UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La universidad católica de Loja

ÁREA TÉCNICA

TITULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Diseño e implementación de una arquitectura basada en web services RESTFul para garantizar la interoperabilidad semántica e integridad de datos académicos.

TRABAJO DE TITULACIÓN.

AUTOR: Narvaez Peña, Efren Rogelio.

DIRECTOR: Piedra Pullaguari, Nelson Oswaldo, PhD.

LOJA – ECUADOR

2017

Septiembre, 2017

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

PhD.
Nelson Oswaldo Piedra Pullaguari

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: Diseño e implementación de una arquitectura basada en web services RESTFul para garantizar la interoperabilidad e integridad de datos académicos realizado por: Narvaez Peña Efren Rogelio, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, Febrero de 2017

f).....
PhD. Nelson Oswaldo Piedra Pullaguari

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo Narvaez Peña Efren Rogelio declaro ser autor del presente trabajo de titulación: Diseño e implementación de una arquitectura basada en web services RESTFul para garantizar la interoperabilidad e integridad de datos académicos, de la Titulación: Ingeniería en Sistemas Informáticos y Computación, siendo: Nelson Oswaldo Piedra Pullaguari director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo de investigación, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f).....
Autor: Narvaez Peña Efren Rogelio
Cédula: 1105539397

DEDICATORIA

Esta tesis va dedicada a mis padres quienes han sido a lo largo de mi vida el pilar fundamental para mi desarrollo personal y académico.

A mis hermanos y demás familiares por todos sus actos de bondad y palabras de motivación en los momentos en las que verdaderamente las necesitaba.
A mis amigos que a lo largo de los años han estado junto a mi brindándome su amistad incondicional.

AGRADECIMIENTO

A todos los docentes de la Escuela de Sistemas Informáticos y Computación por compartir sus amplios conocimientos durante toda la carrera y como agradecimiento especial al Doctor Nelson Piedra por ser quien dirige el presente trabajo de fin de titulación y por brindarme la oportunidad de aportar en los proyectos del departamento de Tecnologías Avanzadas de la Web.

A la unidad de Gestión de Datos Académicos en especial al ingeniero Jorge Landacay por brindarme su ayuda y apoyo incondicional durante el desarrollo del presente trabajo.

A mis amigos y amigas en especial a Cristian Lluay, Christian Sánchez, Fernando Herrera, Luis Mena, Marvin Jimbo, Daniel Valdivieso, Pablo Bastidas, Katherine Carchi y Lizzette Betancourt por compartir conmigo durante todo el proceso de formación académica sus conocimientos y amistad.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La creciente cantidad de información en las organizaciones ha ocasionado problemas de interoperabilidad obligando así al desarrollo y adopción de nuevas tecnologías que permitan dar solución a estos problemas. Una solución para este problema es emplear el enfoque que brinda la Web Semántica para conseguir una interoperabilidad de datos entre los diferentes sistemas.

Otra de las ventajas que ofrece la Web semántica es la construcción de modelos descriptivos de datos basados en ontologías como estructuras más completas para la descripción de los datos. Esto permite una representación formal de un concepto además de la representación semántica y sintáctica del mismo.

La presente investigación describe el diseño de una arquitectura basada en web services RESTful que garantice la interoperabilidad semántica e integridad de datos académicos de la Universidad Técnica Particular de Loja.

La implementación de la solución propuesta en esta investigación se encuentra en el desarrollo de un prototipo de un RESTful API el cual provee información general acerca de Estudiantes, Docentes, Autoridades y Componentes Académicos. Estos datos se encuentran almacenados en Apache Marmotta una plataforma para datos en RDF.

Palabras Clave: Web Semántica, Datos, RDF, Ontología, RESTful, Web Service, Apache Marmotta.

ABSTRACT

The increasing information quantity in the organizations has caused problems of interoperability forcing this way to the development and adoption of new technologies that allow to give solution to these problems. A solution for this problem is to use the approach that offers to us the Semantic Web to obtain an interoperability of the information between the different systems.

Another advantage that offers us the Semantic Web is the construction of models describing data based on ontologies as structures more complete for the description of the data. This allows a formal representation of a concept as well as the representation of semantics and syntax of the same.

