar.
10	codcentroinformante	String	No.	Parámetro de la URL. Especifica el código del centro informante.

Matriz de Datos

Escenario: obtener formularios en estado de validación pendientes a archivar

Tabla 5: matriz de datos.

Escenario	Descripción	URL con parámetros del formulario (id de estado y un rango que puede ser opcional)	Respuesta del sistema	Flujo central
EC 1.1 Obtener formulario en estado de validación pendientes a archivar correctamente.	En este escenario se obtiene un formulario en estado de validación pendientes a archivar correctamente.	V http://10.8.106.89:8086/csw_sige/rest/model/models/id;idmodeloestado=5;start;limit	{ "success": true, "total": 0, "items": [] }	Se introduce la URL en el RESTClient del Mozilla o Chrome, posteriormente se intruduce la URL se presiona el botón SEND y para ver la respuesta se accede a la pestaña Response Body (Preview).
EC 1.2 Obtener formulario en estado de validación pendientes a archivar Faltando parámetros de entrada.	En este escenario se obtiene un formulario en estado de validación pendientes a archivar faltando el start del rango.	I (start) http://10.8.106.89:8086/csw_sige/rest/model/models/id;idmodeloestado=5;limit	{ "success": false, "total": 0, items": [], errorMsg": "Verifique la declaración del parámetro start en la url" }	Se introduce la URL en el RESTClient del Mozilla o Chrome, posteriormente se intruduce la URL se presiona el botón SEND y para ver la respuesta se accede a la pestaña Response Body (Preview).

3.3.5. Resultados de las pruebas

Aceptación

Después de terminada la Capa de Servicios Web se le entrega al jefe de proyecto de SIGE el cual comprueba que cumple con todos los requisitos funcionales y no funcionales. Como resultado se obtiene

una carta de aceptación evidenciando la satisfacción del cliente y por tanto la aceptación de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística.

Seguridad

Para la realización de las pruebas de seguridad a la Capa de Servicios Web, primeramente, se identificaron los distintos roles que están definidos en SIGE y sus respectivos niveles de acceso o privilegios definidos. A continuación se muestra el caso de prueba aplicado en las pruebas de seguridad a la Capa de Servicios Web.

Tabla 6: pruebas de seguridad.

Casos de pruebas	
Descripción	**Resultado**
Las primeras pruebas fueron intentos de violación de la seguridad de la Capa de Servicios Web, tratando de acceder con usuarios no existentes o utilizando usuarios reales con contraseñas erróneas.	Estas pruebas de acceso arrojaron resultados favorables a la Capa de Servicios, ya que no se logró acceder con usuarios no registrados previamente ni con falsas contraseñas, demostrando que la Capa de Servicios Web cuenta con un mecanismo de autenticación seguro, garantizando que solamente accedan usuarios autorizados previamente.
Se accede a la Capa de Servicios Web con un usuario específico y se intenta acceder a funciones no permitidas para ese determinado usuario. Los roles utilizados fueron los siguientes: • Administrador: rol que puede ejercer cualquier funcionalidad en SIGE. • Digitador: rol que solo puede acceder a las funcionalidades de digitar plantillas.	Los usuarios registrados tienen acceso solamente a las funcionalidades predefinidas por los administradores del sistema, impidiendo en todo momento que se lleven a cabo funcionalidades por personas no autorizadas.

Caja Negra

Después de realizar las pruebas de caja negra mediante los casos de prueba asociados a cada caso de uso, se probó el correcto funcionamiento de la Capa de Servicios Web, verificando que las respuestas brindadas ya sea en un caso correcto o incorrecto fueran las esperadas. Durante la aplicación del método de caja negra a la Capa de Servicios Web, fueron detectadas 7 no conformidades en 3 iteraciones a las que se les dio seguimiento y fueron eliminadas a medida que se fue avanzando en el proceso de pruebas. A continuación se muestran las no conformidades por cada iteración realizada:

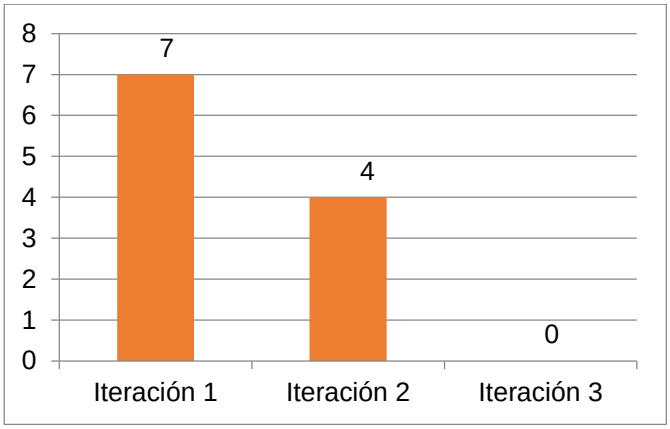


Figura 19: no conformidades.

