Universidad de las Ciencias Informáticas
FACULTAD 6

Título:

API de servicios web en GDR 2.0 para la gestión de reportes desde diversos entornos de diseño.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Gabriel Alejandro Valcarcel Fernández

Frank Ramírez Santana

Tutores:

MsC. Reynaldo Alvarez Luna

Ing. Anaibis Alvarez Morales

Ing. Félix González Martínez

La Habana, Junio de 2014

‘‘Año 56 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Gabriel Alejandro Valcarcel Fernández

_____
Firma del Autor

MsC. Reynaldo Alvarez Luna

_____
Firma del Tutor

Frank Ramírez Santana

_____
Firma del Autor

Ing. Anaibis Alvarez Morales

_____
Firma del Tutor

Ing. Félix González Martínez

_____
Firma del Tutor

DATOS DE CONTACTO

Autores:

Gabriel Alejandro Valcarcel Fernández

Universidad de las Ciencias Informáticas

E-mail: gavarcarcel@estudiantes.uci.cu

Frank Ramírez Santana

Universidad de las Ciencias Informáticas

E-mail: framirez@estudiantes.uci.cu

Tutores:

MsC. Reynaldo Alvarez Luna

Universidad de las Ciencias Informáticas (UCI)

E-mail: rluna@uci.cu

Ing. Anaibis Alvarez Morales

Centro de Tecnologías de Gestión de Datos (DATEC)

Universidad de las Ciencias Informáticas (UCI)

E-mail: aamorales@uci.cu

Ing. Félix González Martínez

Centro de Tecnologías de Gestión de Datos (DATEC)

Universidad de las Ciencias Informáticas (UCI)

E-mail: felix@uci.cu

AGRADECIMIENTOS

Agradecimientos de Gabriel:

A mis padres y abuelos, quien con su confianza y colaboración se convirtieron en la inspiración y el motor para superar las dificultades y afrontar con entereza los retos que la carrera nos ha planteado.

A mi tía por su apoyo moral, material y espiritual en todo momento.

A mi novia que ha estado en cada momento, con amor, dedicación, alentándome en la culminación de este trabajo.

A mis tíos por su apoyo y cariño.

A los tutores MsC. Reynaldo Alvarez Luna, Ing. Anaibis Alvarez Morales y Ing. Félix González Martínez, por transmitirme sus conocimientos y experiencias en cada momento que solicité su cooperación, gracias por su profesionalidad y entrega.

A nuestros docentes, por sus conocimientos, paciencia y habilidades, que me permitieron formarme como ingeniero.

A mis amigos y compañeros de estudios por su compañía, respaldo y apoyo, a lo largo de estos cinco años, que nos han acompañado en la realización de nuestros sueños.

Y a todos aquellos que de una forma u otra han participado en mi formación y culminación de este trabajo, que han confiado en mí, nunca los defraudaré.

Agradecimientos de Frank:

En estos cinco años de estudio, muchas han sido las personas que de una forma u otra contribuyeron a lograr este resultado por lo que creo que es el momento de agradecerles.

Primeramente quiero agradecer a mi mamá Maria y a mi papá Frank por su amor, cariño, apoyo y por la crianza que me dieron, y sobre todo por su sacrificio, sin ellos nada de esto hubiese sido posible.

Agradecimientos

A mi compañero de tesis Gabriel por su trabajo, su entrega y esmero en este trabajo pues de otra manera no hubiésemos podido alcanzar esta meta.

A mis tutores Félix, Anaibis y Luna por estar siempre pendiente de todos los detalles y guiándonos en todo el desarrollo de este trabajo.

A mis compañeros de aula por ayudarme en todo momento que lo molesté para que me explicara alguna de mis tantas dudas.

DEDICATORIA

Dedicatoria de Gabriel:

--A mis padres, que han sido mis mayores inspiradores.

--A mi tía, por ser especial.

--A mi novia, luz de mi vida.

--A mis tíos, por tenerlos a mi lado.

Dedicatoria de Frank:

A mi mamá y a mi papá por ser mis ejemplos y hacer de mi lo que soy hoy.

A mis hermanos por quererme tanto, espero ser un ejemplo para ellos.

A mis abuelos, a mi primo y a mis amistades.

A toda mi familia por confiar y esperar siempre lo mejor de mí.

RESUMEN

La investigación surge en el departamento Integración de Soluciones perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC) de la Universidad de las Ciencias Informáticas (UCI), el cual promueve el desarrollo de soluciones que permiten el análisis, tratamiento y manipulación de la información capturada por un sistema informático. El Generador Dinámico de Reportes (GDR) es una de estas soluciones y actualmente se desarrolla la versión 2.0. El sistema es dependiente de la conectividad de las estaciones de trabajo, esta característica conlleva a un conjunto de dificultades en el diseño de los reportes, tales como el incumplimiento con los cronogramas de trabajo pactados con el cliente, provocado por una depreciación del tiempo laboral, siendo este un indicador donde la conexión de red es un factor a tener en cuenta.

El presente trabajo expone una solución para el empleo asíncrono del GDR 2.0, la cual se ha implementado a partir de almacenar localmente los reportes, que se van creando desde las estaciones de trabajo al invocar un servicio de GDR 2.0. Se implementa una API de servicios web en el GDR 2.0 y una extensión en el iReport que utiliza la estación de trabajo. A partir de esta solución un usuario podrá obtener reportes de modo local aunque estén limitados los servicios de red entre la estación de trabajo y el servidor a partir de los datos de reportes anteriores almacenados en la estación. Esta posibilidad permite utilizar GDR 2.0 en entornos donde esto no era posible.

PALABRAS CLAVE

Asíncrono, conexión, gdr, servicios web.

ABSTRACT

The Center of Data Management Technologies at the University of Computer Sciences promotes the development of software solutions that enable the analysis, processing and manipulation of information captured by a computer system. Dynamic Report Generator (GDR) is one of these solutions and currently the version 2.0 is developed. The system is dependent on the connectivity of workstations, this feature leads to a set of challenges in the design of the reports, such as failure to fulfill work schedules agreed with the client, caused by a depreciation of labor time, this being an indicator where the network connection is a factor to consider.

This paper presents a solution for asynchronous use of GDR 2.0, which has been implemented storing the reports locally, which are created from workstations when they invoke a service of GDR 2.0. Web Services API is implemented in the GDR 2.0 and an extension in iReport for the workstations. From this solution a user can obtain reports locally from the data stored in the station even though network services are limited between the workstation and the server. This capability allows use GDR 2.0 in environments where was not possible.

KEYWORKS

Asynchronous, connection, gdr, web services.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con el desarrollo acelerado de las Tecnologías de la Información y las Comunicaciones (TIC) la informatización de las sociedades se ha incrementado a nivel mundial (1). Esto ha permitido un aumento en la capacidad de generación y almacenamiento de la información, la cual adquiere mayor importancia como un recurso estratégico en la toma de decisiones. Las TIC agrupan un conjunto de tecnologías que permiten la gestión de la información sobre las infraestructuras de telecomunicaciones. Como parte de esta gestión de la información existen diversos mecanismos que posibilitan transferir datos entre dispositivos, de modo directo o a través de redes digitales.

Cuando la trasferencia de datos se produce sobre redes digitales se pueden utilizar los servicios web. Estos constituyen un conjunto de aplicaciones o tecnologías con capacidad para interoperar en la web, proporcionando mecanismos de comunicación entre diferentes aplicaciones.

Para lograr la comunicación entre estas aplicaciones se puede hacer mediante el uso de una Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés de ***Application Programming Interface***), la cual se utiliza de llave de acceso a funciones que permiten utilizar un servicio web provisto por un tercero, dentro de una aplicación web propia, de manera segura. Además de permitir interacciones entre diferentes sitios web, las API posibilitan acceder a los datos o procesar transacciones en la aplicación principal. Para almacenar grandes volúmenes de información se utilizan estructuras de almacenamiento conocidas como Base de Datos (BD), una de las técnicas que ha permitido el incremento de su uso es el empleo de servicios web para el acceso a los datos.

Las personas pueden tomar decisiones a partir del análisis de la información contenida en las BD. El uso de estas estructuras de almacenamiento puede darse por parte de los usuarios aunque tengan diferentes propósitos. El manejo de grandes volúmenes de información de manera conjunta y concurrente se torna muy complejo, de ahí la utilización de los reportes que permiten satisfacer necesidades de información para disimiles usuarios. Cada reporte puede tener un formato determinado para la visualización de los datos de manera más organizada, precisa y fácil de interpretar. Los elementos que se utilizan pueden ser gráficos, diagramas, listas o tablas de contenidos.

Existen diversas herramientas a nivel internacional que se especializan en la gestión de reportes, conocidos como gestores de reportes. Estas herramientas permiten crear mediante el empleo de una interfaz visual

los diseños de los reportes, abstrayendo al usuario del conocimiento profundo para su confección. En la actualidad existen numerosos sistemas para la gestión de reportes, entre los que se encuentran Crystal Report (2), Pentaho Reporting (3) e iReport (4); con varias características como: propiedades de las fuentes de datos, licencia, plataforma y formatos de salida permitiendo la visualización de los reportes.

Cuba, en los avances de la informática y del proceso de informatización de la sociedad se ha involucrado en la creación de sistemas informáticos que permiten generar reportes, entre los que se encuentran: Sistema de Rehabilitación Integral, Generador de Reportes de Pruebas Hemodinámicas para El Diagnóstico de Enfermedades Vasculares Periféricas (5).

La Universidad de las Ciencias Informáticas (UCI) participa de modo protagónico en el desarrollo de proyectos de software que incluyen módulos de reportes (6). El Departamento Integración de Soluciones, perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC) ha estado a cargo de desarrollar sistemas informáticos en pos de satisfacer las necesidades en la gestión de información de organismos y empresas nacionales o extranjeras. Como resultado de la labor de varios años se han obtenido productos y activos de software como el Generador Dinámico de Reportes (GDR). Este software permite el tratamiento del ciclo de vida de los reportes, además del empleo de varias fuentes de datos y facilidades de uso en entornos de trabajo.

Resulta común y creciente el empleo de aplicaciones web en las entidades modernas, la coexistencia de varias de ellas unidas al empleo masivo por múltiples usuarios, provocan dificultades en los servicios que transitan por la red. GDR en su versión actual, la 2.0, se ejecuta en ambiente web, donde la aplicación está ubicada en un servidor y los usuarios se conectan a través de una red, mediante el empleo de un navegador para invocar una instancia de la aplicación.

Además el sistema es dependiente de la conectividad entre el servidor donde se publica el sistema GDR y las estaciones de trabajo que lo usan, sin embargo estas características conllevan a un conjunto de resultados no positivos en el desarrollo de los reportes, destacándose el incumplimiento con los cronogramas de trabajo pactados con el cliente, provocado por una depreciación del tiempo laboral, siendo este un indicador donde la conexión de red es un factor a tener en cuenta, también inciden elementos externos como energía eléctrica y tráfico en la red.

Teniendo en cuenta lo antes expuesto se plantea como **problema de la investigación:** La imposibilidad de gestionar reportes con GDR cuando existen dificultades en la conectividad entre la estación del usuario y el servidor. (Ver Fig. 1*: Diagrama de red sobre el despliegue de GDR 2.0*)

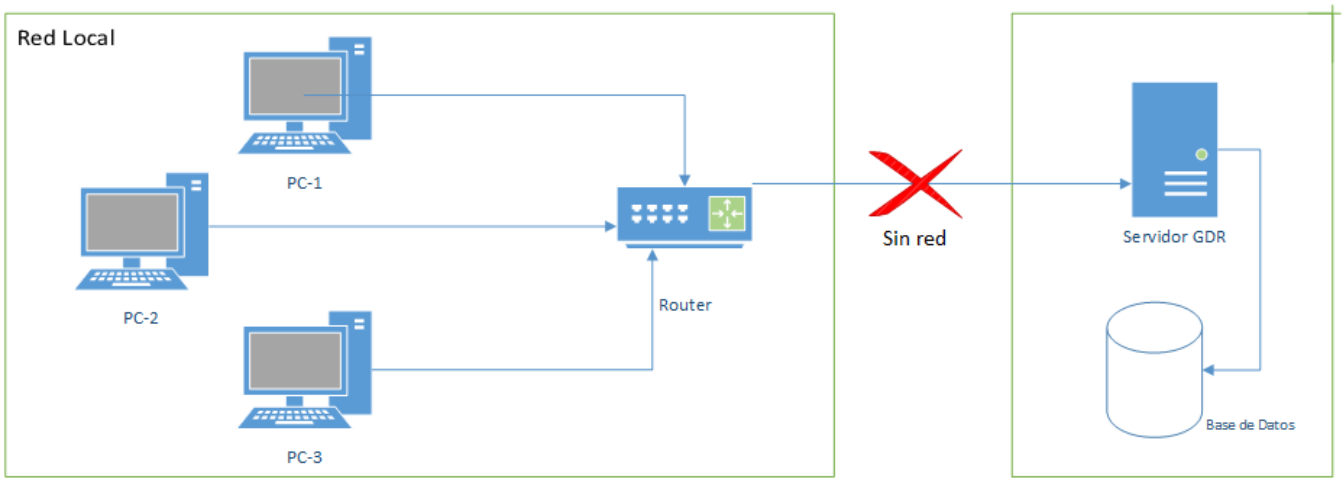


Fig. 1: Diagrama de red sobre el despliegue de GDR 2.0

El presente trabajo tiene como **objeto de estudio:** Sistemas informáticos para la gestión de reportes, enmarcado en el **campo de acción:** Gestión de reportes en GDR 2.0 mediante la publicación de servicios web.

Para resolver el problema anteriormente planteado surge como **objetivo general:** Desarrollar una API de servicios web en GDR 2.0 para la gestión asíncrona de reportes desde diversos entornos de diseño.
A partir del análisis del objetivo general se derivan los siguientes **objetivos específicos:**

✓ Analizar los conceptos, metodología, tecnologías y herramientas afines para el desarrollo de una API de servicios web en GDR 2.0 para la gestión de reportes.

✓ Realizar el análisis y diseño de la API de servicios web.

✓ Realizar la implementación de la API de servicios web.

✓ Implementar una extensión para un entorno de diseño que valide la funcionalidad de la API de servicios web.

Las **preguntas de investigación** son las siguientes:

- ✓ ¿Cómo garantizar la generación de reportes para un cliente que utiliza GDR 2.0 en entornos de red inestables?
- ✓ ¿Cómo validar el funcionamiento de la API de servicios web implementada?

Para el cumplimiento de los objetivos antes expuestos, se definieron las siguientes **tareas de la investigación:**

✓ Realización del estudio bibliográfico de las herramientas de diseño de reportes para identificar sus principales características.

✓ Identificación de los principios y estilos arquitectónicos de GDR 2.0 para el desarrollo de la API de servicios web.

✓ Fundamentación de la selección de la metodología, las herramientas y las tecnologías a emplear en el desarrollo de la API de servicios web en GDR 2.0 para la gestión de reportes.

✓ Descripción de las clases conceptuales y asociaciones candidatas para el contexto de la API de servicios web en GDR 2.0.

✓ Definición de los requisitos funcionales y no funcionales de la API de servicios web en GDR 2.0, para establecer lo que debe hacer y bajo qué circunstancias debe hacerlo.

✓ Descripción del comportamiento detallado del sistema en función de los requisitos funcionales definidos para la API de servicios web en GDR 2.0.

✓ Elaboración del modelo de diseño de la API de servicios web en GDR 2.0 para detallar las clases software del diseño que definen el comportamiento del sistema.

✓ Elaboración del modelo de implementación de la API de servicios web en GDR 2.0 para describir los elementos del modelo de diseño y la estructuración de estos de acuerdo al entorno de desarrollo.

✓ Implementación de una extensión como validación para el consumo de la API de servicios web.

✓ Realización de pruebas funcionales a la solución implementada.

Para llevar a cabo las tareas planteadas se utiliza métodos de la investigación científica.

El **método teórico** utilizado para cumplir con las tareas a desarrollar es:

✓ **Modelación:** El proceso de modelación se implementa a partir de seleccionar un lenguaje de modelado, una metodología para el análisis y desarrollo además el empleo de herramientas CASE soportadas en sistemas informáticos, lo cual permitirá obtener los modelos de la solución que se va a implementar.

El **método empírico** que se utiliza para obtener información sobre el objeto de estudio es:

✓ **Entrevista:** Se utiliza este método como base para la obtención de la información, comprender y precisar bien el problema a resolver por parte de los desarrolladores. (Ver Anexo 1*:* Entrevista al cliente.)

El presente trabajo se ha estructurado en tres capítulos: en el primer capítulo “capítulo 1: Fundamento teórico de la API de servicios web en GDR 2.0” se presentan los conceptos relacionados con el problema a resolver. Se realiza un análisis de las herramientas, metodología y tecnologías que servirán de apoyo a la investigación y que se utilizarán para dar solución al problema planteado. En el segundo capítulo “capítulo 2: Análisis y diseño de la API de servicios web en GDR 2.0”, se definen las funcionalidades a realizar en el sistema y la especificación de las mismas. Se muestra la estructura del sistema a desarrollar a través de una representación visual de las clases conceptuales en el Modelo de Dominio. En el tercer capítulo “capítulo 3: Implementación y validación de la API de servicios web en GDR 2.0” se presenta el modelo de implementación a través del diagrama de componentes. Se describen las pruebas realizadas a la aplicación y los resultados de las mismas verificando su funcionamiento.