The present research describes the design of an architecture based on web services RESTful that interoperability semantics and data integrity academics of the Private Technical University of Loja.

The implementation of the solution proposed in this research is the development of a prototype of a RESTful API, which provides general information about students, teachers, authorities and academic components. These data are stored in Apache Marmotta a platform for data in RDF.

Keywords: Semantic Web, data, RDF, Ontology, RESTful, Web Service, Apache Marmotta.

INTRODUCCIÓN

Actualmente las organizaciones poseen sistemas informáticos que diariamente alimentan a grandes bases de datos y que con el tiempo estas adquieren grandes volúmenes. Estas grandes bases de datos tienden a convertirse en un gran problema para las organizaciones que no poseen una arquitectura de aplicaciones y de datos adecuada. El manejo de grandes volúmenes de datos alimentados por varios sistemas informáticos conlleva a problemas de redundancia e integridad en los datos, razón por la que se dificultan las tareas de comunicación e intercambio de información entre aplicaciones.

El problema que actualmente le impide a la Universidad Técnica Particular de Loja (UTP) lograr una interoperabilidad de datos entre las diferentes aplicaciones que posee radica en la heterogeneidad de las bases de datos que cada sistema posee y a los grandes volúmenes de datos redundantes y poco consistentes que estas aplicaciones manejan.

La interoperabilidad semántica de datos entre aplicaciones pretende ser la solución a este problema, estableciendo tecnologías, estándares, formatos, reglas y medios de comunicación entre las aplicaciones; es necesario integrar tecnologías de web semántica para evitar problemas de integridad en los datos.

Lo que se plantea es definir una arquitectura de aplicaciones y datos para un repositorio central con la información general de Docentes, Estudiantes, Componentes Académicos y la estructura organizacional de la UTPL (Gobierno UTPL). Este repositorio central almacenara los datos en formato de tripletas transformada a RDF los cuales serán alojada dentro de un triplestore y expuestos a travez de una plataforma para *Linked data* (LDP). Para esto es necesario crear procesos automatizados de extracción, transformación y carga que alimenten a ésta plataforma. El desarrollar un API con una arquitectura basada en servicios RESTful que permita la interoperabilidad de esta plataforma con las distintas aplicaciones que necesiten estos datos.

Los datos con los que se trabajarán son provistos por la base de datos del Sistema de Información Estratégica Centralizada (SIEC) la cual maneja una copia de la información de varios sistemas de la UTPL.

El objetivo es crear un repositorio centralizado con datos generales de contacto e información básica de los componentes académicos que la UTPL oferta cada semestre. En donde este repositorio haga uso de tecnología de web semántica para evitar problemas de redundancia e inconsistencia y que estos datos puedan ser utilizados por el resto de aplicaciones que los necesiten por medio de un RESTful API o SPARQL endpoint y lograr una interoperabilidad de datos entre los principales sistemas como son:

- El Entorno Virtual de Aprendizaje - (EVA).

- El Nuevo Sistema de Gestión Académica - (NSGA).
- Distributivo.
- El Sistema de Infracción Académica Científica - (SIAC).
- FileMaker
- El Sistema de Información Estratégica Centralizada - (SIEC).
- El Sistema de Exalumnos.

OBJETIVOS

Objetivo General:

- Definir e implementar un prototipo funcional que permita garantizar la interoperabilidad semántica e integridad de datos institucionales de la UTPL.

Objetivos Específicos:

- Identificar y describir las principales causas a los problemas de interoperabilidad de las diversas fuentes de datos existentes en la UTPL.
- Definir e implementar una arquitectura robusta y escalable basada en servicios web que pueda hacer frente a los problemas de interoperabilidad identificados.
- Centralizar la información de las fuentes de datos distribuidas y heterogéneas que posean datos de: Docentes, Estudiantes, Gobierno UTPL y Componentes Académicos.
- Utilizar tecnologías de Web Semántica para representar y exponer de forma íntegra la información de: Docentes, Estudiantes, Gobierno UTPL y Componentes Académicos.