Conclusiones del capítulo

En el desarrollo del presente capítulo se realizó el modelo de implementación de la Capa de Servicios Web con el propósito de mostrar los componentes del sistema y sus relaciones, a través del diagrama de componentes. Se especificó el uso de los estándares de codificación para lograr un estilo claro y organizado del código durante la fase de implementación. Se realizaron pruebas funcionales, de seguridad y de aceptación para probar el correcto funcionamiento de la Capa de Servicios Web, utilizando el método de caja negra basado en la técnica partición equivalente. Se identificaron 7 no conformidades que fueron resultas posteriormente en 3 iteraciones de la etapa de pruebas.

Conclusiones

Con la realización del presente trabajo de diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado, al lograr desarrollar una Capa de Servicios Web para el Sistema Integrado de Gestión Estadística. Para llegar a este resultado se puede concluir lo siguiente:

- El estudio de los principales aspectos relacionados con los Servicios Web permitió sentar las bases para el posterior desarrollo de la Capa de Servicios Web.
- A partir del análisis y diseño de la Capa de Servicios Web para el Sistema Integrado de Gestión Estadística se obtuvo como resultado los artefactos y diagramas necesarios para guiar el desarrollo de la Capa de Servicios.
- Como resultado de la implementación se obtuvo la Capa de Servicios Web cumpliendo con los 30 requisitos funcionales identificados en la fase de elaboración.
- Las pruebas funcionales, de seguridad y de aceptación permitieron probar el correcto funcionamiento de la Capa de Servicios Web.

Recomendaciones

- Incluir el mecanismo de trazas de SIGE para la Capa de Servicios Web.

Bibliografía

ALLAMARAJU, S. 2010. *RESTful Web Services Cookbook*. 1st Edition. Estados Unidos de América: o´Reilly. ISBN 978-0-596-80168-7.

BELL, D. 2004. UML basics: The component diagram. [en línea]. [Consulta: 18 abril 2015]. Disponible en: https://www.ibm.com/developerworks/rational/library/dec04/bell/.

BERTINO, E. y MARTINO, L. 1995. *Sistemas de bases de datos orientadas a objetos.* [en línea]. S.l.: Díaz de Santos. ISBN 0-201-65356-7. Disponible en: http://books.google.com.cu/books?id=-XohLQySVNMC&pg=PR5&hl=es&source=gbs_selected_pages&cad=2#v=onepage&q&f=false.

BURKE, B. 2013. *RESTful Java with JAX-RS 2.0*. 2nd Edition. Estados Unidos de América: o´Reilly. ISBN 978-1-449-36134-1.

BUSCHMANN, F., MEUNIER, R., ROHNERT, H., SOMMERLAD, P. y STAL, M. 2008. *Pattern-Oriented Software Architecture: a System of Patterns* [en línea]. S.l.: Wiley India Pvt. [Consulta: 16 marzo 2015]. ISBN 8126516119, 9788126516117. Disponible en: https://msdn.microsoft.com/es-es/library/bb972242.aspx.

BUSINESSDICTIONARY 2015. What is framework? definition and meaning. *Framework* [en línea]. [Consulta: 13 marzo 2015]. Disponible en: http://www.businessdictionary.com/definition/framework.html.

BUZZ 2015. Patrones de Casos de Uso. *SG Buzz* [en línea]. [Consulta: 16 marzo 2015]. Disponible en: http://sg.com.mx/content/view/510.

CANA, J.M., 2014. *Modelo de Diseño* [en línea]. 2014. S.l.: universidad Andina del Cusco, Facultad de Ingeniería. [Consulta: 20 marzo 2015]. Disponible en: http://es.scribd.com/doc/241689285/Modelo-de-Diseno-pdf.

CATALANI, E. 2012. Primeros Pasos con REST (Transferencia de estado Representacional) parte 1. *Primeros Pasos con REST* [en línea]. [Consulta: 11 noviembre 2014]. Disponible en: http://exequielc.wordpress.com/2012/10/12/primeros-pasos-con-rest-transferencia-de-estado-representacional-parte-1/.