CAPÍTULO 1: FUNDAMENTO TEÓRICO DE LA API DE SERVICIOS WEB EN GDR 2.0

Introducción

En el presente capítulo se muestran los conceptos relacionados con el problema de la investigación. Se realiza un estudio bibliográfico de las aplicaciones existentes para generar reportes y su utilidad en la sociedad, además se aborda el tema de los servicios web mediante el estudio de los principales estándares que estos aplican para garantizar la comunicación entre diferentes aplicaciones, también se describen las herramientas, metodología y tecnologías que servirán de apoyo a la investigación para dar solución al problema planteado.

1.1 Conceptos asociados al dominio del problema

En el presente epígrafe se conceptualiza un conjunto de elementos relacionados con el objeto de estudio de la investigación, los cuales se describen a continuación para una mejor comprensión del trabajo en general.

Reporte

En el glosario de términos del proyecto GDR se define un reporte como: *"Un documento generado por el sistema, que presenta de manera estructurada y/o resumida, datos relevantes guardados o generados por la misma aplicación de tal manera que se vuelvan útiles para los fines propuestos."* (7)

Servicios web

Ignacio García Valcárcel y Eduardo Munilla Calvo plantean que: *"(…) los servicios web significan la evolución de la informática distribuida, cuyo principio arquitectónico es permitir que aplicaciones de un entorno se conecten y compartan datos y contenido con aplicaciones de otro entorno distinto."* (8)

Red informática

Conjunto de ordenadores y dispositivos electrónicos conectados entre sí cuya finalidad es compartir recursos, información y servicios. (9)

1.2 API

Francis y Dominique, en su libro *"La alquimia de las multitudes"*, la definen como: *"(…) puertas que los creadores de los programas abren de forma voluntaria para permitir que otros creadores puedan apropiarse de los elementos que interesen de dichos programas y añadir sus propios servicios."* (10)

Funcionamiento de las API

Una API de programación representa una interfaz de comunicación entre componentes de software, se trata del conjunto de llamadas a determinadas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, entre las capas inferiores y las superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general. De esta forma, los programadores se benefician de las ventajas haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. (11)

Ventajas de una API

- Hace más fácil el desarrollo de un programa, ya que debe proveer todos los conceptos para construirlo.
- Está diseñado especialmente para los programadores, ya que garantiza que todos los programas que utilizan API, tendrán interfaces similares. Asimismo, esto le facilita al usuario aprender la lógica de nuevos programas.
- Cuando se realiza una solicitud, el servidor llamará a la API, brindando la ventaja de disponer de una mayor cantidad de servicios. (11)

1.3 Generadores de reportes

Los reportes constituyen una ayuda para los usuarios de los sistemas en el momento de llevar el seguimiento del negocio, permitiendo a los directivos tener los indicadores que muestran los resultados de determinadas acciones y ayudándolos a tomar decisiones sobre el futuro de la empresa. Para facilitar la gestión de la información del organismo, organizaciones o empresas se utilizan generadores de reportes. Estas herramientas permiten crear desde simples listados hasta complejos reportes de datos, ofreciendo a los usuarios la información deseada, la cual es obtenida a través de conexiones realizadas a las fuentes de datos. Existen diversos sistemas generadores de reportes tales como:

Pentaho Reporting

Pentaho Reporting es un conjunto de herramientas de reporte de código abierto que permiten la creación de informes relacionales y análisis de una amplia gama de fuentes de datos. (12) Es una herramienta fácil de utilizar y flexible para que los diseñadores de reportes creen de forma ágil los reportes de acuerdo a los estándares y necesidades del cliente. Permite la distribución de los resultados de los análisis en varios formatos como PDF, HTML, Excel y en texto plano. Existen tres productos con diferentes enfoques Pentaho Report Designer, Pentaho Report Design Wizard y Web ad-hoc reporting. (13)

Crystal Reports

Es una aplicación de inteligencia empresarial utilizada para diseñar y generar reportes desde una base de datos. Varias aplicaciones, como Microsoft Visual Studio, incluyen una versión de Crystal Reports como una herramienta de propósito para generar los informes y reportes. Es compatible con sistemas operativos como: Microsoft Windows XP con Service Pack (SP) 2, Windows Server 2003 con SP 1 o superior. Es un software propietario que integra estrechamente capacidades de diseño, modificación y visualización en aplicaciones .Net y Java. Permite exportar los informes a diferentes formatos (Crystal Reports (rpt), Microsoft Excel, html, Microsoft Word, PDF). Sus informes pueden ser almacenados en plataformas Windows y permite publicar informes Crystal en un servidor web. (14)

Generador Dinámico de Reportes (GDR)

El Generador Dinámico de Reportes es un sistema especializado en la generación de reportes. Es una herramienta multiplataforma, que permite al usuario el diseño de reportes personalizados. Es un sistema basado en tecnologías web. Fue desarrollado con el lenguaje de programación PHP, en conjunto con el marco de trabajo Symfony. Posee una interfaz gráfica desarrollada con el marco de trabajo de Ext JS. El sistema permite conectarse a diversas fuentes de datos entre las que destacan gestores de bases de datos como: PostgreSQL, MySQL, SQLite, Oracle y Microsoft SQL Server. Mediante esta herramienta los reportes se pueden exportar a los formatos de salida HTML, PDF, EXCEL y CSV (6).

iReport

iReport es una herramienta visual de diseño para JasperReports. Es un programa de software libre, escrito en las secuencias de comandos de código de Java, que se distribuye sin costo y por tiempo ilimitado. Permite a los usuarios editar visualmente cualquier tipo de informe complejo con gráficas, imágenes y

subinformes. Permite la visualización de los informes en varios formatos, como PDF, XLS, RTF, CSV, XML o HTML.

Tabla comparativa entre las diferentes herramientas para la generación de reportes:

Tabla 1: Comparación de las herramientas

Herramientas	Criterios de comparación				
	Licencia	Plataforma	Compatibilidad con JasperReport	Publicar reportes mediante servicios	API
Pentaho Reporting	Libre	Escritorio Web	No	No	No
Crystal Reports	Privada	Escritorio Web	No	Si	No
GDR	Libre	Web	Si	No	No
iReport	Libre	Escritorio	Si	No	No

Se realizó el estudio de un conjunto de herramientas que permiten el desarrollo del ciclo de vida de los reportes con el objetivo de seleccionar las que presenten una API de servicios web para complementar con prácticas y estilos ya probados la API de servicios web para GDR 2.0, permitiendo realizar varias operaciones con reportes en ausencia de red así como facilitar la migración de estos mediante un proceso de compatibilización.

Entre las herramientas de gestión de reportes estudiadas se encuentran: Pentaho Reporting, Crystal Reports e iReport, a pesar de que estas herramientas no presentan una API para los servicios que permita identificar elementos de interfaces y buenas prácticas, se selecciona el iReport para validar el funcionamiento de la API de servicios web de GDR 2.0, pues tiene características positivas que potencian el desarrollo de la tesis debido a que es compatible con el Generador Dinámico de Reportes (GDR)

desarrollado en el Departamento Integración de Soluciones, permite el diseño del 98 % de los componentes de la librería de JasperReport que presenta licencia libre, además de cumplir con las políticas de migración a software libre de Cuba organizadas por el centro de Servicios Integrales de Migración, Asesoría y Soporte (SIMAYS). Las restantes herramientas resaltan elementos negativos como el empleo de licencias de altos costos, servidores con arquitecturas diferentes y poco conocimiento de estas; así como las altas barreras legales y financieras para extender una pieza de software propietario y adaptarla a las necesidades particulares de un problema específico. Sin embargo el sistema Crystal Reports tiene ventajas que constituyen un aporte sustancial ya que publica reportes en otros servidores, para la realización de esta funcionalidad emplean un conjunto de mecanismos como: Abrir archivos .rpt directamente desde el escritorio, aunque se esté trabajando sin conexión, se beneficia de la autenticación para garantizar la seguridad de la información.

1.4 Metodología y herramientas empleadas en la solución

1.4.1 Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y soporte documental que permite la creación de software. Sirve de guía en el proceso de desarrollo de software y se agrupan en dos grandes grupos, las tradicionales y las ágiles. Las tradicionales se enfocan en el plan del proyecto, estableciendo rigurosamente las actividades implicadas, los artefactos que se deben producir, además de las herramientas y notaciones que se utilizarán. Por otra parte las metodologías ágiles se centran en satisfacer las necesidades del cliente, dándole mayor valor al individuo. (15)

Para caracterizar el API de servicios web en GDR 2.0 y estimar cuán ágil o robusto debía ser el enfoque a utilizar, se emplea el modelo de Boehm & Turner. Este método plantea cinco criterios mediante los que se estará valorando el proyecto; estos son: tamaño del equipo de desarrollo, criticidad del producto, dinamismo de los cambios, cultura del equipo y experiencia del personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta para seleccionar uno u otro enfoque. (16) Después de realizar el estudio en el proyecto se determinó darle un enfoque ágil.

OpenUp

OpenUp es una metodología ágil que aplica un enfoque iterativo e incremental dirigida a la gestión y desarrollo de proyectos de software. Se centra en una arquitectura temprana para reducir al mínimo los

riesgos y organizar el desarrollo del software. Mantiene las características esenciales del Proceso Unificado de Rational (RUP, por sus siglas en inglés de ***Rational Unified Process***), donde se incluyen el desarrollo incremental, el empleo de casos de uso y escenarios, el diseño basado en la arquitectura; permitiendo el desarrollo de aplicaciones mediantes la generación de artefactos ligeros apropiados, utilizando el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés de ***Unified Modeling Language***) e incrementando así las probabilidades de éxito en función del costo, tiempo y alcance. (17)

OpenUp se conforma por cuatro fases:

1. **Concepción:** Esta fase tiene como objetivo capturar las necesidades de los involucrados.
2. **Elaboración:** Su propósito es establecer la arquitectura base del sistema.
3. **Construcción:** Tiene como objetivo completar el desarrollo del sistema basado en la arquitectura definida.
4. **Transición:** Esta última fase tiene como propósito asegurar que el sistema es entregado a los usuarios además evalúa la funcionalidad y rendimiento del último entregable de la fase de construcción.

Se decide utilizar la metodología OpenUp como guía en el proceso de desarrollo de software al beneficiar el desarrollo de proyectos pequeños y de corto plazo. Es modificable lo que permite al cliente incorporar ideas nuevas. Es la metodología de desarrollo propuesta por la línea base de la arquitectura del Departamento Integración de Soluciones de DATEC y es la empleada en el proyecto GDR 2.0 donde se implementa la API de servicios web.

1.4.2 Lenguaje de Modelado

El modelado de sistemas software es una técnica que ayuda al ingeniero de software a visualizar el sistema a construir. De ahí que los modelos pueden utilizarse para la comunicación con el cliente, grupo de desarrollo y otros factores que intervienen en el proceso de construcción de software.

El Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés de ***Unified Modeling Language***), en su versión 2.0. Permite especificar, construir, documentar y visualizar el modelado de clases, además de incluir

los procesos de negocio y funciones del sistema. Define las API o componentes reutilizables, la construcción de los diagramas de clases, los componentes del sistema, despliegue e interacción. Presenta capacidades para especificar comportamientos brindando apoyo a las metodologías de desarrollo de software abstrayéndose de los sistemas informáticos, soporta un diseño Orientado a Objetos (OO, por sus siglas en inglés de ***Object Oriented***), donde se denotan en la representación visual del modelo aspectos como la visibilidad de los atributos o métodos, las relaciones entre clases y la variedad existente de las mismas (18).

Se decide utilizar UML debido a que es el lenguaje de modelado propuesto por la línea base de la arquitectura del Departamento Integración de Soluciones de DATEC y es la empleada en el proyecto GDR 2.0 donde se implementa la API de servicios web.

1.4.3 Herramienta CASE de Modelado

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés de ***Computer Aided Software Engineering***) se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, desarrolladores e ingenieros de software, durante el ciclo de vida de un software, proporcionándole un aumento en su productividad y logrando un mayor ahorro de tiempo. (19)

Visual Paradigm for UML

Visual Paradigm for UML 8.0 es una herramienta que soporta el ciclo de vida completo de desarrollo de un software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Permite representar gráficamente varios diagramas y facilita la generación de código. Es una herramienta multiplataforma, fácil de instalar y utilizar (20).

Partiendo de la arquitectura definida por el proyecto GDR 2.0 y del estudio realizado, se determinaron características importantes que presenta la herramienta. Se decide utilizar Visual Paradigm for UML en su versión 8.0 para guiar el modelado de la solución.

1.4.4 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) es una colección de programas que permiten a los usuarios crear y mantener una base de datos. Es un sistema de software que facilita los procesos de definición, construcción y manipulación de la base de datos para distintas aplicaciones. Sirve de interfaz entre el usuario, la base de datos y la aplicación, contribuyendo a su integridad, confidencialidad y seguridad. Los

sistemas de gestión de bases de datos son diseñados para manejar grandes volúmenes de datos de una forma clara, sencilla y ordenada. (21)

PostgreSQL

PostgreSQL es un sistema gestor de base de datos relacional OO, es decir que la información se representa mediante objetos. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos, lo cual garantiza la estabilidad del sistema. Dispone de versiones tanto para Linux como para Windows. Su arquitectura brinda fiabilidad, integridad de datos y estabilidad. (22) Posee una amplia comunidad de desarrollo en Cuba, principalmente en la UCI en el centro DATEC se desarrolla una personalización de este gestor.

Como gestor de base de datos se utilizará PostgreSQL en su versión 9.1, por las características que presenta y por ser el gestor de base de datos propuesto por la línea base de la arquitectura del Departamento Integración de Soluciones.

1.4.5 Administrador de bases de datos

Un administrador de bases de datos tiene la responsabilidad de mantener y operar las bases de datos que conforman el sistema de información de una entidad. Es el software que permite que una institución centralice sus datos, los administre eficientemente y proporcione acceso mediante programas de aplicación.

PgAdmin III

PgAdmin III es una aplicación gráfica para la administración del gestor de base de datos PostgreSQL. Es compatible con plataformas como: Linux, Solaris, Mac OS y Windows. Presenta una interfaz que permite la gestión de la información de una forma más rápida. Crear desde simples consultas SQL hasta consultas de una alta complejidad. Permite trabajar eficientemente con las tablas de la base de datos. Soporta grandes volúmenes de información y presenta una amplia documentación. Es una herramienta de código abierto respaldada por una amplia comunidad de desarrolladores que encaminan sus esfuerzos al perfeccionamiento de la misma. (23) Por todas las facilidades que ofrece esta herramienta, fue seleccionada por el proyecto GDR para el desarrollo de sus aplicaciones.

Se decide utilizar PgAdmin III para la administración del gestor de base de datos por ser la seleccionada por la línea base de la arquitectura del Departamento Integración de Soluciones de DATEC y es la empleada en el proyecto de desarrollo de GDR 2.0 donde se implementa la API de servicios web.

1.4.6 Lenguajes de Programación

Los lenguajes de programación son lenguajes artificiales que pueden utilizarse para definir una secuencia de instrucciones para su procesamiento por una computadora. Son herramientas que permiten crear programas y software. Los lenguajes de programación están formados por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, además el significado de sus elementos y expresiones. Son utilizados para controlar el comportamiento lógico de una máquina y para expresar algoritmos con precisión. Estos lenguajes permiten ser leídos y escritos por personas y a su vez resultan independientes del modelo de computadora a utilizar. Existen varios lenguajes de programación tales como: Python, Java y C++. (24)

PHP

Procesador Hipertexto (PHP, por sus siglas en inglés de ***Hypertext Pre-processor***) es un lenguaje que está implementado especialmente para el desarrollo web. PHP es un lenguaje de programación de alto nivel que se ejecuta en el servidor y está diseñado para la creación de páginas web dinámicas con acceso a información almacenada en una base de datos. Permite aplicar técnicas de programación orientada a objetos (POO, por sus siglas en inglés de ***Object-Oriented Programming***), de modo que los distintos tipos de datos que se usen estén unidos a sus operaciones. De esta manera los datos y el código se combinan en entidades llamadas objetos. PHP es un lenguaje que puede ser ejecutado en la mayoría de los sistemas operativos tales como: Linux, Mac OS y Microsoft Windows. Posee un soporte para bases de datos MySQL, PostgreSQL, Oracle, MS SQL Server. Este lenguaje es distribuido de forma gratuita y posee una amplia documentación en su página oficial (25).

Para el desarrollo de la API de servicios web en GDR 2.0 se decide utilizar PHP en su versión 5.3 por las características antes expuestas y por ser el lenguaje de programación en que está implementado el sistema GDR 2.0 del lado del servidor y es el que establece la línea de arquitectura del Departamento Integración de Soluciones del Centro DATEC de la UCI.

1.4.7 Marco de trabajo

Los marcos de trabajo en la informática son utilizados debido a que representan una estructura de software compuesta de componentes personalizables e intercambiables, estos permiten el desarrollo de aplicaciones más rápidas permitiendo a los desarrolladores optimizar el trabajo. Los marcos de trabajo desde su creación han sido muy utilizados, debido a que facilita una mejor estructura y organización en sus proyectos. Para

las aplicaciones desarrolladas en PHP existe diversos marcos de trabajo de software libres de calidad que pueden ser utilizados, entre los que se encuentran Zend Framework, KakePHP y Symfony.