CAPÍTULO I: ESTADO DEL ARTE

1.1. Introducción

En este capítulo se abarcan temas relacionados a la integración de datos y las diferentes técnicas y tecnologías utilizadas para lograr dicha integración de las fuentes de información. Se enfatiza de forma especial a la integración semántica de datos y las tecnologías que hacen posible la exitosa implementación de esta técnica. Tecnologías como los servicios web RESTful, Apache Marmota una plataforma para Linked data y todas las tecnologías que hacen posible la web Semántica. Así también un análisis de las diferentes arquitecturas de sistemas existentes para determinar la más adecuada para implementarla en los capítulos siguiente dentro de la solución. Todo este capítulo pretende situar al lector en el contexto actual sobre el que se desarrolla el presente proyecto y de esta forma facilitar la compresión de algunos conceptos y temáticas que se profundizan en los capítulos.

1.2. Integración de datos

La integración de datos involucra la combinación de los datos que residen en diferentes fuentes que se encuentran distribuidas de forma física y lógica; en la mayoría de los casos estas fuentes de datos usan diferentes tipos de sistemas de gestión de base de datos. La integración de datos proporciona a los usuarios una visión unificada de los mismos. Integrar las fuentes de datos se torna relevante en una variedad de situaciones, que incluyen tanto comerciales (cuando dos empresas similares deben fusionar sus bases de datos), científica (por ejemplo, la combinación de los resultados de investigación de diferentes repositorios) y de negocios (Utilización de técnicas de BI para la mejora continua). La integración de datos se ha convertido en el foco de un extenso trabajo teórico, y numerosos problemas aún permanecen abiertos sin resolver.

1.3. El problema de la integración de datos

La integración de múltiples sistemas de información tiene como objetivo la combinación de varios sistemas seleccionados de manera que formen un nuevo conjunto unificado de datos y ofrezca al usuario la ilusión de interactuar con un sistema único de información. Los usuarios disponen de una vista lógica homogénea de los datos que se distribuye físicamente entre más fuentes de datos heterogéneas. Para ello, todos los datos tienen que ser representados usando los mismos principios de abstracción (modelo unificado de datos globales unificadas y semántica). Esta tarea incluye la detección y resolución de conflictos de esquema y datos sobre la estructura y la semántica.
En general, los sistemas de información no están diseñados para la integración. Por lo tanto, cada vez que se desea un acceso integrado a diferentes sistemas, las fuentes de origen y los datos que no encajan entre sí tienen que ser centralizados por procesos adicionales de extracción y carga. Este no es el único problema de la integración, mientras que el objetivo es siempre ofrecer una visión homogénea y unificada de datos de diferentes fuentes, la tarea de

integración en particular puede depender de:

- La arquitectura de un sistema de información
- El contenido y la funcionalidad de los componentes del sistema
- El tipo de información que se gestiona mediante sistemas de componentes (datos alfanuméricos, datos multimedia; estructurados, semi- estructurados de datos no estructurados)
- Requisitos relativos a la autonomía de los componentes del sistemas
- Uso previsto del sistema integrado de información (de sólo lectura o acceso de escritura)
- Requisitos de rendimiento, y los recursos disponibles (tiempo, dinero, recursos humanos, know-how, etc.)

Además, varios tipos de heterogeneidad típicamente tienen que ser considerados. Estos incluyen las diferencias en:

- Hardware y sistemas operativos,
- El software de gestión de datos,
- Modelos de datos, esquemas, y la semántica de datos
- *Middleware* o lógica de intercambio de información entre aplicaciones.
- Interfaces de usuario, y reglas de negocio.(Ziegler & Dittrich, 2010)

1.4. Enfoques para la integración de datos

Aplicando una perspectiva arquitectónica para dar una visión general de las diferentes formas de abordar el problema de la integración. La clasificación presentada se basa en Building the Information Society de (Jacquart, 2015) y distingue la integración en enfoques de acuerdo con el nivel de abstracción en el que se realiza la integración.
Los sistemas de información pueden ser descritos mediante una arquitectura en capas, como se muestra en la Figura 1.1: En la capa superior, los usuarios tienen acceso a datos y servicios a través de diversas interfaces que se ejecutan en la parte superior de aplicaciones diferentes. Las aplicaciones pueden utilizar *middleware* - monitores de procesamiento de transacciones (TP), *middleware messageoriented* (MOM), *SQL-middleware*, etc. - a los datos de acceso a través de una capa de acceso a datos. Los datos en sí son administrados por un sistema de almacenamiento de datos. Por lo general, los sistemas de gestión de bases de datos (DBMS) se utilizan para combinar el acceso a los datos y la capa de almacenamiento.