CONML 2015. FAQs de ConML. *Preguntas habituales acerca de ConML* [en línea]. [Consulta: 12 marzo 2015]. Disponible en: http://www.conml.org/FAQ.aspx.

DARÍO, R. 2014. Metodologías de Desarrollo Ágiles Vs. Metodologías Tradicionales. [en línea]. [Consulta: 3 marzo 2015]. Disponible en: http://rdsoporteymantenimientodepc.blogspot.com/2014/03/metodologias-de-desarrollo-agiles-vs.html.

DEMICHIEL, L. 2013. *JSR 338: JavaTM Persistence API, Version 2.1*. 2.1. USA: oracle America, Inc.

Developing and Securing RESTful Web Services for Oracle WebLogic Server 2014. Estados Unidos de América: oracle.

ECLIPSE.ORG 2012. OpenUP. *Introduction to OpenUp* [en línea]. [Consulta: 4 marzo 2015]. Disponible en: http://epf.eclipse.org/wikis/openup/.

ERICH GAMMA, JOHN VLISSIDES, RALPH JOHNSON y RICHARD HELM 1994. *Design Patterns: elements of Reusable Object-Oriented Software* [en línea]. S.l.: Pearson Education. [Consulta: 16 marzo 2015]. ISBN 0321700694, 9780321700698. Disponible en: https://msdn.microsoft.com/es-es/library/bb972242.aspx.

ERL, T. 2004. *Service-Oriented Architecture: a Field Guide to Integrating XML and Web Services.* S.l.: Prentice Hall.

ESEIDA 2015. Tema 6. Patrones de diseño. *Ingeniería de software* [en línea]. [Consulta: 17 marzo 2015]. Disponible en: https://eseida.wikispaces.com/file/view/Tema+6+-+Patrones+de+Dise%C3%B1o.pdf.

EXES 2015. Características del lenguaje | Curso de Introducción a Java. [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://www.mundojava.net/caracteristicas-del-lenguaje.html?Pg=java_inicial_4_1.html.

EXTREMEPROGRAMMING.ORG 2013. Extreme Programming. [en línea]. [Consulta: 9 mayo 2015]. Disponible en: http://www.extremeprogramming.org/.

FIELDING, R.T. 2000. *Architectural styles and the design of network-based software architectures* [en línea]. Doctoral dissertation. Irvine: university of California. [Consulta: 20 noviembre 2014]. Disponible en: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

GALIANO, G.S. 2015. *Estudio comparativo de servidores multimedia*. Febrero 2015. S.l.: Área de innovación y Desarrollo, S.L. ISBN 978-84-943486-5-5.

GROSSO, A. 2011. Patrones GRASP. *Prácticas de Software* [en línea]. [Consulta: 17 marzo 2015]. Disponible en: http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

HENNEBRUEDER, S. 2013. *Guide to Java Persistence and Hibernate*. 2. S.l.: s.n.

IBM 2014. Introducción a SOA y servicios web. [en línea]. [Consulta: 16 noviembre 2014]. Disponible en: http://www.ibm.com/developerworks/ssa/webservices/newto/service.html.

JENDROCK, E., MARKITO, W., HAASE, K., EVANS, I. y CERVERA, R. 2014. *Java Platform, Enterprise Edition*. S.l.: oracle.

JUAREZ, E.A.O. 2013. Open UP: Metodología Open UP. [en línea]. [Consulta: 4 marzo 2015]. Disponible en: http://openupeaojmp.blogspot.com/2013/09/metodologia-open-up.html.

KALIN, M. 2013. *Java Web Service*. 2nd Edition. United States of America.: o´Reilly. ISBN 978-1-449-36511-0.

LARMAN, C. 2003. *UML y patrones* [en línea]. Segunda Edición. S.l.: Prentice Hall. Disponible en: http://is.ls.fi.upm.es/docencia/is2/documentacion/ModeloDominio.pdf.

LEÓN, R.A.H. y GONZÁLEZ, S.C. 2011. *El proceso de investigación científica*. Raúl G. Torricella Morales. Ciudad de la Habana: universitaria. ISBN 978-959-16-1307-3.

LETELIER, P. y PENADÉS, C. 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *26* [en línea], vol. 05. [Consulta: 27 mayo 2015]. ISSN 1666-1680. Disponible en: http://www.cyta.com.ar/ta0502/v5n2a1.htm.