Symfony 2.0.18

Symfony es un framework PHP de software libre que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional. Permite la programación orientada a objetos, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Dispone de varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Es compatible con varios gestores de bases de datos, como: MySQL, PostgreSQL, Oracle y Microsoft SQL Server y es multiplataforma. (26) Es el marco de trabajo propuesta por la línea base de la arquitectura del Departamento de Integración de Soluciones para el desarrollo de las aplicaciones y el empleo en el desarrollo actual de GDR 2.0.

1.4.8 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés de ***Integrated Development Environment***), es un medio de desarrollo que ha sido empaquetado como un programa de aplicación, generalmente está compuesto por un conjunto de herramientas de programación en las que se encuentra el compilador, editor de código, depurador y constructor de interfaces gráficas. Proporciona un marco de trabajo amigable para diversos lenguajes de programación.

NetBeans IDE

El NetBeans es un IDE gratuito y de código abierto. Es una plataforma para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Permite elaborar aplicaciones para los ambientes tradicionales escritorio y web, también para dispositivos portátiles o móviles. Ofrece una amplia documentación y recursos de capacitación. Es compatible con sistemas operativos como: Windows, Linux, Mac OS X y Solaris. Está desarrollado en lenguaje Java. Tiene una interfaz amigable e intuitiva, además brinda la funcionalidad de completamiento de código. (27)

Este IDE es compatible tanto para Java como para PHP que son los lenguajes de programación utilizados para el desarrollo de GDR. Es el IDE seleccionado por el grupo de arquitectura del Departamento Integración

de Soluciones. Para la implementación de la API se decide utilizar como entorno de desarrollo integrado el NetBeans en su versión 7.3 por las características antes expuestas.

1.5 Arquitectura para el desarrollo de los servicios web

Un servicio web es diseñado para soportar una interacción entre dos computadoras o más a través de una red. Esta tecnología permite la comunicación entre aplicaciones o componentes de aplicaciones, mediante formatos y protocolos de comunicación estándares, independientemente del lenguaje de programación, plataforma y formas de presentación.

Se decide utilizar como arquitectura para el desarrollo de los servicios web REST (por sus siglas en inglés de ***Representational State Transfer***), establecido en la línea de arquitectura del proyecto GDR. Los servicios web basados en REST intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar. Por tanto, este estilo se centra más en interactuar sobre recursos y estado, que con mensajes y operaciones. (28)

REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como los recursos son definidos y diseccionados; el término frecuentemente se utiliza cuando se desea describir a una interfaz que transmite datos específicos de un domino sobre HTTP.

Los objetivos de este estilo de arquitectura son:

- Escalabilidad de la interacción con los componentes. La web ha crecido grandemente sin degradar su rendimiento. Una prueba de ello es la variedad de clientes que pueden acceder a través de la web: estaciones de trabajo, sistemas industriales y dispositivos móviles.
- Generalidad de interfaces. Gracias al protocolo HTTP, los clientes pueden interactuar con cualquier servidor HTTP sin ninguna configuración especial.
- Puesta en funcionamiento independiente. Este hecho es una realidad que debe tratarse cuando se trabaja en Internet. Los clientes y servidores pueden ser puestos en funcionamiento durante años. Por tanto, los servidores antiguos deben ser capaces de entenderse con clientes actuales y viceversa. Diseñar un protocolo que permita este tipo de características resulta muy complicado. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URLs, a través de la habilidad para crear nuevos métodos y tipos de contenido.

- Compatibilidad con componentes intermedios. Los más populares intermediaros son varios tipos de proxy[1] para la web y las caches que se utilizan para mejorar el rendimiento. Otros permiten reforzar las políticas de seguridad donde destacan los firewalls y gateway[2] permitiendo encapsular sistemas no propiamente web. Por tanto, la compatibilidad con intermediarios permite reducir la latencia de interacción además de reforzar la seguridad y encapsular otros sistemas. (29)

Implementación de servicios web con REST

REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP. A continuación se lista un conjunto de pasos lógicos necesarios para el desarrollo:

1. Definir las URL para cada funcionalidad que se desee realizar.

```
http://localhost/ruta_del_servicio/login
```

Fig. 2: Ejemplo de URL para php

2. Se implementan las funcionalidades definiendo los parámetros de entrada y de salida. Estos servicios web generarán una respuesta a una petición realizada por un cliente del servicio, respuesta que generalmente será en formato JSON o XML.

```
public function login($data)
{
    ..............

 return json_encode($data);
}
```

Fig. 3: Ejemplo en código de php

login: En la petición se envía los parámetros, la función comprueba el tipo de petición y que sea del tipo GET, en caso contrario enviara una respuesta de error. Una vez recibida la petición se verificarán los

[1] Proxy: En una red informática, es un programa o dispositivo que realiza una acción en representación de otro.
[2] Gateway: Es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación.

parámetros enviados, se consultará a la base de datos y en caso de que los datos sean correctos se devolverá la respuesta en formato JSON con los datos del usuario y las correspondientes cabeceras de que la consulta se realizó correctamente (200).

3. Se publica el servicio web permitiendo visibilidad vía URL a las funcionalidades implementadas.

Consumo de servicios web REST en Java.

Una petición completa a una aplicación o cliente de servicio web REST debe incluir dentro del encabezado y del cuerpo HTTP de la petición todos los parámetros, contexto y datos que necesita el servidor para generar la respuesta.

Para consumir el servicio se debe analizar en un momento inicial que método se desea consultar, REST indica que a la hora de obtener un recurso el método que se debe utilizar es GET, para añadirlo POST, para actualizarlo PUT y para eliminarlo DELETE. Si el servicio web que se va a consumir es por el método GET simplemente se utiliza la clase de java URL para abrir la conexión mediante el método openConnection, por defecto este método crea una conexión utilizando el método GET, y devuelve un objeto de la clase de java HttpURLConnection, esta clase tiene varios métodos, uno para conocer el código de la respuesta del servidor, por ejemplo 200 si el servidor retorno una respuesta sin errores, otro método para obtener el flujo devuelto por el servidor o sea el InputStream, el cual se lee y se convierte a un tipo de dato String.

Para los restantes métodos se emplea la librería externa HttpClient, que posee varias clases como HttpPost, HttpPut y HttpDelete, que se utilizan para crear una conexión por los métodos POST, PUT y DELETE respectivamente.

Una vez capturado el texto correspondiente a la respuesta del servidor se emplea la librería JsonApi, la cual interpreta el texto y permite crear objetos de la clase JsonObject, que da la posibilidad de obtener el arreglo JSON y acceder a cada uno de sus valores.

1.6 Conclusiones del capítulo

La investigación realizada sobre los conceptos principales para el análisis de la situación antes expuesta, fue el punto de partida para lograr una mejor comprensión sobre la solución que se desea desarrollar. A partir del estudio realizado a las herramientas que permiten la gestión de reportes existentes, se detectó que éstas no presentan una API de servicios web. Se selecciona la herramienta iReport como forma de

validación de la API de servicios web. Además, fue seleccionada OpenUp como metodología de desarrollo para guiar el ciclo de vida de la API de servicios web. Como herramienta para el modelado Visual Paradigm 8.0 empleando el lenguaje de modelado UML 2.0. Durante la implementación de la solución se hará uso del IDE NetBeans 7.3, utilizando como lenguajes de programación PHP 5.3 soportados por los marcos de trabajos Symfony 2.0.18. Además para la construcción de la base de datos se usará como gestor PostgreSQL 9.1 y para su administración PgAdmin III.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA API DE SERVICIOS WEB EN GDR 2.0

Introducción

En el presente capítulo se identifican las clases del dominio y la relación entre estas, los requerimientos funcionales y no funcionales que debe cumplir la API de servicios web para GDR 2.0. Se realiza el modelado de casos de uso del sistema y su descripción textuales. Este capítulo tiene el objetivo de construir el diseño de la solución a implementar y que contribuya a dar la solución deseada a las necesidades que dieron su origen.

2.1 Modelo de Dominio

Un modelo del dominio captura los tipos más importantes de objetos que existen, o los eventos que suceden en el entorno donde estará el sistema, se definen estos conceptos y se unen o relacionan en un diagrama UML. (30)

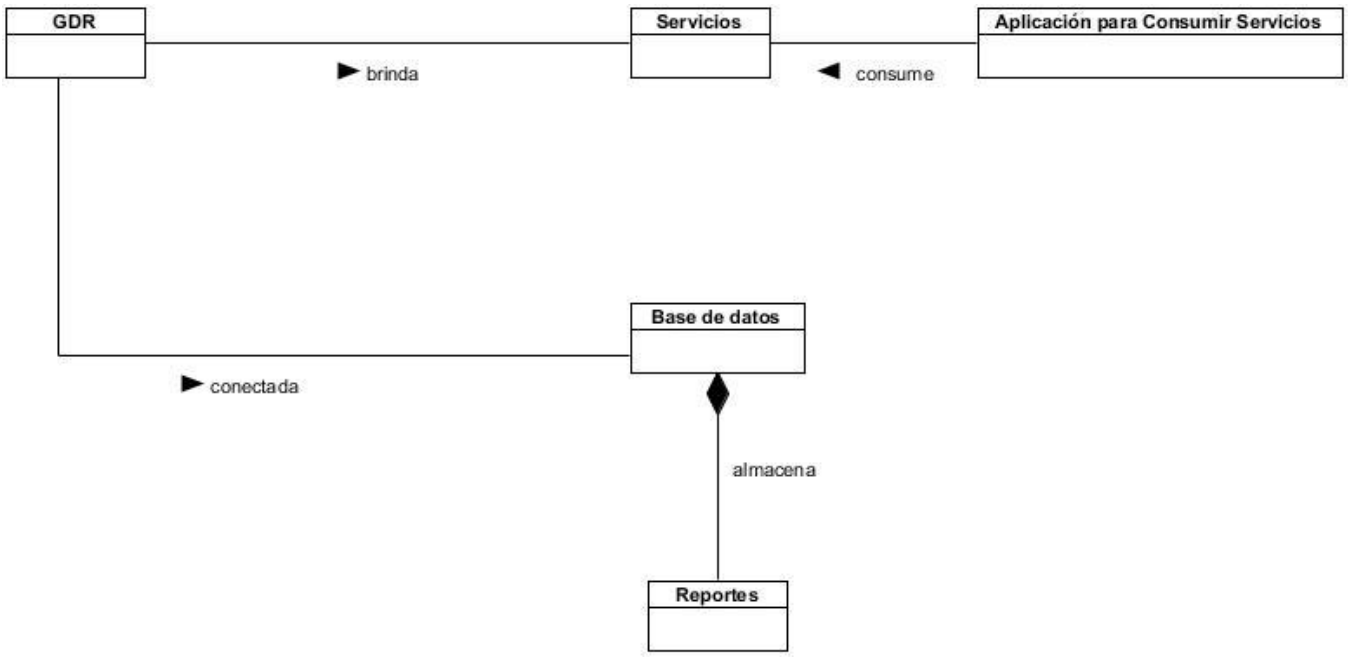


Fig. 4: Modelo de dominio del sistema

La Fig. 4*: Modelo de dominio del sistema* representa el modelo de dominio del sistema, donde una aplicación externa consume los servicios que brinda el proyecto GDR el cual permite a sus clientes consultar la base de datos correspondiente a su organización y generar reportes con la información consultada. La administración de estos reportes se realiza desde un ambiente de escritorio. GDR 2.0 publica servicios para llevar el ciclo de vida de los reportes, permitiendo realizar operaciones sobre los datos de GDR.

2.1.1 Definición de las clases del modelo del dominio

Aplicación para Consumir Servicios: Esta clase representa la aplicación de escritorio que va a consumir los servicios publicados por el GDR 2.0.

GDR: Representa el proyecto Generador Dinámico de Reporte en su versión 2.0 del Departamento Integración de Soluciones, el cual permite el tratamiento del ciclo de vida de los reportes.

Servicios: Representa los diferentes servicios web que el proyecto GDR en su versión 2.0 ofrece para la gestión de reportes de forma asíncrona.

Base de Datos: Aplicación informática que maneja la información del sistema GDR entre los que destacan los reportes y usuarios.

Reportes: Es un documento generado por el sistema que presenta de manera estructurada y/o resumida los datos de acuerdo a un interés específico.

2.2 Requisitos del software

En la ingeniería del software, los requisitos del sistema establecen con detalles las funciones, servicios y restricciones operativas del sistema. Establecen lo que el software debe hacer y bajo qué circunstancias debe hacerlo. (31)

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. (31)

RF1: Publicar servicio web en GDR 2.0 para permitir listar reportes.

Descripción: La API debe permitir mostrar los reportes disponibles.

Salida: Muestra un listado de reportes.

RF2: Publicar servicio web en GDR 2.0 para permitir adicionar reportes.
Entrada: Datos del reporte nuevo.
Descripción: El sistema debe permitir guardar en la base de datos de GDR los reportes creados.
Salida: Se adicionan nuevos reportes a la base de datos de GDR 2.0.

RF3: Publicar servicio web en GDR 2.0 para permitir eliminar reportes.
Entrada: Id del reporte que se desea eliminar.
Descripción: Los reportes se eliminarán de la base de datos de GDR 2.0.
Salida: Se eliminan reportes en la base de datos de GDR 2.0.

RF4: Publicar servicio web en GDR 2.0 para permitir modificar reportes.
Entrada: Id del reporte que se desea modificar y los datos nuevos.
Descripción: Modificar los reportes en GDR 2.0.
Salida: Se actualizan los reportes con los cambios efectuados.

RF5: Publicar servicio web en GDR 2.0 para permitir la autenticación del usuario.
Entrada: Se pasan como parámetros el usuario y la contraseña del cliente.
Descripción: Se comprueba si el usuario y la contraseña pasados por parámetros son correctos y si el usuario tiene acceso a los reportes.
Salida: El usuario podrá acceder a los reportes.

RF6: Publicar servicio web en GDR 2.0 para adicionar usuario de GDR 2.0.
Entrada: Datos del usuario nuevo.
Descripción: Permitir adicionar un nuevo usuario a la base de datos del GDR 2.0.
Salida: Nuevo usuario adicionado.

RF7: Publicar servicio web en GDR 2.0 para modificar usuario de GDR 2.0.
Entrada: Datos del usuario nuevo y su id.
Descripción: Permitir modificar un usuario a la base de datos del GDR 2.0.
Salida: Usuario de GDR 2.0 modificado.

RF8: Publicar servicio web en GDR 2.0 para eliminar usuario de GDR 2.0.

Entrada: Id del usuario que se desea eliminar.

Descripción: Permitir eliminar un usuario de la base de datos del GDR 2.0.

Salida: Usuario eliminado de la base de datos de GDR 2.0.

RF9: Publicar servicio web en GDR 2.0 para listar usuario de GDR 2.0.

Descripción: Permitir listar los usuarios de la base de datos del GDR 2.0.

Salida: Lista de usuarios.

RF10: Publicar servicio web en GDR 2.0 para adicionar rol de GDR 2.0.

Entrada: Datos del nuevo rol.

Descripción: Permitir adicionar un nuevo rol a la base de datos del GDR 2.0.

Salida: Nuevo rol incorporado.

RF11: Publicar servicio web en GDR 2.0 para modificar rol de GDR 2.0.

Entrada: Datos del rol nuevo y su id.

Descripción: Permitir modificar un rol a la base de datos del GDR 2.0.

Salida: Rol de GDR 2.0 modificado.

RF12: Publicar servicio web en GDR 2.0 para eliminar rol de GDR 2.0.

Entrada: Id del rol que se desea eliminar.

Descripción: Permitir eliminar un rol de la base de datos del GDR 2.0.

Salida: Eliminar rol de GDR 2.0.

RF13: Publicar servicio web en GDR 2.0 para listar roles de GDR 2.0.

Descripción: Permitir listar los roles de la base de datos del GDR 2.0.

Salida: Lista de roles.

RF14: Publicar servicio web en GDR 2.0 para asignar rol.

Entrada: Id del rol e id del usuario.

Descripción: Permitir asignarle un rol a un usuario.

Salida: Rol asignado a un usuario.

RF15: Publicar servicio web en GDR 2.0 para asignar usuario.

Entrada: Id del rol e id del usuario.

Descripción: Permitir asignarle un usuario a un rol.

Salida: Usuario con un determinado rol.

RF16: Publicar servicio web en GDR 2.0 para asignar permisos.

Entrada: Id del rol o del usuario al que se le va asignar permisos, sobre que módulo se van asignar los permisos y que permisos va a permitir.

Descripción: Permitir asignarle permisos de lectura, escritora y borrar a los usuarios o a los roles.

Salida: Usuario o rol con nuevos permisos.

RF17: Publicar servicio web en GDR 2.0 para obtener permisos.

Entrada: Id del rol o del usuario del que se desea obtener permisos, sobre que módulo tiene permisos y que permisos tiene.

Descripción: Permitir obtener los permisos de un determinado usuario o rol.

Salida: Lista de permisos de un determinado usuario o rol sobre un determinado módulo.

2.2.2 Requisitos no funcionales

Estos requerimientos a menudo se aplican al sistema en su totalidad y no a características o servicios individuales del sistema. Por su parte, son los que imponen restricciones en el diseño o la implementación. (32)

RNF: Software

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos de software:

- Debe estar instalado GDR en su versión 2.0.
- SO: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.