En general, el problema de la integración se puede tratar en cada una de las capas del sistema presentados. Para esto, los siguientes enfoques principales están disponibles:

Integración manual Los usuarios interactúan directamente con todos los sistemas de información pertinentes e integran manualmente los datos seleccionados. Es decir, los usuarios tienen que hacer frente a diferentes interfaces de usuario y lenguajes de consulta. Además, los usuarios necesitan tener un conocimiento detallado de la ubicación, la representación de datos lógicos y la semántica de datos.

Interfaz de usuario común En este caso, el usuario se suministra con una interfaz de usuario común (por ejemplo, un navegador web) que proporciona una apariencia uniforme. Los datos de los sistemas de información pertinentes todavía se presentan por separado de modo que la homogeneización e integración de datos aún tiene que ser hecho por los usuarios (por ejemplo, como en los motores de búsqueda).

Integración de aplicaciones Este enfoque utiliza aplicaciones de integración que tienen acceso a diversas fuentes de datos y devuelven resultados integrados para el usuario. Esta solución es práctica para un pequeño número de sistemas de componentes. Sin embargo las aplicaciones se vuelven cada vez más grandes como el número de interfaces de sistemas y formatos de datos para homogeneizar e integrar.

Integración con middleware *Middleware* proporciona funcionalidad reutilizable que generalmente se utiliza para resolver aspectos dedicados al problema de la integración, por ejemplo, como se ha hecho por SQL-*middleware*. Mientras que las aplicaciones están exentos de la aplicación de la funcionalidad común de integración, aún se necesitan esfuerzos de integración en aplicaciones. Además, diferentes herramientas de *middleware* por lo general tienen que combinarse para construir sistemas integrados.

Acceso de datos uniforme En este caso, una integración lógica de los datos se lleva a cabo en el nivel de acceso a datos. Aplicaciones globales están provistos de una visión global unificada de datos distribuidos físicamente, aunque sólo de datos virtual está disponible en este nivel. sistemas de información locales mantienen su autonomía y pueden soportar capas de acceso a datos adicionales para otras aplicaciones. Sin embargo, el suministro mundial de datos integrados físicamente puede llevar mucho tiempo ya que el acceso de datos, la homogeneización y la integración tienen que ser hechas en tiempo de ejecución.

Almacenamiento Central de datos Aquí la integración de datos física se realiza mediante la transferencia de datos a un nuevo almacenamiento de datos. Fuentes locales pueden o bien ser retirados o permanecen operativos. En general la integración física de datos proporciona acceso rápido a datos. Sin embargo si se retiran las fuentes de datos locales las aplicaciones que acceden a ellos tienen que ser migrado al nuevo almacenamiento de datos.

1.5. Integración semántica

La tecnología de base de datos se introdujo en las empresas desde finales de 1960 para soportar aplicaciones de negocio (inicialmente más simples). A medida que el número de aplicaciones y repositorios de datos creció rápidamente la necesidad de datos integrados se hizo evidente. Como consecuencia de ello, la primera integración se acercó en forma de sistemas multibase y se desarrollaron en torno a 1980.(Hurson & Bright, 1991)

Ésta fue una primera piedra en una notable historia de la investigación en el área de integración de datos. La continua evolución de estas técnicas por los mediadores como por ejemplo, (Carey et al., 1995) y los sistemas de agentes basados en ontologías, *peer-to-peer (P2P)*, y los enfoques de integración basados en servicios web(Franklin et al., 2005). En general los enfoques de integración primero se basan en un modelo de datos relacional o funcional y se dieron cuenta de las soluciones una vez acoplados herméticamente, proporcionando un único esquema global. Para superar sus limitaciones relativas a los aspectos de la abstracción, clasificación y taxonomías, los enfoques de integración orientada a objetos (Bukhres & Elmagarmid, 1996) se adoptaron para realizar la homogeneización estructural y la integración de los datos. Con la llegada de Internet y las tecnologías web, la atención se centró en la integración de datos puramente bien estructurados para incorporar también los datos no estructurados.