MARIN, M.D.A. 2008. Definición de lenguaje de programación. Tipos. Ejemplos. *Cátedra de programación* [en línea]. [Consulta: 12 marzo 2015]. Disponible en: http://catedraprogramacion.forosactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.

MEDINA, A. 2015. Metodologia de desarrollo de software. *Academia* [en línea]. [Consulta: 3 marzo 2015]. Disponible en: http://www.academia.edu/4984909/Metodologia_de_desarrollo_de_software.

MICROSOFT 2015. Revisiones de código y estándares de codificación. [en línea]. [Consulta: 21 abril 2015]. Disponible en: https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

NAVARRO, R., 2010. *REST vs Web Services* [en línea]. 2010. S.l.: eLP-DSIC-UPV. Disponible en: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf.

ORACLE 2014. *Understanding EclipseLink*. S.l.: The Eclipse Foundation.

ORACLE 2015a. Introduction to the Java Persistence API. [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html.

ORACLE 2015b. Java SE 7 Features and Enhancements. [en línea]. [Consulta: 21 abril 2015]. Disponible en: http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html.

ORACLE 2015c. Jersey. *RESTful Web Services in Java.* [en línea]. [Consulta: 19 marzo 2015]. Disponible en: https://jersey.java.net/.

ORACLE 2015d. NetBeans. *The Source for Java Technology Collaboration* [en línea]. [Consulta: 19 marzo 2015]. Disponible en: https://community.java.net/community/netbeans.

PERICAS-GEERTSEN, S. y POTOCIAR, M. 2013. *JAX-RS: Java™ API for RESTful Web Services*. 2.0. 500 Oracle Parkway, Redwood Shores, CA 94065 USA.: oracle.

POSTGRESQL.ORG 2003. PostgreSQL: File Browser. *pgadmin3* [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://www.postgresql.org/ftp/pgadmin3/.

POSTGRESQL.ORG 2013. PostgreSQL. [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://www.postgresql.org/docs/9.3/static/release-9-3.html.

POSTGRESQL.ORG.ES 2010. Sobre PostgreSQL. [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

PRESSMAN, R.S. 2010. *Ingeniería del Software, un enfoque práctico*. 5. S.l.: Mc Graw Hill.

PUTTE, G.V. de, JANA, J., KEEN, M., KONDEPUDI, S., MASCARENHAS, R., OGIRALA, S., RUDROF, D., SULLIVAN, K. y SWITHINBANK, P. 2004. *Using Web Services for Business Integration*. IBM. S.l.: iBM. ISBN SG24-6583-00.

RAMÍREZ, I. 2015. NetBeans IDE. *Aplicación para desarrolo en Java* [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://netbeans-ide.softonic.com/.

REYNOSO, C. y KICCILLOF, N., 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft* [en línea]. marzo 2004. S.l.: universidad de Buenos Aires. [Consulta: 20 marzo 2015]. Disponible en: http://www.willydev.net/descargas/prev/Estiloypatron.pdf.

ROUSE, M. 2015. What is Unified Modeling Language (Unified Modeling Language)? [en línea]. [Consulta: 12 marzo 2015]. Disponible en: http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language.

SAMPIERI, R.H., COLLADO, C.F. y LUCIO, P.B. 2006. *Metodología de la investigación*. Cuarta. México: infagon web, S.A. de C.V. ISBN 970-10-5753-8.

SCRUM.ORG 2015. What is Scrum? [en línea]. [Consulta: 10 mayo 2015]. Disponible en: https://www.scrum.org/resources/what-is-scrum.

SOFT112 2014. Visual Paradigm for UML Standard 8.0. [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://visual-paradigm-for-uml-standard.soft112.com/.

SOFTWAREENGINEERINSIDER.COM 2014. What Are Software Engineering CASE Tools? *Software Engineer Insider* [en línea]. [Consulta: 12 marzo 2015]. Disponible en: http://www.softwareengineerinsider.com/articles/case-tools.html.

SOMERVILLE, I. 2007. *Software Engineering* [en línea]. 8. S.l.: addison-Wesley. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=9269&subdir=/Sommerville_8va_edicion.

SPARXSYSTEMS 2013. Sparx Systems - Tutorial UML 2. *Diagrama de Despliegue* [en línea]. [Consulta: 17 marzo 2015]. Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

STENBERG, J. 2013. RESTful Web Services Framework Jersey 2.0 Released, Implementing JAX-RS 2.0 Specification. [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://www.infoq.com/news/2013/06/Jersey_2_0_released.