RNF: Hardware

Las PC clientes deben cumplir con los requerimientos mínimos, como se muestra a continuación:

- ✓ Debe tener instalado un diseñador de reporte configurado para consumir los servicios de la API.
- ✓ Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 256 MB.
- ✓ 10 GB de espacio de disco duro.

El servidor donde se instalará la aplicación se debe cumplir con los siguientes requisitos:

- ✓ Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Como mínimo requiere 1 GB de memoria RAM.
- ✓ Necesita espacio en disco duro igual o superior a 10 GB.

RNF: Usabilidad

El sistema debe estar bien estructurado y cada método debe tener una descripción explicando su funcionalidad para un mejor entendimiento de otros desarrolladores.

RNF: Rendimiento

El sistema debe mantener tiempos de respuestas en un marco razonable de tres segundos, permitiendo que existan al menos 100 usuarios conectados de forma simultánea.

RNF: Confiabilidad

La información manejada por el sistema debe estar protegida de acceso no autorizado.

RNF: Disponibilidad

GDR podrá disponer de los servicios de la API en todo momento.

2.3 Diagrama de caso de uso del sistema

Los diagramas de casos de uso del sistema documentan el comportamiento de un sistema desde el punto de vista funcional. Los casos de uso son una unidad coherente, expresada como transacción entre los actores y el sistema, estos determinan los requisitos funcionales del sistema. Su propósito es enumerar los actores y los casos de uso, y demostrar las relaciones que puedan existir entre ellos. (18)

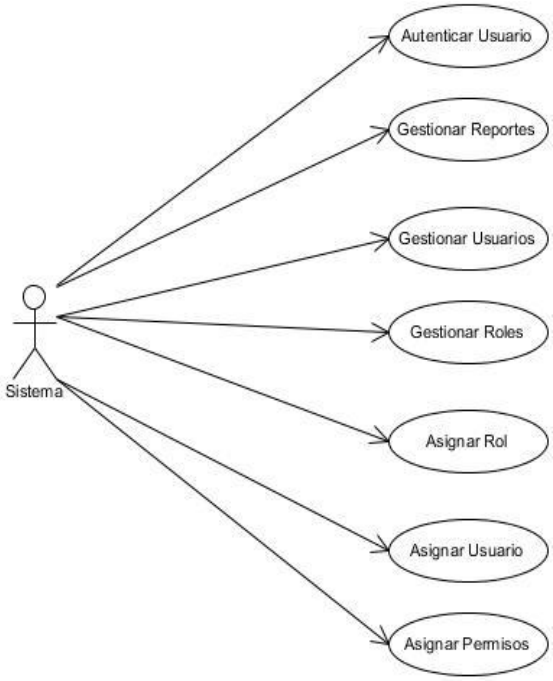


Fig. 5: Caso de uso del sistema

Descripción de los caso de uso del sistema

Autenticar Usuario: Permite realizar la autenticación de los usuarios del sistema además de posibilitarles desconectarse.

Gestionar Reportes: Permite adicionar, editar, eliminar y listar reportes.

Gestionar Usuarios: Permite adicionar, editar, eliminar, y listar usuarios.

Gestionar Roles: Permite adicionar, editar, eliminar, y listar un rol.

Asignar Rol: Permite asignarle uno o varios roles a un usuario seleccionado.

Asignar Usuario: Permite asignarle uno o varios usuarios a un rol seleccionado.

Asignar Permisos: Permite asignarle permisos a los usuarios o a los roles.

Descripción del caso de uso "Gestionar reportes"

Tabla 2: Descripción del caso de uso "Gestionar Reportes"

Caso de uso:	Gestionar Reportes.
Actores:	Sistema

Resumen:	Permite la gestión de reportes (adicionar, listar, modificar, eliminar) en GDR.2.0.
Precondiciones:	Se gestionaron los reportes en GDR 2.0.
Referencias	RF 1, RF 2, RF 3, RF 4
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Valida que los datos son correctos.	2. Permite realizar varias acciones con los reportes: -Adicionar reporte, ver sección **"Adicionar reporte".** -Modificar reporte, ver sección **"Modificar reporte".** -Listar reporte, ver sección **"Listar reporte".** -Eliminar reporte, ver sección **"Eliminar reporte".**

Sección "Adicionar reporte"	
Acción del Actor	**Respuesta del Sistema**
1. Envía el token del usuario y la petición (nombre del archivo, nombre del reporte, el texto del reporte).	
	2. Valida que la solicitud sea enviado por POST.
	3. Envía los datos y el token.
	4. Valida datos y token.
	5. Verifica el token.
	6. Adiciona el reporte.
	7. Mensaje ("Se adicionó correctamente").

Prototipo de Interfaz
No hay prototipo.

Flujo Alterno al paso 4 "Token o datos del reporte incorrecto"	
Acción del Actor	**Respuesta del Sistema**
	4a. Si el token o los datos del reporte no son válido envía un mensaje de error ("Datos incorrectos").

Flujo Alterno al paso 5 "Verificar token "	
Acción del Actor	**Respuesta del Sistema**

	5a. Si no existe ese token en la base de datos se manda un mensaje de error (“Token inválido”).
Sección “Listar reportes”	
Acción del Actor	**Respuesta del Sistema**
1. Envía el token del usuario.	
	2. Valida que la solicitud sea enviado por GET.
	3. Envía el token.
	4. Valida el token.
	5. Verifica token.
	6. Busca lista con los reportes.
	7. Envía lista de reportes.
Prototipo de Interfaz	
No hay prototipo.	
Flujo Alterno al paso 4 “Token no válido”	
Acción del Actor	**Respuesta del Sistema**
	4a. Si el token no es válido envía un mensaje de error (“Datos incorrectos”).
Flujo Alterno al paso 5 “Token incorrecto”	
Acción del Actor	**Respuesta del Sistema**
	5a. Si no existe ese token en la base de datos se envía un mensaje de error (“Token no válido”).
Sección “Modificar reporte”	
Acción del Actor	**Respuesta del Sistema**
1. Envía el token del usuario y los datos nuevos del reporte que se desea modificar.	
	2. Valida que la solicitud sea enviado por PUT.
	3. Envía los datos y el token.
	4. Valida los datos y el token.
	5. Verifica el token.

	6. Verifica si existe el reporte.
	7. Modifica el reporte.
	8. Envía un mensaje (“Se modificó correctamente”).
Prototipo de Interfaz	
No hay prototipo.	
Flujo Alterno al paso 4 “Token no válido”	
Acción del Actor	**Respuesta del Sistema**
	4a. Si el token o los datos no son válidos envía un mensaje de error (“Datos incorrectos”).
Flujo Alterno al paso 5 “Token no existente”	
	5a. Si no existe ese token en la base de datos se envía un mensaje de error (“Token no válido”).
Flujo Alterno al paso 6 “Reporte no existente”	
	6a. Si no existe el reporte en la base de datos se envía un mensaje de error (“El reporte no existe”).
Sección “Eliminar reporte”	
Acción del Actor	**Respuesta del Sistema**
1. Envía el token del usuario, el id del reporte que desea eliminar.	
	2. Valida el que la solicitud sea enviado por DELETE.
	3. Envía los datos y el token.
	4. Valida el token.
	5. Verifica el token.
	6. Busca reporte.
	7. Elimina reporte.
	8. Envía un mensaje (“Se eliminó correctamente”).
Prototipo de Interfaz	
No hay prototipo.	
Flujo Alterno al paso 4 “Token incorrecto”	
Acción del Actor	**Respuesta del Sistema**

	4a. Si el token no es válido envía un mensaje de error (“Datos incorrectos”).
Flujo Alterno al paso 5 “ El token no existe”	
	5a. Si no existe ese token en la base de datos se envía un mensaje de error (Token no válido).
Flujo Alterno al paso 6 “ El reporte no existe”	
Acción del Actor	**Respuesta del Sistema**
	6a. Si el reporte no existe envía un mensaje de error (“Reporte no encontrado”).

2.4 Patrones de caso de uso del sistema

Los patrones de caso de uso son comportamientos que deben existir en el sistema, describe su uso y cómo interactúa con los usuarios. Estos patrones capturan mejores prácticas para modelar casos de uso. A continuación se presentan los empleados en la solución del API.

- CRUD completo

  Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual. Consta de un caso de uso, llamado Información CRUD o Gestionar información. Modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

  Se manifiesta el uso de este patrón en los Casos de Uso: Gestionar Reportes, Gestionar Usuario y Gestionar Roles.

2.5 Diagrama de clases del diseño para el caso de uso “Gestionar Reportes”

El diagrama de clases del diseño (DCD) muestra los atributos y métodos de cada clase, representando la colaboración y las responsabilidades de cada una de ellas, entorno al sistema que conforman.

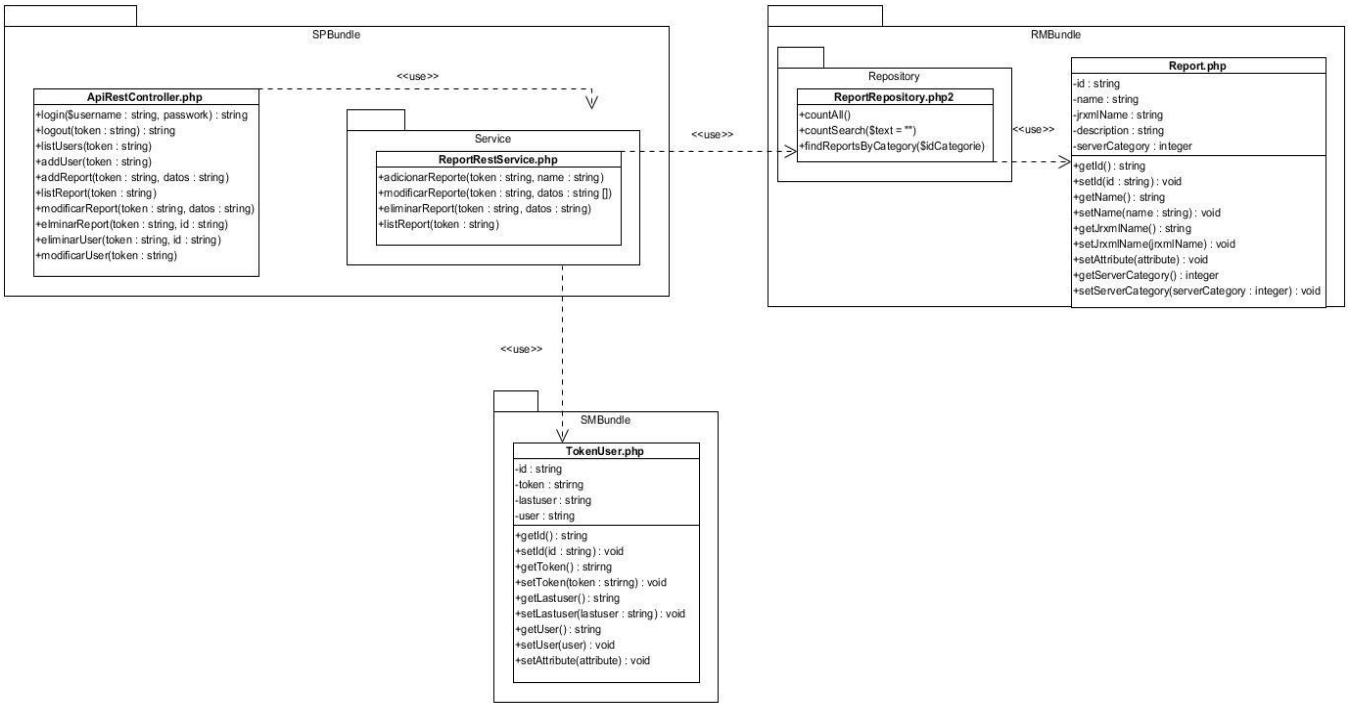


Fig. 6: Diagrama de clases del diseño para el caso de uso “Gestionar Reportes”

Descripción de las clases del diseño

En el diagrama de clases para el caso de uso Gestionar Reportes, la clase “ApiRestController” contiene las acciones que dan respuestas a las peticiones realizadas por el sistema, dicha clase se relaciona con “ReportRestService” contenida en el paquete “service” que se encuentra dentro del bundle “SPBundle” (hace referencia al módulo Administrador de seguridad). La clase “ReportRestService” está relacionada con “TokenUser” encargada de generar el token único de los usuarios, también tiene una relación con la clase “ReportRepository.php” responsable de contener los métodos para realizar las acciones sobre “Report” la cual tiene los atributos de los reportes de GDR 2.0.

2.6 Patrones de arquitectura

Los patrones arquitectónicos son plantillas que describen los principios estructurales globales que construyen las distintas arquitecturas de software. Proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos.

La solución existente utilizará el patrón arquitectónico Modelo Vista Controlador (MVC), pues se hará uso del marco de trabajo Symfony 2.0 para el desarrollo de la API de servicio web. MVC es un patrón de

arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo que ha sido usado en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo. (33)

Está formado por tres niveles:

- El **Modelo** representa la información con la que trabaja la aplicación. Su lógica de negocio en este nivel se encuentra aplicado en las clases: GDRUser, GDRRole, TokenUser y Report.
- La **Vista** o interfaz de usuario es utilizada por los usuarios para interactuar con la aplicación. Debido a que el usuario no interactúa directamente sobre la API de servicios web esta no presenta una vista.
- El **Controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. En este nivel se encuentran las clases: ApiRestController, ReporteRestService, UserRestService, RolRestService y SecurityRestService.

2.7 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptado para resolver un problema de diseño general en un contexto particular. Constituyen parte de la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Entre los patrones de diseño encontramos los patrones denominados Banda de los 4 (GOF, por sus siglas en inglés de ***Gang Of Four***) y los Patrones de Asignación de Responsabilidades (GRASP, por sus siglas en inglés de ***General Responsibility Assignment Software Patterns***). (34)

2.7.1 Patrones GRAP

Controlador: Un Controlador es un objeto que se encarga de manejar un evento del sistema. Define además el método de su operación. (35)

El uso de este patrón se evidencia en la clase ApiRestController, ReportRestService, SegurityRestService, UserRestService y RolRestService. Los controladores mencionados anteriormente son los encargados de interactuar con el modelo.

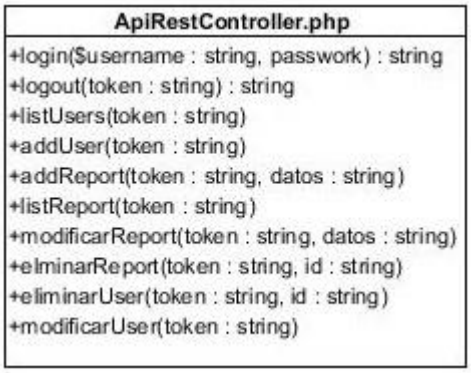


Fig. 7: Clase controladora

Creador: El creador guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea muy común en sistemas orientados a objetos. El intento básico del patrón creador es encontrar un creador que necesite estar conectado al objeto creado en un evento en particular.

La creación de objetos es una de las actividades más comunes en un sistema orientado a objetos. Consecuentemente es muy útil tener un principio general para la asignación de la creación de responsabilidades. (35)
El uso de este patrón se evidencia en la clase ReportRestService, UserRestService y RolRestService encargadas de crear una instancia de una entidad y de esta manera asignarle atributos.

Alta cohesión: Es la asignación de responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase.
En la solución se observa el uso de este patrón en las clases del modelo, dígase gdruser, gdrrole, report y tokenuser, las cuales poseen la estructura necesaria para almacenar la información de una manera coherente y de acuerdo a la responsabilidad que tenga asignada cada clase.

Bajo Acoplamiento: Este patrón guía la asignación de responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible, de manera que de producirse una modificación en algunas de ellas se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las mismas.
En la solución se observa el uso de este patrón en las clases controladoras del módulo, tales como ReportRestService, UserRestService y RolRestService las cuales heredan de la clase ApiRestController

logrando un bajo acoplamiento y reutilización de los métodos existentes en la clase anteriormente mencionada.

2.7.2 Patrones GOF

Fachada: Proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. (36)

Se representa en la clase ApiRestController que hace la función de fachada con las clases ReportRestService, UserRestService, RolRestService y SecurityRestService.

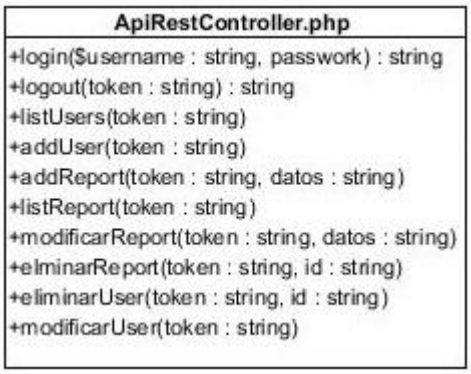


Fig. 8: Representación del patrón fachada

Iterador: Proporcionar una forma de acceder secuencialmente a los elementos de un objeto compuesto por agregación sin necesidad de desvelar su representación interna. Se representa en las clases ReportRestService, UserRestService, RolRestService y SecurityRestService.

Mediador: Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Se representa en la clase ApiRestController.