Sin embargo, la integración es algo más que un problema estructural o técnico. Técnicamente, es bastante fácil conectar diferentes DBMS relacional (a través de ODBC o JDBC). Más exigente es la integración de los datos descritos por los modelos de datos diferentes; aún peor son los problemas causados por los datos con la semántica heterogéneos. Por ejemplo, tener sólo el nombre de “pérdida” para denotar una relación en un sistema de información de la empresa no proporciona información suficiente para decidir, sin duda, si la pérdida que representa una pérdida de libros, una pérdida realizada, o una futura pérdida esperada y si los valores de las tuplas que reflejan sólo una pérdida más o menos estimada o una pérdida cuantificada con precisión. La integración de dos relaciones “pérdida” con (implícitas) semántica heterogénea conduce a resultados erróneos y conclusiones completamente sin sentido. Por lo tanto, la semántica explícita y que precisa integrables de los datos son esenciales para la integración. Tenga en cuenta que ninguno de los principales enfoques de integración ayudan a resolver la heterogeneidad semántica; ni el XML que solo proporciona información estructural de una solución.

En el área de las base de datos, la semántica puede considerarse como la interpretación de las personas de los datos y esquemas, de acuerdo con su comprensión del mundo en un contexto determinado. En la integración de datos, el tipo de semántica considerada generalmente, es la semántica del mundo real, y que se ocupa del ”mapeo de objetos del mundo real al modelo de objetos en el mundo computacional “las cuestiones que implican la interpretación

humana, el significado y uso de los datos y la información "(Aris M. Ouksel, 2010). En esta configuración, la integración semántica es la tarea de agrupación.

Combinar o completar datos de diferentes fuentes, teniendo en cuenta la semántica de datos explícita y precisa, con el fin de evitar que los datos semánticamente incompatibles se fusionen estructuralmente. Es decir, la integración semántica tiene que asegurarse de que sólo los datos relacionados con la misma entidad real o concepto se fusionen. Un requisito previo para esto es resolver la ambigüedad referente a los meta-datos de la información a integrar. Estos deben ser claros para obtener todos los supuestos implícitos relevantes e información de contexto subyacente.

Una idea para superar la heterogeneidad semántica en el área de la base de datos es especificar de forma exhaustiva la semántica del mundo real, previstos de todos los datos y elementos del esquema. Por desgracia es imposible definir completamente lo que un elemento de datos o esquema denota o significa en el mundo de base de datos (Navathe et al., 2000). Por lo tanto los esquemas de bases normalmente no proporcionan suficiente semántica explícita para interpretar los datos consistentemente y sin ambigüedad (Sheth & Larson, 1990). Estos problemas se agravan aún más por el hecho de que la semántica pueden ser incluida en los modelos de datos y esquemas conceptuales. Por otra parte no hay semántica absoluta que sea válida para todos los usuarios potenciales. Estas dificultades relativas a la semántica son la razón de muchos retos para que la investigación todavía esté abierta en el área de integración de datos.

Ontologías Definidas como descripciones explícitas y formales de los conceptos y sus relaciones que existen en un cierto universo de discurso, junto con un vocabulario común para referirse a estos conceptos - pueden contribuir a resolver el problema de la heterogeneidad semántica. En comparación con otros sistemas de clasificación, como las taxonomías, tesauros, o palabras clave, las ontologías permiten modelos más precisos de dominio (Khalid et al., 2014). Con respecto a una ontología de un grupo de usuarios particular, se compromete en la semántica del proveedor de las fuentes datos de datos para que la integración puede hacerse explícita. Basándose en esta comprensión compartida, el peligro de la heterogeneidad semántica puede ser reducido. Por ejemplo, las ontologías se pueden aplicar en el ámbito de la Web Semántica para conectar de forma explícita información de los documentos de Internet para su definición y el contexto en forma procesable por máquina; por lo tanto, los servicios semánticos, como la recuperación de documentos semántica, puede ser proporcionada.