TECHTARGET 2015. What is CamelCase? [en línea]. [Consulta: 21 abril 2015]. Disponible en: http://searchsoa.techtarget.com/definition/CamelCase.

TOMCAT.APACHE.ORG 2015a. Apache Tomcat 8. [en línea]. [Consulta: 21 abril 2015]. Disponible en: https://tomcat.apache.org/tomcat-8.0-doc/index.html.

TOMCAT.APACHE.ORG 2015b. Apache Tomcat. [en línea]. [Consulta: 21 abril 2015]. Disponible en: http://tomcat.apache.org/.

USERS.DCC 2015. Tutorial de UML. *Casos de Uso* [en línea]. [Consulta: 13 marzo 2015]. Disponible en: http://users.dcc.uchile.cl/~psalinas/uml/casosuso.html.

W3C CONSORTIUM 2004. Web Services Architecture. *Web Services Architecture* [en línea]. [Consulta: 20 noviembre 2014]. Disponible en: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#whatis.

W3C.ES 2014. Guía Breve de Servicios Web. [en línea]. [Consulta: 19 noviembre 2014]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

WANG, J. 2013. *JSR-000206 Java™API for XML Processing («Specification»)*. 1.6. Redwood Shores, CA 94065 USA: oracle America, Inc.

WEBBER, J., PARASTATIDIS, S. y ROBINSON, I. 2010. *REST in practice*. 1st Edition. Estados Unidos de América: o´Reilly. ISBN 978-0-596-80582-1.

WELICKI, L. 2015. Patrones y Antipatrones: una Introducción - Parte II. [en línea]. [Consulta: 17 marzo 2015]. Disponible en: https://msdn.microsoft.com/es-es/library/bb972251.aspx.

WIKIBOOKS.ORG 2013. Java Persistence/What is new in JPA 2.1? [en línea]. [Consulta: 19 marzo 2015]. Disponible en: http://en.wikibooks.org/wiki/Java_Persistence/What_is_new_in_JPA_2.1%3F.

ZAYAS, C.Á. de 1995. *Metodología de la investigación científica*. Santiago de cuba: Centro de estudios de educación superior.

Anexos

Anexo 1: Cuestionario de la entrevista

Descripción: cuestionario de la entrevista aplicado para recoger la información tanto primaria como secundaria para el desarrollo de la Capa de Servicios para el Sistema Integrado de Gestión estadística. La entrevista se le realizó al ingeniero: Héctor Luis Reyes Zaldívar. Se utilizan las preguntas directas.

1. ¿Por qué surge la propuesta de desarrollar una Capa de Servicios Web para el Sistema Integrado de Gestión Estadística?

2. ¿A qué escenarios se adaptaría la Capa de Servicios una vez desarrollada?

3. ¿La utilizarían todos los organismos del país o solo los organismos asociados a la ONEI?

4. En un organismo se desempeñan muchos roles, ¿qué rol específico utilizaría la Capa de Servicios?, ¿o la utilizarían todos los roles?