2.8 Diagrama de secuencia

El diagrama de secuencia se encuentra entre los diagramas de interacción, encargados de modelar el comportamiento dinámico de un sistema y de mostrar la forma en que un grupo de objetos interactúan entre sí a lo largo del tiempo. Este diagrama consta de objetos, mensajes entre estos objetos y una línea de vida del objeto representada por una línea vertical.

Diagrama de secuencia del CU Gestionar Reporte de la sección: Adicionar Reporte.

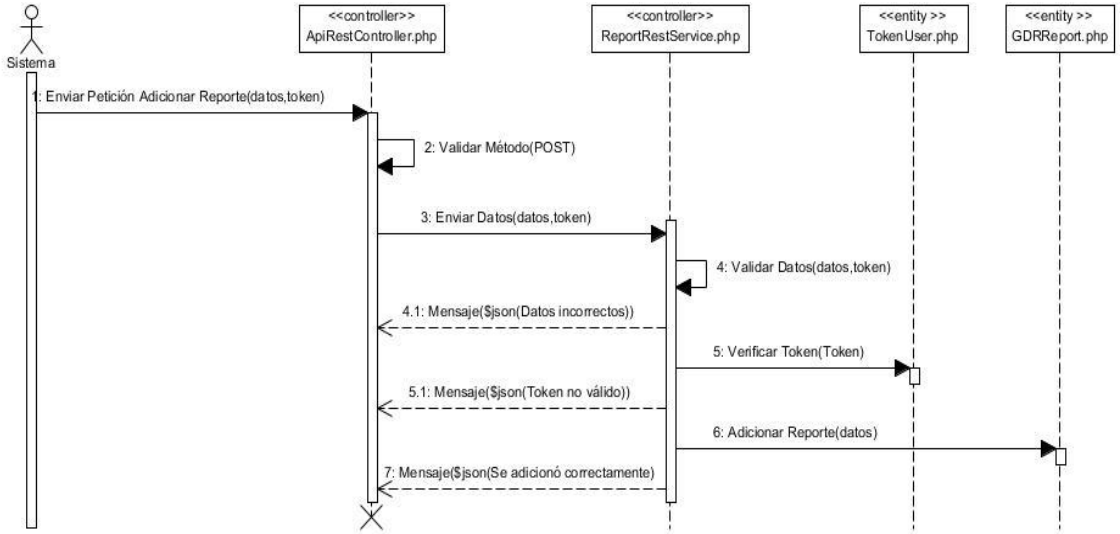


Fig. 9: Diagrama de secuencia del CU Adicionar Reporte

Como se muestra en la figura anterior la clase controladora ApiRestController recibe la petición que le envía el sistema y el token del usuario, posteriormente valida que el método de entrada sea por POST, de ser este correcto envía los datos para la clase "ReportRestService" encargado de validarlos, para el caso contrario envía un mensaje de "Datos incorrectos". Si los datos son correctos, verifica si el token es válido, de ser este correcto inserta el reporte en la base de datos y muestra un mensaje (Se adicionó correctamente). Si el token no es correcto muestra un mensaje (Token no válido).

2.9 Modelo de datos

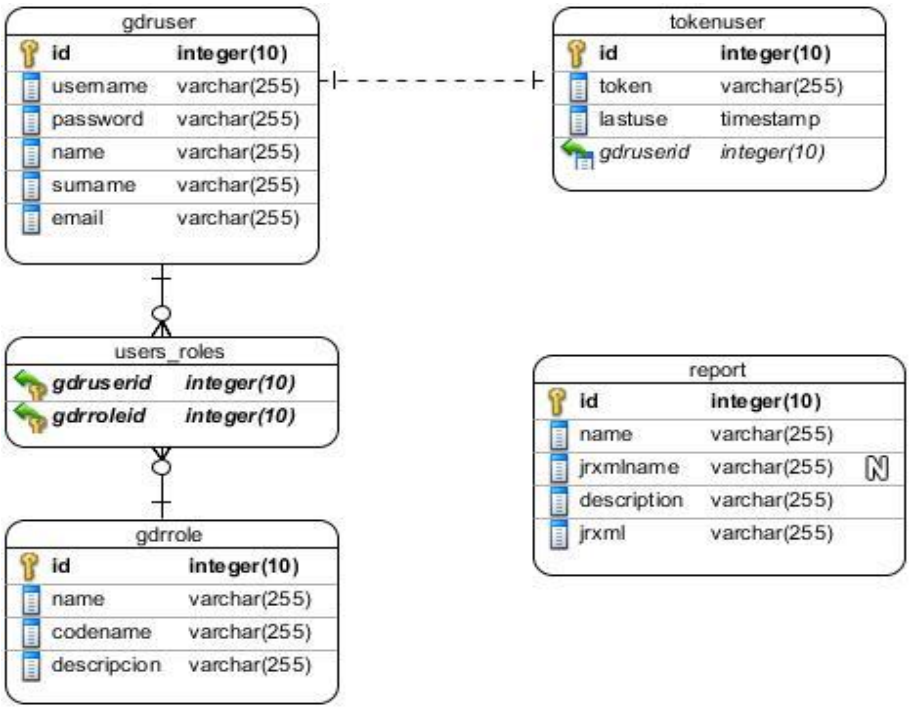


Fig. 10: Modelo de datos de la API de servicios web

Descripción del modelo de datos

En la tabla siguiente se muestra una descripción de cada una de las entidades del modelo de datos de la API de servicios web.

Tabla 3: Descripción de las tablas de la base de datos

Nombre: gdruser		
Descripción: Almacena los datos de los usuarios.		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Identificador auto-incremental.
username	varchar(255)	Usuario del cliente.
password	varchar(255)	Contraseña del cliente.
name	varchar(255)	Nombre del cliente
surname	varchar(255)	Apellidos del cliente
email	varchar(255)	Correo electrónico del cliente

Nombre: tokenuser		
Descripción: Almacena los token de los usuarios.		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Identificador auto-incremental.
Token	varchar(255)	Token del usuario.
lastuser	timestamp	Ultima hora que el usuario hizo algún cambio.
gdruserid	Integer(10)	Id del usuario que se le asignó el token.
Nombre: report		
Descripción: Almacena los reportes.		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Identificador auto-incremental.
name	varchar(255)	Nombre del reporte.
jrxmlname	varchar(255)	Nombre del jxml.
descrpcion	varchar(255)	Descripción del reporte.
jrxml	varchar(255)	El reporte.
Nombre: gdrrole		
Descripción: Almacena los roles		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Identificador auto-incremental.
name	varchar(255)	Nombre del rol.
codename	varchar(255)	Codificador del rol.
descrpcion	varchar(255)	Descripción del rol.
Nombre: users_roles		
Descripción: Almacena la relación entre los usuarios y los roles.		
Atributo	**Tipo**	**Descripción**
gdruserid	Integer(10)	Identificador del usuario.
gdrroleid	Integer(10)	Identificador del rol.

2.10 Diagrama de despliegue

El diagrama de despliegue provee un modelado detallado de la forma en que los componentes se desplegarán. Se compone por nodos, dispositivos y asociaciones, donde los nodos representan elementos de procesamiento, los dispositivos son nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela y las asociaciones son conectores que representan el protocolo utilizado entre los nodos del modelo.

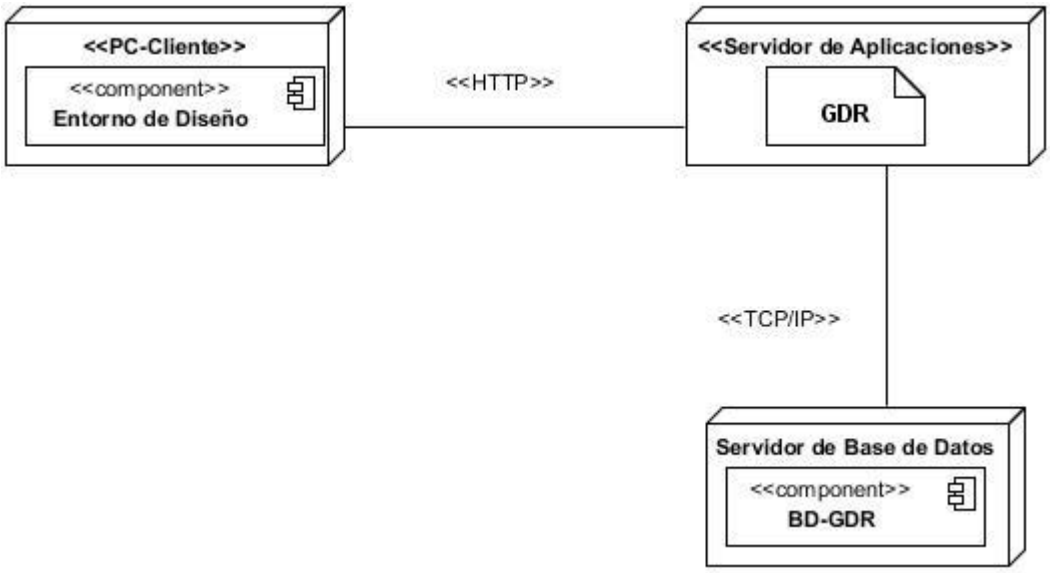


Fig. 11: Diagrama de despliegue

Descripción del diagrama de despliegue

En la figura anterior se muestra el diagrama de despliegue el cual representa tres nodos principales. El nodo PC-Cliente que requiere de un entorno de diseño, el nodo Servidor de Base de Datos en el cual debe estar instalada la base de datos de GDR 2.0 y el nodo Servidor de Aplicaciones, en el cual debe estar instalado el GDR 2.0 con el API de servicios web.

El nodo PC-Cliente estará conectado al nodo Servidor de Aplicaciones por HTTP como protocolo de comunicación. La conexión entre el Servidor de Aplicaciones y el Servidor de Base de Datos se realizará mediante el protocolo de comunicación TCP/IP.

2.11 Conclusiones del capítulo

La elaboración del modelo del dominio y su descripción permitió una mayor comprensión para el desarrollo del API de servicios web. La identificación y especificación de los requisitos no funcionales, así como de los 17 requisitos funcionales que fueron agrupados en siete casos de uso. Permitió definir las funcionalidades y características que debe proveer el API de servicios web. La elaboración de los diagramas de clases del diseño y los diagramas de secuencia propició una mejor comprensión de la distribución de las clases, así como de las relaciones y responsabilidades de cada una de ellas. La aplicación de los patrones de diseño GRASP (Controlador, Creador, Alta cohesión y Bajo acoplamiento) y GOF (Fachada, Iterador y Mediador) posibilitó asignar las responsabilidades adecuadas a cada una de las clases. Además se identificaron el actor del sistema, los casos de uso y la relación existente entre ellos reflejada en el diagrama de casos de uso, apoyado de la descripción detallada de los casos de uso del sistema para obtener así un mayor entendimiento de los requisitos funcionales previamente establecidos. Además se muestra el diagrama de despliegue para representar el entorno físico de cómo quedaría la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA API DE SERVICIOS WEB EN GDR 2.0

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 API de servicios web. Se describen las pruebas realizadas, con el objetivo de detectar algún posible error en la API de servicios web y se desarrolla una extensión al iReport para validar su funcionamiento.

3.1 Implementación de la API de servicios web en GDR 2.0

3.1.1 Diagrama de componentes

Un diagrama de componentes representa la separación de un sistema de software en componentes físicos y muestra las dependencias entre estos. Cada subsistema corresponde a un paquete físico y cada componente a un módulo, fichero o biblioteca existente en la memoria de almacenado. (37)

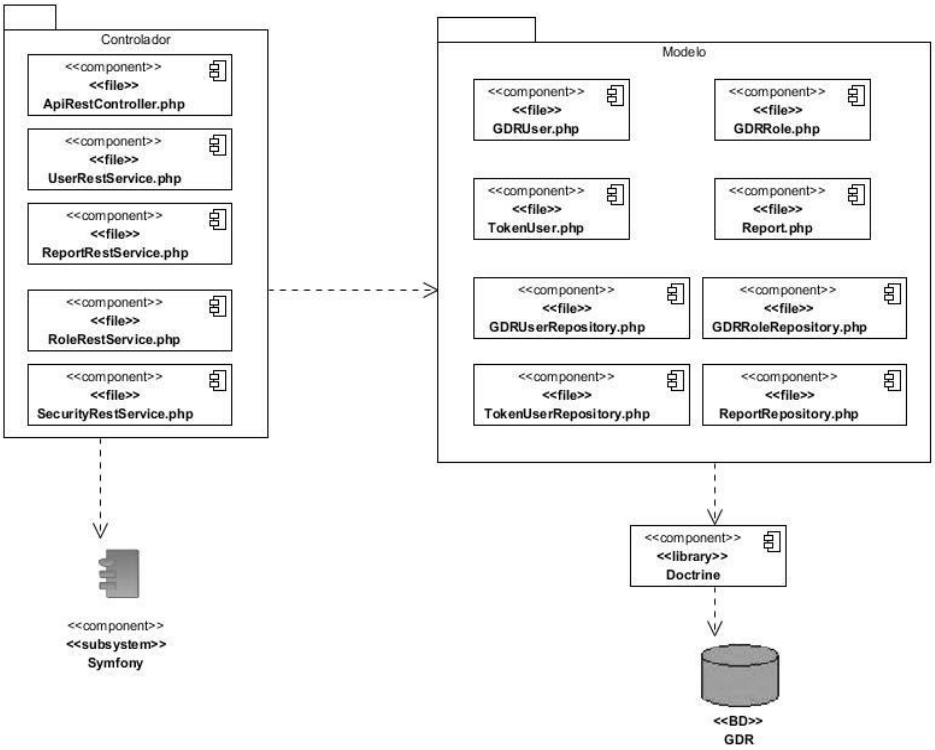


Fig. 12: Diagrama de componentes de la API de servicios web

Descripción del diagrama de componentes

En la figura anterior se expone el diagrama de componente de la solución el cual cuenta con dos paquetes de implementación básicos. Estos paquetes son:

- ✓ El Paquete de Clases Controlador: que contendrá las clases que manipulan los eventos y se apoya en subsistema modelo para dar respuesta a las peticiones.
- ✓ El Paquete de Clases Modelo: que agrupa las clases que interactúan con la base de datos y velan por el cumplimento de las reglas del negocio.

3.1.2 Estilo y estándares de codificación

Una guía de estilo es un conjunto de reglas para dar formato a los programas. El tener un estilo uniforme facilita la lectura del código. Seguir normas de estilo agiliza encontrar la esencia de los programas. Para lograr la legibilidad de un programa es importante considerar aspectos tales como el nombre de los identificadores, escribir el código con cierta alineación y líneas en blanco en lugares apropiados así como realizar una buena documentación. (38)

Se utilizó el estándar de codificación que define el Proyecto Generador Dinámico de Reportes donde:

- El código PHP debe estar delimitado siempre por la forma completa de las etiquetas PHP estándar: <?php ?>
- La longitud recomendable para una línea de código es de hasta 80 caracteres. Esto significa que los desarrolladores deben intentar mantener cada línea de su código por debajo de los 80 caracteres, siempre que sea posible. No obstante, líneas más largas pueden ser aceptables en algunas situaciones. El tamaño máximo de cualquier línea de código PHP es de 120 caracteres.
- Los nombres de clases pueden contener sólo caracteres alfanuméricos. Los números están permitidos en los nombres de clase, pero desaconsejados en la mayoría de casos.
- Para cualquier otro archivo, sólo caracteres alfanuméricos, barras bajas (_) y guiones (-) están permitidos. Los espacios en blanco están estrictamente prohibidos.
- Los nombres de variables pueden contener caracteres alfanuméricos. Las barras bajas (_) no están permitidas. Los números están permitidos en los nombres de variable pero no se aconseja en la mayoría de los casos.

- Los nombres de funciones deben empezar siempre con una letra minúscula. Cuando un nombre de función consiste en más de una palabra, la primera letra de cada nueva palabra debe estar en mayúsculas. Esto es llamado comúnmente como formato "camelCase". (39)

En la siguiente figura se muestra un fragmento del código donde se cumple con estos estilos y estándares.

```
public function listReports($token)
{
    $em = $this->em;
    $json = $this->validateToken($token);
    if ($json['success'] == true) {
        $repository = $em->getRepository('RMBundle:Report');
        $reports = $repository->findAll();
        $reports2json = array();
        foreach ($reports as $u) {
            $reports2json[] = $u->getData();
        }

        $json = array('success' => true, 'reports' => $reports2json);

    }
    return json_encode($json);
}
```

Fig. 13: Fragmento de código del método listReports() de la clase ReportRestService

3.2 Desarrollo de una extensión en el iReport para la validación del API de servicios web de GDR 2.0

La extensión desarrollada para el iReport será utilizada como forma de validación del API de servicios web de GDR 2.0. Mediante la extensión se consume el API de servicios web y de esta forma se comprueba el funcionamiento de cada uno de los requisitos funcionales de la API de servicio web en GDR 2.0, garantizando la gestión de reportes de forma asíncrona desde diversos entornos de diseño.

3.2.1 Descripción de la propuesta

La extensión a desarrollar debe satisfacer los siguientes requisitos funcionales (RF):

RF1: Crear reporte desde la extensión.
RF2: Eliminar reporte desde la extensión.
RF3: Modificar reporte desde la extensión.
RF4: Listar reporte desde la extensión.
RF5: Exportar fichero XML desde la extensión.
RF6: Importar fichero XML desde la extensión.
RF7: Guardar cambios en el XML desde la extensión.
RF8: Crear usuario desde la extensión.
RF9: Eliminar usuario desde la extensión.
RF10: Modificar usuario desde la extensión.
RF11: Listar usuario desde la extensión.
RF12: Asignar rol al usuario desde la extensión.
RF13: Crear rol desde la extensión.
RF14: Eliminar rol desde la extensión.
RF15: Modificar rol desde la extensión.
RF16: Listar rol desde la extensión.
RF17: Asignar usuario al rol desde la extensión.
RF18: Autenticar usuario en GDR 2.0 desde la extensión.
RF19: Cerrar sección del usuario en GDR 2.0 desde la extensión.
RF20: Guardar proyecto desde la extensión.
RF21: Cargar proyecto desde la extensión.
RF 22: Publicar cambios en GDR 2.0 desde la extensión.
RF23: Actualizar la extensión respecto a GDR 2.0 desde la extensión.

Requisitos No Funcionales de la extensión

Hardware

Las PC clientes que trabaje con la extensión debe cumplir con los siguientes requerimientos mínimos:

- ✓ Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Al menos 256 MB de memoria RAM.
- ✓ Un mínimo de 10 GB de espacio libre en disco.

Software

- ✓ Debe estar instalado iReport 5.1 con la plataforma actualiza del NetBeans 7.3.
- ✓ Sistema operativo Windows o Ubuntu,

Arquitectura de una extensión

El NetBeans IDE soporta un acoplamiento flexible de los módulos que es descrito por los datos de su fichero de manifiesto, junto con los datos especificados en los archivos relacionados con XML. La estructura de un módulo es un archivo JAR que consta con la siguiente definición: (40)

- Manifiesto de archivo (manifest.mf)
- Capa de archivo (layer.xml)
- Los archivos de clase, que representan las clases de la solución.
- Los recursos como iconos, properties bundles, etc

manifest.mf: Este archivo es una descripción textual del módulo y su entorno. Cuando se carga un módulo, el archivo de manifiesto es el primer archivo leído por el sistema de módulos.
layer.xml: Proporciona información específica del usuario y define la integración de un módulo en la plataforma. En este archivo se especifica todo lo que un módulo quiere añadir a la plataforma, que van desde las acciones de los elementos de menú a los servicios. Además este formato de archivo es un sistema jerárquico de archivos que contiene las carpetas, archivos y atributos. Al inicio de la aplicación todos los archivos de capas existentes se resumen a un sistema de archivos virtual. Este es el sistema de archivos denominado Sistema, pues constituye la configuración de ejecución de la plataforma NetBeans.
En la siguiente figura se muestra la estructura de la extensión.

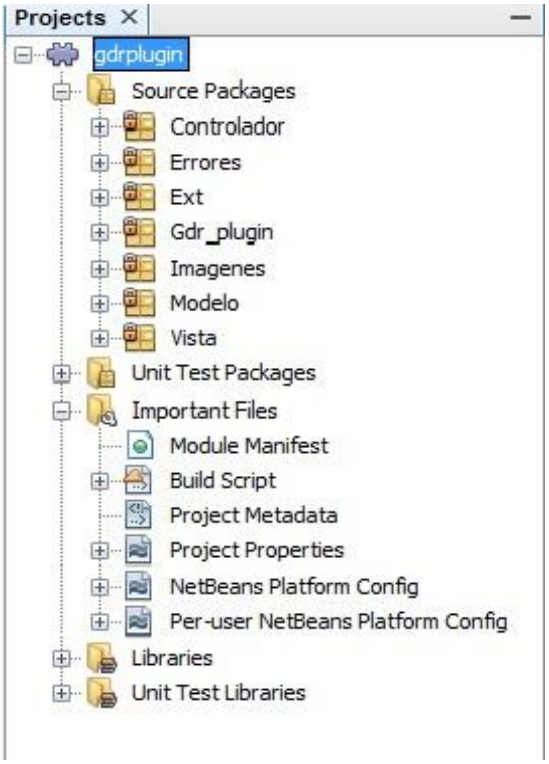


Fig. 14: Arquitectura de una extensión al iReport para consumir la API de servicios web

3.2.2 Validación de la API de servicios web desde la extensión

Con el uso de la extensión se puede lograr la gestión de reportes, de usuarios y roles desde la herramienta de diseño iReport, permitiendo obtener la información almacenada en la base de datos de GDR 2.0 y trabajar con la misma desde la extensión. El uso de la extensión permite la gestión de reportes aunque exista inestabilidad en la red, si existen problemas con la conexión se pueden gestionar los reportes desde el iReport debido a que esta permite realizar una salva temporal en la PC del cliente y una vez solucionado los problemas de la conexión publicar los cambios en GDR 2.0. La comunicación entre la API de servicios web y la extensión se realiza mediante HTTP donde cada objeto tiene su propia URL y para intercambiar los datos se utiliza JSON.

3.2.3 Interfaz de la extensión

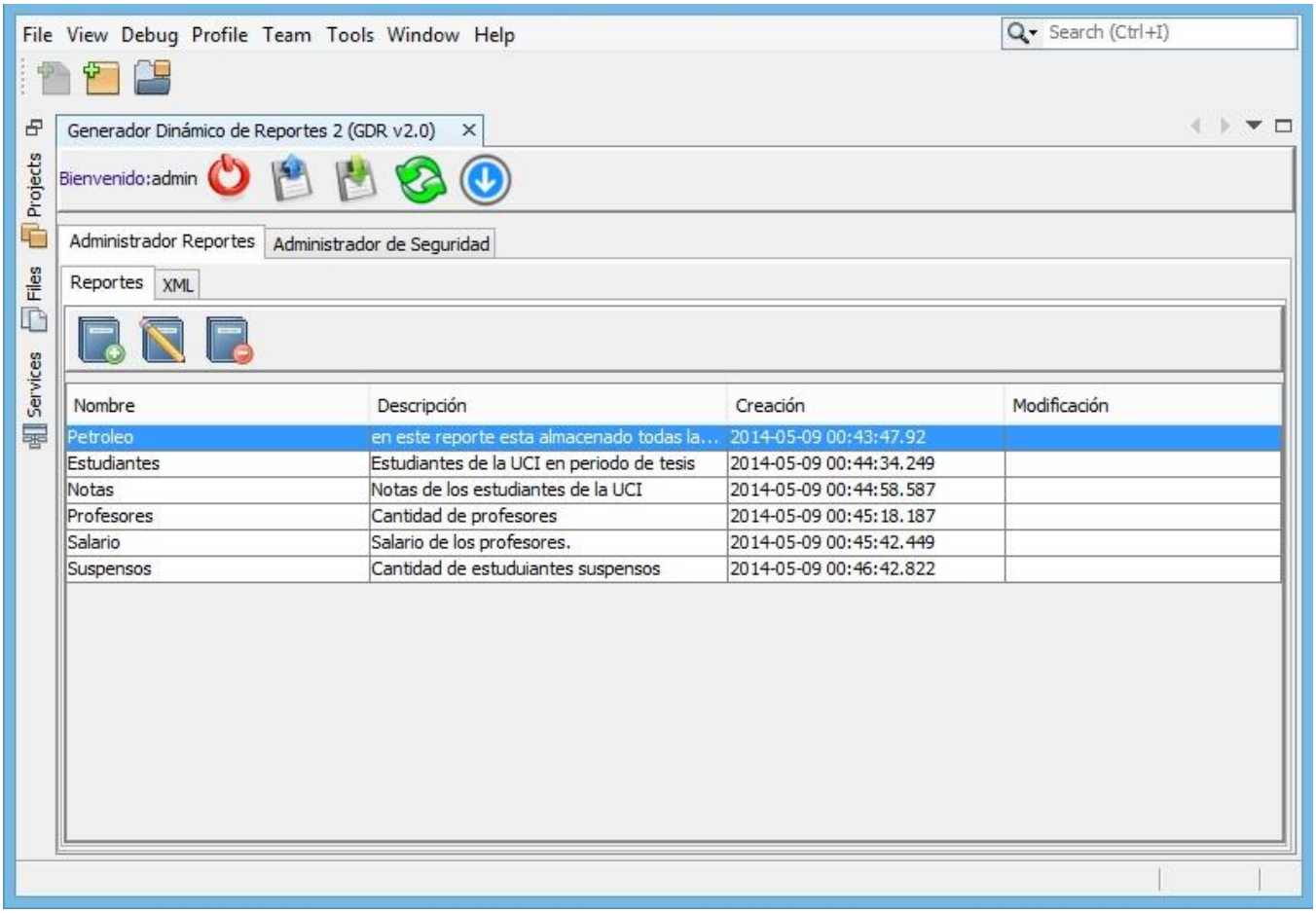


Fig. 15: Interfaz de la extensión

3.3 Estrategia de prueba

Se elaboró una estrategia de pruebas para detectar posibles errores de implementación o usabilidad de la API de servicios web en GDR 2.0.

Nivel de Prueba

En el desarrollo de la API de servicios web se aplicarán las siguientes pruebas para detectar posibles errores en el componente:

1. **Prueba de desarrollador:** Es el primer nivel de pruebas, diseñadas e implementadas por el equipo de desarrollo, donde el programador es quien revisa su código fuente.
   - **Método de prueba**

Caja negra: Se centra en los requisitos funcionales, permitiendo al ingeniero del software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Los casos de prueba de caja negra pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta, y que se mantiene la integridad de la información externa.

- **Técnica de Prueba de caja negra**

Técnica de la Partición de Equivalencia: Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. En la siguiente tabla se muestran los Casos de Prueba para el caso de uso “Gestionar Reporte”.

Tabla 4: Casos de prueba para el caso de uso “Gestionar reporte”

Nombre de la sección	Escenario de la sección	Descripción de la fundamentación
Adicionar reporte	EC 1.1 Adicionar reporte de forma exitosa.	En este escenario se realiza la adición de un nuevo reporte al sistema correctamente.
	EC 1.2 Faltan datos obligatorios.	En este escenario se realiza la adición de un nuevo reporte al sistema con datos en blanco.

	EC 1.3 Reporte existente.	En este escenario se realiza la adición de un nuevo reporte con un nombre ya existente.
	EC 1.4 Datos incorrectos.	En este escenario se realiza la adición de un nuevo reporte al sistema con datos incorrectos.
Modificar reporte	EC 2.1 Editar reporte de forma exitosa.	En este escenario se modifica un reporte del sistema correctamente.
	EC 2.2 Faltan datos obligatorios.	En este escenario se modifica un reporte del sistema con datos en blanco.
	EC 2.3 Reporte no existente.	En este escenario se modifica un reporte no existente.
	EC 2.4 Datos incorrectos.	En este escenario se modifica un reporte con datos incorrectos.
Eliminar reporte	EC 3.1 Eliminar reporte de forma exitosa.	En este escenario se elimina un reporte del sistema correctamente.
	EC 3.2 Faltan datos obligatorios.	En este escenario se elimina un reporte del sistema con datos en blanco.
	EC 3.3 Reporte no existente.	En este escenario se elimina un reporte no existente.
	EC 3.4 Datos incorrectos.	En este escenario se elimina un nuevo reporte del sistema con datos incorrectos.
Listar reporte	EC 4.1 Listar reporte de forma exitosa.	En este escenario se listan los reportes existentes en el sistema correctamente.
	EC 4.2 Token de usuario incorrecto.	En este escenario se listan los reportes del sistema con datos incorrectos.

Descripción de las variables

En la Tabla 5*: Descripción de las variables* se detallan las variables que se encuentran asociadas al caso de uso “Gestionar reporte”.

Tabla 5: Descripción de las variables

Nº	Nombre del campo	Clasificación	Valor nulo	Descripción
1	token	Campo de texto	No	Cadena de caracteres alfanuméricos, generada por el sistema, único para cada usuario.

2	id	Campo de texto	No	Identificador del reporte.
3	name	Campo de texto	No	Es el nombre que va a tener el reporte.
4	descripcion	Campo de texto	No	Descripción del reporte.
5	jrxml	Campo de texto	No	Reporte.

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema.

Matriz de datos

Los números de las variables corresponden con los números asociados en la Tabla 5*: Descripción de las variables*.

Para especificar si los valores de los campos son correctos o incorrectos se usará:

V: indicador válido.

I: indicador inválido.

Tabla 6: Matriz de datos para el CU Gestionar reportes de la sección Adicionar usuario

Escenario	Variables					Respuesta del sistema	Resultado de la prueba	Flujo central
	1	2	3	4	5			
Adicionar reporte	V		V	V	V	{"success":true,"message":"Report persisted","id":"NUEVO"}	satisfactoria	-
	V			V	V	{"success":false,"error":{"message": "Parámetros nulos","code":10002}	satisfactoria	-
	V		V	V	V	{"success":false,"error":{"message": "Report already exist","code":10001}	satisfactoria	-
	I		V	V	V	{"success":false,"message":"token invalid"}	satisfactoria	-
	V		I	V	V	{"success":false,"error":{"message": "Escriba el nombre correctamente","code":10001}	satisfactoria	-

Resultados

Con el propósito de encontrar posibles fallos en la API de servicios se creó un caso de prueba por cada caso de uso. Se realizaron cuatro iteraciones de pruebas por cada iteración del desarrollo. En la primera

iteración se encontraron siete no conformidades dándole respuesta en su totalidad. Para una segunda iteración se encontraron cuatro no conformidades, las cuales fueron solucionadas completamente. En una tercera iteración se encontraron dos no conformidades dándole respuesta a las dos. En la última iteración no se encontraron no conformidades. En la siguiente figura se representan los resultados en forma de gráfica.

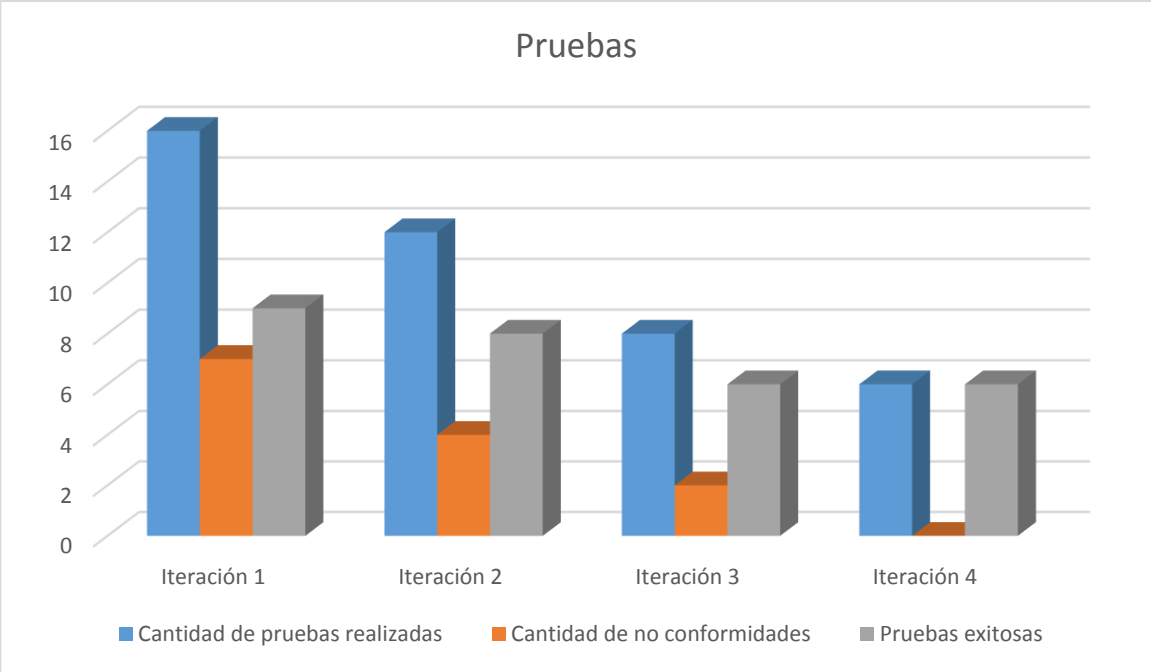


Fig. 16: Resultados de las pruebas

2. **Pruebas de Integración:** Son un conjunto de pruebas unitarias, funcionales, regresión y aceptación que se realizan para probar el software. Incluye también comprobar que lo programado por los diferentes desarrolladores funciona en un entorno real.

Resultado:

Una vez terminada la implementación de la API de servicios web y realizadas las pruebas a nivel de desarrollador se procede a la integración con el proyecto GDR 2.0. A este proyecto se le agrega el SPBundle que es donde están desarrollados todos los servicios web relacionados con la API, además se agrega una nueva entidad a la base de datos TokenUser la cual va a almacenar el token del usuario que este autenticado en el sistema.

Después de integrar este componente a GDR 2.0 se realizaron pruebas funcionales para determinar posibles errores. Obteniendo los resultados correctos de las funcionalidades del sistema.

3. **Prueba de Sistema:** Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.
   - **Tipo de prueba**

Prueba de seguridad: La prueba de seguridad está enfocada al ámbito de la aplicación, asegurando que los usuarios solo accedan cuando estén registrados en el sistema.

Resultados:
Para la realización de las pruebas de seguridad a la API, primeramente, se identificaron los distintos tipos usuarios que están definidos en el sistema y sus respectivos niveles de acceso o privilegios definidos; luego se creó un usuario de cada tipo y se procedió con las pruebas.