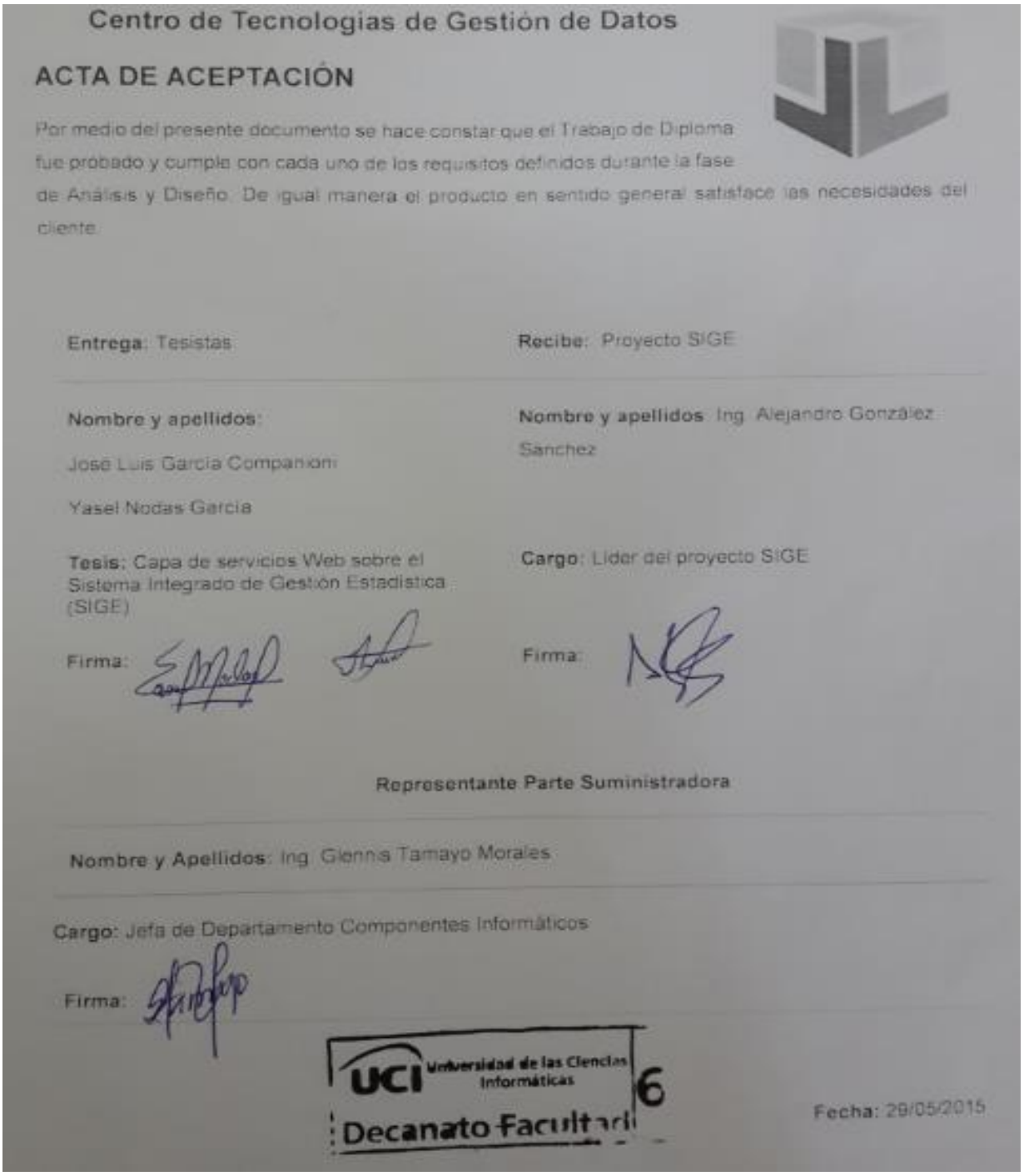
5. ¿Qué objetivo se persigue con el desarrollo de la Capa de Servicios?

6. SIGE es un sistema muy amplio. ¿Todos los procesos que tiene actualmente SIGE serán llevados a la Capa de Servicios Web?, en caso que no sean todos atendiendo a una propuesta inicial ¿cuáles serían los procesos de SIGE que serán llevados a la Capa de Servicios?

7. ¿Qué es lo que la Capa de Servicios debe hacer?, ¿cuáles son los requerimientos funcionales?

8. Una vez seleccionados los requerimientos funcionales, ¿cómo se podría distribuir a su modo de ver en casos de uso?

Anexo 2: Carta de aceptación de la Capa de Servicios Web



Centro de Tecnologías de Gestión de Datos

ACTA DE ACEPTACIÓN

Por medio del presente documento se hace constar que el Trabajo de Diploma fue probado y cumple con cada uno de los requisitos definidos durante la fase de Análisis y Diseño. De igual manera el producto en sentido general satisface las necesidades del cliente.

Entrega: Tesistas

Nombre y apellidos:

José Luis García Companioni

Yasel Nodas García

Tesis: Capa de servicios Web sobre el Sistema Integrado de Gestión Estadística (SIGE)

Firma:

Recibe: Proyecto SIGE

Nombre y apellidos: Ing. Alejandro González Sánchez

Cargo: Líder del proyecto SIGE

Firma:

Representante Parte Suministradora

Nombre y Apellidos: Ing. Glennis Tamayo Morales

Cargo: Jefa de Departamento Componentes Informáticos

Firma:

UCI Universidad de las Ciencias Informáticas

Decanato Facultad 6

Fecha: 29/05/2015

Anexo 3: Interfaz principal del RESTClient