- ✓ Las primeras pruebas fueron intentos de violación de la seguridad del sistema, tratando de acceder a éste con usuarios no existentes o utilizando usuarios reales con contraseñas erróneas.

Estas pruebas de acceso arrojaron resultados favorables al sistema, ya que no se logró acceder con usuarios no registrados previamente ni con falsas contraseñas, demostrando que el API cuenta con un mecanismo de autenticación seguro, garantizando que solamente trabajen con el programa personas autorizadas previamente.

- ✓ Se accede al sistema con un usuario específico y se intenta acceder a funciones no permitidas para ese determinado usuario, como son: la gestión de usuarios, gestión de roles y asignar roles.

Este mismo procedimiento se ejecutó con los tres tipos de usuarios con que cuenta la aplicación y los resultados de las pruebas fueron las siguientes. Los usuarios registrados en el sistema 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.

Prueba de rendimiento: La prueba de rendimiento se enfoca en la capacidad de recibir peticiones mediante la utilización de alguna herramienta, verificando cuantas peticiones pueda sostener el sistema sin que este se vea afectado, así como la velocidad de respuesta de este.

Resultados:

Se realizaron pruebas de rendimiento para comprobar la velocidad con la que el sistema ejecuta peticiones de 100 usuarios conectados concurrentemente. Para ejecutar este tipo de prueba se usó la herramienta Apache-JMeter 2.9. Como resultado se obtuvo que los tiempos de espera son de 2.4 segundos, (ver Fig. 17*: Resultado de las pruebas de rendimiento*).

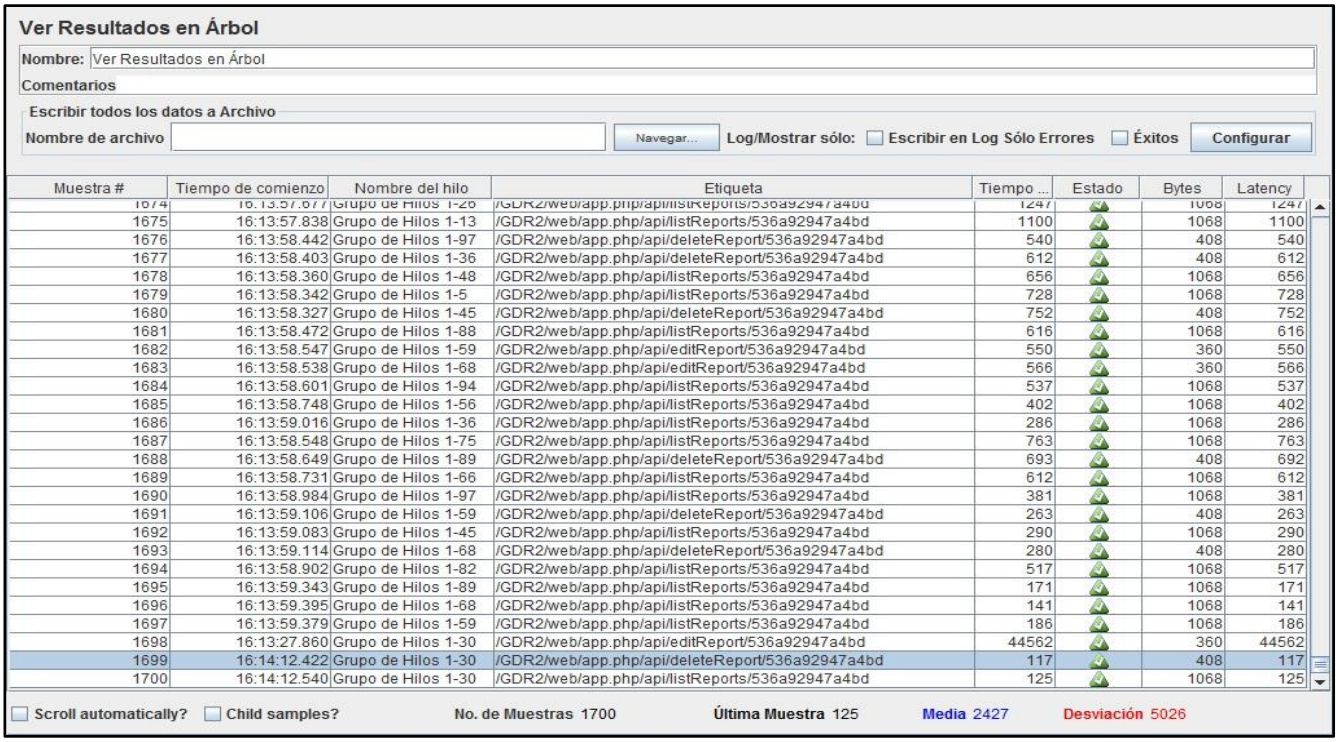
Las prestaciones de hardware que posee la PC cliente y el servidor de la aplicación donde se realizaron las pruebas se muestran a continuación:

PC cliente:

- Ordenador Pentium IV, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM de 256 MB.

PC servidor:

- Ordenador Pentium IV, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM de 1 GB.



Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo ...	Estado	Bytes	Latency
1675	16:13:57.838	Grupo de Hilos 1-13	/GDR2/web/app.php/api/listReports/536a92947a4bd	1100		1068	1100
1676	16:13:58.442	Grupo de Hilos 1-97	/GDR2/web/app.php/api/deleteReport/536a92947a4bd	540		408	540
1677	16:13:58.403	Grupo de Hilos 1-36	/GDR2/web/app.php/api/deleteReport/536a92947a4bd	612		408	612
1678	16:13:58.360	Grupo de Hilos 1-48	/GDR2/web/app.php/api/listReports/536a92947a4bd	656		1068	656
1679	16:13:58.342	Grupo de Hilos 1-5	/GDR2/web/app.php/api/listReports/536a92947a4bd	728		1068	728
1680	16:13:58.327	Grupo de Hilos 1-45	/GDR2/web/app.php/api/deleteReport/536a92947a4bd	752		408	752
1681	16:13:58.472	Grupo de Hilos 1-88	/GDR2/web/app.php/api/listReports/536a92947a4bd	616		1068	616
1682	16:13:58.547	Grupo de Hilos 1-59	/GDR2/web/app.php/api/editReport/536a92947a4bd	550		360	550
1683	16:13:58.538	Grupo de Hilos 1-68	/GDR2/web/app.php/api/editReport/536a92947a4bd	566		360	566
1684	16:13:58.601	Grupo de Hilos 1-94	/GDR2/web/app.php/api/listReports/536a92947a4bd	537		1068	537
1685	16:13:58.748	Grupo de Hilos 1-56	/GDR2/web/app.php/api/listReports/536a92947a4bd	402		1068	402
1686	16:13:59.016	Grupo de Hilos 1-36	/GDR2/web/app.php/api/listReports/536a92947a4bd	286		1068	286
1687	16:13:58.548	Grupo de Hilos 1-75	/GDR2/web/app.php/api/listReports/536a92947a4bd	763		1068	763
1688	16:13:58.649	Grupo de Hilos 1-89	/GDR2/web/app.php/api/deleteReport/536a92947a4bd	693		408	692
1689	16:13:58.731	Grupo de Hilos 1-66	/GDR2/web/app.php/api/listReports/536a92947a4bd	612		1068	612
1690	16:13:58.984	Grupo de Hilos 1-97	/GDR2/web/app.php/api/listReports/536a92947a4bd	381		1068	381
1691	16:13:59.106	Grupo de Hilos 1-59	/GDR2/web/app.php/api/deleteReport/536a92947a4bd	263		408	263
1692	16:13:59.083	Grupo de Hilos 1-45	/GDR2/web/app.php/api/listReports/536a92947a4bd	290		1068	290
1693	16:13:59.114	Grupo de Hilos 1-68	/GDR2/web/app.php/api/deleteReport/536a92947a4bd	280		408	280
1694	16:13:58.902	Grupo de Hilos 1-82	/GDR2/web/app.php/api/listReports/536a92947a4bd	517		1068	517
1695	16:13:59.343	Grupo de Hilos 1-89	/GDR2/web/app.php/api/listReports/536a92947a4bd	171		1068	171
1696	16:13:59.395	Grupo de Hilos 1-68	/GDR2/web/app.php/api/listReports/536a92947a4bd	141		1068	141
1697	16:13:59.379	Grupo de Hilos 1-59	/GDR2/web/app.php/api/listReports/536a92947a4bd	186		1068	186
1698	16:13:27.860	Grupo de Hilos 1-30	/GDR2/web/app.php/api/editReport/536a92947a4bd	44562		360	44562
1699	16:14:12.422	Grupo de Hilos 1-30	/GDR2/web/app.php/api/deleteReport/536a92947a4bd	117		408	117
1700	16:14:12.540	Grupo de Hilos 1-30	/GDR2/web/app.php/api/listReports/536a92947a4bd	125		1068	125

Fig. 17: Resultado de las pruebas de rendimiento

En la siguiente tabla se muestran los tiempos medios de respuesta arrojados por la API de servicios web para GDR 2.0 denotando el tiempo de respuesta en segundos para distintas cantidades de usuarios.

Tabla 7: Estrés del sistema

Cantidad de usuarios	Tiempo de espera (segundos)
50	2.1
100	2.4
150	4.1
180	7.9
200	9.1

Se puede observar que el sistema para una cantidad mayor de 200 usuarios concurrentes tiene un tiempo de respuesta superior a los 9 segundos, el cual está por encima del tiempo permisible por el Departamento Integración de Soluciones que es de 8 segundos.

3.4 Conclusiones del capítulo

En el desarrollo del presente capítulo 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 de seguridad, funcionales e integración para detectar posibles errores en el funcionamiento de la API de servicios web, utilizando el método de caja negra basado en la técnica de partición de equivalencia. Se desarrolló una extensión al iReport para validar el funcionamiento de la solución. Se midió el rendimiento del sistema a través de las pruebas de carga y estrés automatizándolas con la herramienta Apache-JMeter 2.9. Se identificaron 13 no conformidades que fueron resultas satisfactoriamente.

CONCLUSIONES GENERALES

Luego de la realización del presente trabajo de diploma se concluye lo siguiente:

- ✓ A partir de los objetivos específicos se desarrolló la API de servicios web, el cual posibilitó la generación de reportes con GDR 2.0 cuando existen dificultades en la conectividad entre la estación del usuario y el servidor.
- ✓ La fundamentación de la metodología, herramientas y tecnologías empleadas en el desarrollo de la API de servicios web, permitió identificar el ambiente de desarrollo de acuerdo a las necesidades del proyecto.
- ✓ El análisis y diseño de la API de servicios web para GDR 2.0, permitió obtener como resultado los artefactos generados para la implementación.
- ✓ La implementación de las funcionalidades de la API de servicios web viabilizó el cumplimiento de los requisitos funcionales definidos para la solución.
- ✓ El desarrollo de la extensión al iReport validó el funcionamiento de la API de servicios web, demostrando la gestión de reportes en GDR 2.0 desde otro entorno de diseño.
- ✓ La realización de pruebas a la API de servicios web en GDR 2.0, permitió verificar el funcionamiento de la misma.

RECOMENDACIONES

- ✓ Extender la API de servicios web en GDR 2.0 para la gestión de reportes desde diversos entornos de diseño, a partir de otras tecnologías de servicios web como SOAP.
- ✓ Extender otras funcionalidades de GDR 2.0 a servicios web para incrementar la reutilización en otros entornos de diseño.

REFERENCIAS BIBLIOGRÁFICAS

1. AZNAR, Inmaculada [et al.]. *El impacto de las TICs en la sociedad del milenio.* 4, Granada : s.n., 2005, Vol. II. 1695-324X.

2. SAP Crystal Solutions. [En línea] SAP, 2014. [Citado el: 8 de Febrero de 2014.] http://www.crystalreports.com/.

3. Pentaho. [En línea] Pentaho, 2014. [Citado el: 8 de Febrero de 2014.] http://community.pentaho.com/projects/reporting/.

4. Jaspersoft Community. [En línea] Jaspersoft Corporation, 2014. [Citado el: 8 de Febrero de 2014.] http://community.jaspersoft.com/project/ireport-designer.

5. PASCAU, Alexander [et al.]. *Generador De Reportes De Pruebas Hemodinámicas Para El Diagnóstico De Enfermedades Vasculares Periféricas.* Santiago de Cuba : s.n. 2009.

6. ABREU, Aldis Joan [et al.]. *Generador dinámico de reportes.* 11, La Habana : s.n., 2012, Vol. 5. 2343.

7. LÓPEZ, Marleysi. *Glosario de Términos.* La Habana : Generador Dinámico de Reporte. 5107.

8. GARCÍA, Ignacio [et al.]. *E-Business Colaborativo: Cómo implantar software libre, servicios web y el grid computing para ahorrar costes y mejorar las comunicaciones en su empresa.* Madrid : GRAMADOSA, 2003. 84-95428-98-9.

9. ROMANOS, Juan Pablo. Scribd. [En línea] 2014. [Citado el: 29 de Febreo de 2014.] http://es.scribd.com/doc/74942047/Redes-Definicion-y-Tipos.

10. FRANCIS, Pisani y DOMINIQUE, Piolet. *La alquimia de las multitudes.* Barcelona : Paidós Ibérica, 2009. 978-84-493-2196-2.

11. Ecured. [En línea] Marzo de 2010. [Citado el: 17 de Enero de 2014.] http://www.ecured.cu/index.php/API.

12. Reporting Tales. [En línea] [Citado el: 19 de noviembre de 2013.] http://www.on-reporting.com/what-is-pentaho-reporting/.

13. Dataprix. [En línea] [Citado el: 20 de noviembre de 2013.] http://www.dataprix.com/723-caracter-sticas-pentaho.

14. TuERP Software. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.tuerp.com/g/crystal-reports.

15. TINOCO, Oscar [et al.]. Criterios de selección de metodologías de desarrollo de software. [En línea] http://www.redalyc.org/articulo.oa?id=81619984009.

16. BOERAS, Mairelys [et al.]. *Aplicando el método de Boehm y Turner.* 6, La Habana : Ediciones Futuro, 2012, Vol. 5.

17. OpenUp. [En línea] [Citado el: 22 de noviembre de 2013.] http://es.scribd.com/doc/37116717/Open-Up.

18. RUMBAUGH, Jameset [et al.]. *El Lenguaje Unificado de Modelo.* Madrid : addison Wesley, 2000. 84-7829-037-0.

19. Scribd. [En línea] [Citado el: 24 de noviembre de 2013.] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

20. Company, Visual paradigm. [En línea] 2010. [Citado el: 21 de Enero de 2014.] http://www.visual-paradigm.com/features/uml-and-sysml-modeling/.

21. Sistemas gestores de bases de datos. [En línea] [Citado el: 21 de noviembre de 2013.] http://www.csi-csif.es/andalucia/modules/mod_ense/revista/pdf/Numero_30/TERESA_GARZON_1.pdf.

22. PostgreSQL-es. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.postgresql.org.es/node/313.

23. pgAdmin. [En línea] [Citado el: 19 de noviembre de 2013.] http://www.pgadmin.org/index.php.

24. Lenguajes de Programacion. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

25. VÁZQUEZ, Carlos. *Programación en PHP5. Nivel Básico.* Ferrol : s.n., 2008.

26. symfony.es. [En línea] [Citado el: 29 de noviembre de 2013.] http://symfony.es/que-es-symfony.

27. NetBeans. [En línea] [Citado el: 19 de noviembre de 2013.] https://netbeans.org/index_es.html.

28. THOMAS Fielding, Roy . *Architectural Styles and the Design of Network-based Software Architectures.* California : s.n., 2000. 0-599-87118-0.

29. NAVARRO, Rafael. *REST vs Web Services.* 2007.

30. LARMAN, Craig. *UML y Patrones 2da Edición.* Madrid : Prentice Hall, 2003. 8420534382.

31. SOMMERVILLE, Ian. *Ingenería de Software. 6ta edición.* s.l. : Addison Wesley, 2012.

32. *Ingeneria del Software.* Madrid : Pearson, 2005. 84-7829-074-5.

33. Universidad de Alicante. [En línea] [Citado el: 2 de diciembre de 2013.] http://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/.

34. PRITO, Félix. Patrones de diseño. [En línea] 2009. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

35. VISCONTI, Marcello y ASTUDILLO, Hernán. *Fundamentos de Ingenería de software.* Chile : Universidad Técnica Federico Santa María.

36. *Patrones del "Gang of Four".* Madrid : Unidad Docente de Ingeniería del Software , 2009.

37. SOMERVILLE, Ian. Ingenería de software. [En línea] 2010. [Citado el: 18 de Febrero de 2014.] http://www.filecrop.com/ian-sommerville-ingeneria-de-software-septima-edicion.pdf.html.

38. LÓPEZ, Amparo. Guía de estilo para codificación en Java. [En línea] [Citado el: 24 de Marzo de 20013.] http://hp.fciencias.unam.mx/~alg/normas/estilo.html.

39. HERNÁNDEZ, Yasmany. *Estándares de codificación para PHP. Proyecto Generador Dinámico de Reportes.* La Habana : s.n., 2013.

40. GARCÍA, Sheyla y GONZÁLEZ, Félix. *Extensión del NetBeans IDE para el diseño de Interfaces Gráficas de Usuario con Ext JS.* La Habana : s.n., 2013.

BIBLIOGRAFÍA

1. AZNAR, Inmaculada [et al.]. *El impacto de las TICs en la sociedad del milenio.* 4, Granada : s.n., 2005, Vol. II. 1695-324X.

2. SAP Crystal Solutions. [En línea] SAP, 2014. [Citado el: 8 de Febrero de 2014.] http://www.crystalreports.com/.

3. Pentaho. [En línea] Pentaho, 2014. [Citado el: 8 de Febrero de 2014.] http://community.pentaho.com/projects/reporting/.

4. Jaspersoft Community. [En línea] Jaspersoft Corporation, 2014. [Citado el: 8 de Febrero de 2014.] http://community.jaspersoft.com/project/ireport-designer.

5. PASCAU, Alexander [et al.]. *Generador De Reportes De Pruebas Hemodinámicas Para El Diagnóstico De Enfermedades Vasculares Periféricas.* Santiago de Cuba : s.n. 2009.

6. ABREU, Aldis Joan [et al.]. *Generador dinámico de reportes.* 11, La Habana : s.n., 2012, Vol. 5. 2343.

7. LÓPEZ, Marleysi. *Glosario de Términos.* La Habana : Generador Dinámico de Reporte. 5107.

8. GARCÍA, Ignacio [et al.]. *E-Business Colaborativo: Cómo implantar software libre, servicios web y el grid computing para ahorrar costes y mejorar las comunicaciones en su empresa.* Madrid : GRAMADOSA, 2003. 84-95428-98-9.

9. ROMANOS, Juan Pablo. Scribd. [En línea] 2014. [Citado el: 29 de Febreo de 2014.] http://es.scribd.com/doc/74942047/Redes-Definicion-y-Tipos.

10. FRANCIS, Pisani y DOMINIQUE, Piolet. *La alquimia de las multitudes.* Barcelona : Paidós Ibérica, 2009. 978-84-493-2196-2.

11. Ecured. [En línea] Marzo de 2010. [Citado el: 17 de Enero de 2014.] http://www.ecured.cu/index.php/API.

12. Reporting Tales. [En línea] [Citado el: 19 de noviembre de 2013.] http://www.on-reporting.com/what-is-pentaho-reporting/.

13. Dataprix. [En línea] [Citado el: 20 de noviembre de 2013.] http://www.dataprix.com/723-caracter-sticas-pentaho.

14. TuERP Software. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.tuerp.com/g/crystal-reports.

15. TINOCO, Oscar [et al.]. Criterios de selección de metodologías de desarrollo de software. [En línea] http://www.redalyc.org/articulo.oa?id=81619984009.

16. BOERAS, Mairelys [et al.]. *Aplicando el método de Boehm y Turner.* 6, La Habana : Ediciones Futuro, 2012, Vol. 5.

17. OpenUp. [En línea] [Citado el: 22 de noviembre de 2013.] http://es.scribd.com/doc/37116717/Open-Up.

18. RUMBAUGH, Jameset [et al.]. *El Lenguaje Unificado de Modelo.* Madrid : addison Wesley, 2000. 84-7829-037-0.

19. Scribd. [En línea] [Citado el: 24 de noviembre de 2013.] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

20. Company, Visual paradigm. [En línea] 2010. [Citado el: 21 de Enero de 2014.] http://www.visual-paradigm.com/features/uml-and-sysml-modeling/.

21. Sistemas gestores de bases de datos. [En línea] [Citado el: 21 de noviembre de 2013.] http://www.csi-csif.es/andalucia/modules/mod_ense/revista/pdf/Numero_30/TERESA_GARZON_1.pdf.

22. PostgreSQL-es. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.postgresql.org.es/node/313.

23. pgAdmin. [En línea] [Citado el: 19 de noviembre de 2013.] http://www.pgadmin.org/index.php.

24. Lenguajes de Programacion. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

25. VÁZQUEZ, Carlos. *Programación en PHP5. Nivel Básico.* Ferrol : s.n., 2008.

26. symfony.es. [En línea] [Citado el: 29 de noviembre de 2013.] http://symfony.es/que-es-symfony.

27. NetBeans. [En línea] [Citado el: 19 de noviembre de 2013.] https://netbeans.org/index_es.html.

1. AZNAR, Inmaculada [et al.]. *El impacto de las TICs en la sociedad del milenio.* 4, Granada : s.n., 2005, Vol. II. 1695-324X.

2. SAP Crystal Solutions. [En línea] SAP, 2014. [Citado el: 8 de Febrero de 2014.] http://www.crystalreports.com/.

3. Pentaho. [En línea] Pentaho, 2014. [Citado el: 8 de Febrero de 2014.] http://community.pentaho.com/projects/reporting/.

4. Jaspersoft Community. [En línea] Jaspersoft Corporation, 2014. [Citado el: 8 de Febrero de 2014.] http://community.jaspersoft.com/project/ireport-designer.

5. PASCAU, Alexander [et al.]. *Generador De Reportes De Pruebas Hemodinámicas Para El Diagnóstico De Enfermedades Vasculares Periféricas.* Santiago de Cuba : s.n. 2009.

6. ABREU, Aldis Joan [et al.]. *Generador dinámico de reportes.* 11, La Habana : s.n., 2012, Vol. 5. 2343.

7. LÓPEZ, Marleysi. *Glosario de Términos.* La Habana : Generador Dinámico de Reporte. 5107.

8. GARCÍA, Ignacio [et al.]. *E-Business Colaborativo: Cómo implantar software libre, servicios web y el grid computing para ahorrar costes y mejorar las comunicaciones en su empresa.* Madrid : GRAMADOSA, 2003. 84-95428-98-9.

9. ROMANOS, Juan Pablo. Scribd. [En línea] 2014. [Citado el: 29 de Febreo de 2014.] http://es.scribd.com/doc/74942047/Redes-Definicion-y-Tipos.

10. FRANCIS, Pisani y DOMINIQUE, Piolet. *La alquimia de las multitudes.* Barcelona : Paidós Ibérica, 2009. 978-84-493-2196-2.

11. Ecured. [En línea] Marzo de 2010. [Citado el: 17 de Enero de 2014.] http://www.ecured.cu/index.php/API.

12. Reporting Tales. [En línea] [Citado el: 19 de noviembre de 2013.] http://www.on-reporting.com/what-is-pentaho-reporting/.

13. Dataprix. [En línea] [Citado el: 20 de noviembre de 2013.] http://www.dataprix.com/723-caracter-sticas-pentaho.

14. TuERP Software. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.tuerp.com/g/crystal-reports.

15. TINOCO, Oscar [et al.]. Criterios de selección de metodologías de desarrollo de software. [En línea] http://www.redalyc.org/articulo.oa?id=81619984009.

16. BOERAS, Mairelys [et al.]. *Aplicando el método de Boehm y Turner.* 6, La Habana : Ediciones Futuro, 2012, Vol. 5.

17. OpenUp. [En línea] [Citado el: 22 de noviembre de 2013.] http://es.scribd.com/doc/37116717/Open-Up.

18. RUMBAUGH, Jameset [et al.]. *El Lenguaje Unificado de Modelo.* Madrid : addison Wesley, 2000. 84-7829-037-0.

19. Scribd. [En línea] [Citado el: 24 de noviembre de 2013.] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

20. Company, Visual paradigm. [En línea] 2010. [Citado el: 21 de Enero de 2014.] http://www.visual-paradigm.com/features/uml-and-sysml-modeling/.

21. Sistemas gestores de bases de datos. [En línea] [Citado el: 21 de noviembre de 2013.] http://www.csi-csif.es/andalucia/modules/mod_ense/revista/pdf/Numero_30/TERESA_GARZON_1.pdf.

22. PostgreSQL-es. [En línea] [Citado el: 22 de noviembre de 2013.] http://www.postgresql.org.es/node/313.

23. pgAdmin. [En línea] [Citado el: 19 de noviembre de 2013.] http://www.pgadmin.org/index.php.

24. Lenguajes de Programacion. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

25. VÁZQUEZ, Carlos. *Programación en PHP5. Nivel Básico.* Ferrol : s.n., 2008.

26. symfony.es. [En línea] [Citado el: 29 de noviembre de 2013.] http://symfony.es/que-es-symfony.

27. NetBeans. [En línea] [Citado el: 19 de noviembre de 2013.] https://netbeans.org/index_es.html.

28. THOMAS Fielding, Roy . *Architectural Styles and the Design of Network-based Software Architectures.* California : s.n., 2000. 0-599-87118-0.

29. NAVARRO, Rafael. *REST vs Web Services.* 2007.

30. LARMAN, Craig. *UML y Patrones 2da Edición.* Madrid : Prentice Hall, 2003. 8420534382.

31. SOMMERVILLE, Ian. *Ingenería de Software. 6ta edición.* s.l. : Addison Wesley, 2012.

32. *Ingeneria del Software.* Madrid : Pearson, 2005. 84-7829-074-5.

33. Universidad de Alicante. [En línea] [Citado el: 2 de diciembre de 2013.] http://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/.

34. PRITO, Félix. Patrones de diseño. [En línea] 2009. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

35. VISCONTI, Marcello y ASTUDILLO, Hernán. *Fundamentos de Ingenería de software.* Chile : Universidad Técnica Federico Santa María.

36. *Patrones del "Gang of Four".* Madrid : Unidad Docente de Ingeniería del Software , 2009.

37. SOMERVILLE, Ian. Ingenería de software. [En línea] 2010. [Citado el: 18 de Febrero de 2014.] http://www.filecrop.com/ian-sommerville-ingeneria-de-software-septima-edicion.pdf.html.

38. LÓPEZ, Amparo. Guía de estilo para codificación en Java. [En línea] [Citado el: 24 de Marzo de 20013.] http://hp.fciencias.unam.mx/~alg/normas/estilo.html.

39. HERNÁNDEZ, Yasmany. *Estándares de codificación para PHP. Proyecto Generador Dinámico de Reportes.* La Habana : s.n., 2013.

40. GARCÍA, Sheyla y GONZÁLEZ, Félix. *Extensión del NetBeans IDE para el diseño de Interfaces Gráficas de Usuario con Ext JS.* La Habana : s.n., 2013.

41. HERRERA, Cristhian. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=ireport.

42. LABRA, Jose Emilio. [En línea] [Citado el: 27 de noviembre de 2013.] http://di002.edv.uniovi.es/~labra/cursos/XMLAvanzado/ServiciosWeb.html#%281%29.

43. 2ndQuadrant. [En línea] [Citado el: 19 de noviembre de 2013.] http://www.2ndquadrant.com/es/postgresql/caracteristicas-de-postgresql-91.

44. Arquitectura de Software: Estilos y patrones. [En línea] [Citado el: 27 de noviembre de 2013.] http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf.

45. Comparativo de las plataformas j2ee y .net para el desarrollo deservicios web. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.scribd.com/doc/12786595/Articulo-ANALISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET.

46. Curso de Java . [En línea] [Citado el: 27 de noviembre de 2013.] http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm.

47. Definicion abc. [En línea] [Citado el: 17 de noviembre de 2013.] http://www.definicionabc.com/comunicacion/reporte.php.

48. Definicion.DE. [En línea] [Citado el: 17 de noviembre de 2013.] http://definicion.de/plugin/#ixzz2lmo0sNBs.

49. El patrón Modelo-Vista-Controlador (MVC). [En línea] [Citado el: 2 de diciembre de 2013.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

50. eNYe-Sec. [En línea] 12 de marzo de 2010. [Citado el: 17 de noviembre de 2013.] http://www.enye-sec.org/textos/servicios_web.pdf.

51. VISCONTI, Marcello y ASTUDILLO, Hernán. Fundamentos de Ingenieria de Software. [En línea] [Citado el: 2 de diciembre de 2013.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

52. Introducción Open UP. [En línea] [Citado el: 22 de noviembre de 2013.] https\\businesscase.googlecode.com.

53. Manual Generador de Reportes de Pentaho. [En línea] Pentaho Corporation, 2007. [Citado el: 24 de noviembre de 2013.] http://www.onuva.com/wp-content/uploads/2012/07/Manual_PRD.pdf .

54. Microsoft. [En línea] [Citado el: 27 de noviembre de 2013.] http://msdn.microsoft.com/es-es/library/bb972242.aspx.

55. Patrones de Diseño. [En línea] [Citado el: 2 de diciembre de 2013.] http://siul02.si.ehu.es/~alfredo/iso/06Patrones.pdf.

56. php. [En línea] [Citado el: 25 de noviembre de 2013.] http://php.net/.

57. SAD. [En línea] [Citado el: 20 de noviembre de 2013.] http://mistock.lcompras.biz/index.php?option=com_content&view=article&id=1249:ireport&catid=59:tallersoftware&Itemid=116.

58. Todo XML. [En línea] [Citado el: 23 de noviembre de 2013.] https://sites.google.com/site/todoxmldtd/referencia/referencia-de-xml/caracteristicas-de-xml.

59. Up to Down. [En línea] [Citado el: 19 de noviembre de 2013.] http://netbeans-ide.uptodown.com.

60. Visual Paradigm. [En línea] [Citado el: 21 de noviembre de 2013.] www.visual-paradigm.com/.

61. W3C. [En línea] [Citado el: 29 de noviembre de 2013.] http://www.w3.org/.

62. WebSphere. [En línea] IBM. [Citado el: 17 de noviembre de 2013.] http://publib.boulder.ibm.com/infocenter/iadthelp/v7r0/index.jsp?topic=/org.eclipse.jst.ws.consumption.ui.doc.user/concepts/cwsdlud.html.

63. DEITEL, Harvey M. y DEITEL, Paul. *JAVA Como Programar.* Porto Alegre : ARMED editora s.a, 2003. 0-13-034151-7.

64. KABIR, Mohammed J. *La biblia de Servidor Apache2.* Madrid : ANAYA MULTIMEDIA, 2003. 84-415-1468-2.

65. LABRA, Jose Emilio. Introducción a los Servicios Web. [En línea] Octubre de 2006. [Citado el: 23 de Diciembre de 2013.] http://di002.edv.uniovi.es/~labra/cursos/XML/ServiciosWeb.pdf.

66. AMARO, Sarah D., VALVERDE, Jorge C. Metodologías Ágiles. [En línea] [Citado el: 2 de diciembre de 2013.] http://www.sisman.utm.edu.ec/libros/FACULTAD%20DE%20CIENCIAS%20ZOOT%C3%89CNICAS/CARRERA%20DE%20INGENIER%C3%8DA%20EN%20INFORMATICA%20AGROPECUARIA/07/INGENIERIA%20DEL%20SOFTWARE%20I/METODOLOGIAS%20AGILES.pdf.

ANEXOS

Anexo 1: Entrevista al cliente.

Fecha:

Local:

Cliente:

Dirección:

Preguntas:

1. Que desea su empresa/organización que haga el sistema.
2. Cuáles son las funcionalidades que quiere su empresa/organización que se realicen primero teniendo en cuenta la complejidad del negocio.
3. Qué tipo de producto de software le sería más útil (escritorio, web, aplicación de teléfono, etc).
4. Que tecnologías les sería factible a su empresa/organización emplear teniendo en cuenta las licencias del software o si prefiere optar por tecnologías libres.
5. Quiere su empresa/organización que se le incorpore elementos de seguridad. Cuáles serían las condiciones ideales para medir en su ambiente de despliegue teniendo en cuenta la clasificación de la información.
6. Para que condiciones de hardware se desplegaría el producto, existen limitaciones tangibles en cuanto a presupuesto en este tema.
7. Tiene definido alguna prueba de software específica que quieran aplicar al sistema además de las que se seleccionen por equipo para comprobar el funcionamiento del producto.

GLOSARIO DE TÉRMINOS

PDF: Formato de documento portátil, (por sus siglas en inglés de ***Portable Document Format***) es un formato de almacenamiento de documentos.

HTML: Lenguaje de marcado de hipertexto, (por sus siglas en inglés de ***HyperText Markup Language***) es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto.

RTF: formato de texto enriquecido, (por sus siglas en inglés de ***Rich Text Format***) es un formato de archivo informático para el intercambio de documentos multiplataforma. La mayoría de procesadores de texto son capaces de leer y escribir documentos RTF.

XML: lenguaje de marcas extensible, (por sus siglas en inglés de ***eXtensible Markup Language***), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

URL: Un localizador de recursos uniforme, más comúnmente denominado URL (por sus siglas en inglés de ***Uniform Resource Locator***), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación.

CSV: (por sus siglas en inglés de ***Comma-Separated Values***) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla.

MySQL: Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

SQLite: Es un sistema de gestión de bases de datos relacional.

SQL: El lenguaje de consulta estructurado (por sus siglas en inglés de ***Structured Query Language***) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

Oracle: Es un sistema de gestión de base de datos objeto-relacional.

Microsoft SQL Server: Es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional.

IReport: Es una aplicación de código abierto basado en Java que permite a diseñadores y desarrolladores en general diseñar y modelar los reportes visualmente.

API: Interfaz de programación de aplicaciones, es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Microsoft: Es una empresa dedicada al sector de la informática. Microsoft desarrolla, fabrica, licencia y produce software y equipos electrónicos.

IBM: (por sus siglas en inglés de ***International Business Machines***) es una empresa que fabrica y comercializa hardware y software para computadoras.

SAP: Es una empresa de informática alemana que comercializa un conjunto de aplicaciones de software empresarial.

Servlets: Programa que se ejecuta en un servidor para generar páginas web de forma dinámica.

Fig: Figura.