En la investigación de base de datos, modelos y ontologías de dominio único se aplicaron primero en superar la heterogeneidad semántica. Como menciona Yigsl Arens en (ARENS et al., 2003), un modelo de dominio se utiliza como una sola ontología a la que se asignan los contenidos de las fuentes de datos. Por lo tanto, se les puede pedir que las consultas se expresen en términos de la ontología global. En general, los enfoques de una sola ontología

son útiles para los problemas de integración en todas las fuentes de información, para ser integrados proporcionan casi el mismo punto de vista sobre un dominio (Wache et al., 2001). En caso de que los puntos de vista de dominio de las fuentes difieren, la búsqueda de un punto de vista común se hace difícil. Para superar este problema, multi-ontología se acerca como OBSERVADOR (Mena et al., 2013) y describen cada fuente de datos con su propia ontología; a continuación, estas ontologías locales tienen que ser asignadas, ya sea para una ontología global o interactúen entre unas y para establecer un entendimiento común.

1.6. Interoperabilidad

"Definir su nivel de madurez de interoperabilidad de datos, ampliar sus capacidades y desarrollar un portafolio de datos, pueden ayudar a orientar a las organizaciones en su camino hacia la omnipresencia en el intercambio de información".(Janssen et al., 2014)

Aun más complejo que la integración *ad hoc de software y hardware* compatible, la interoperabilidad de datos requiere de la capacidad para lograr un acceso continuo a los datos , combinar y procesar datos procedentes de múltiples fuentes.

Desde una perspectiva de sistemas, el IEEE define interoperabilidad como "la capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información que ha sido intercambiada."(Hilliard, 1991) Al nivel de datos, donde la interoperabilidad es muy importante para la creación de aplicaciones basadas en la información; la interoperabilidad es la capacidad de dos o más conjuntos de datos para ser enlazados, combinados y procesados.

1.6.1. Los niveles

La tecnología móvil, las arquitecturas orientadas a servicios, la Web semántica y las tecnologías de servicios Web han simplificado enormemente la tareas de interoperabilidad, según (Hilliard, 1991) la interoperabilidad comúnmente ocurre en cuatro niveles:

Interoperabilidad Técnica: Se refiere a la conectividad de la red, lo que asegura que los sistemas están conectados a través de redes cableadas o inalámbricas. Ahora los sistemas pueden conectarse en cualquier momento y desde cualquier lugar, permitiendo el intercambio de datos en tiempo real. Vincular diferentes sistemas operativos y aplicaciones de software escritos en diferentes lenguajes de programación puede lograrse mediante la adopción de las tecnologías de servicios Web.

Interoperabilidad Sintáctica: Se refiere a la aplicación y la adhesión a normas y estándares que aseguren que los datos estructurados pueden ser intercambiados sobre la infraestructura de operatividad técnica que requiere adherirse al mismo formato de datos para cumplir con el intercambio eficiente de datos a través estándares predefinidos, a menudo a través de tecnologías de servicios Web y XML.

Interoperabilidad Semántica: Asegura que la información se interprete de la misma manera. La creación de ontologías y el uso de tecnologías semánticas para razonar acerca de las ontologías permiten la interoperabilidad semántica. Esto requiere metadatos que describan lo que tratan los datos, quién los publicó, su calidad, y los posibles usos o combinaciones de usos con otros datos. Aunque este último no suele incluirse en los metadatos, tener conocimiento es esencial para su uso.

Interoperabilidad Pragmática: Se refiere a todos los aspectos organizativos y de colaboración de calidad y confianza, a servicios y acuerdos de disponibilidad, y a la sensibilidad del sentido del contexto.

1.7. Servicios web

Conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios los solicitan llamando a estos procedimientos a través de la Web. A su vez proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar. La figura 1 muestra como una aplicación cliente es capaz de interoperar con varios sistemas terrígenos con la ayuda de una serie de servicios web.

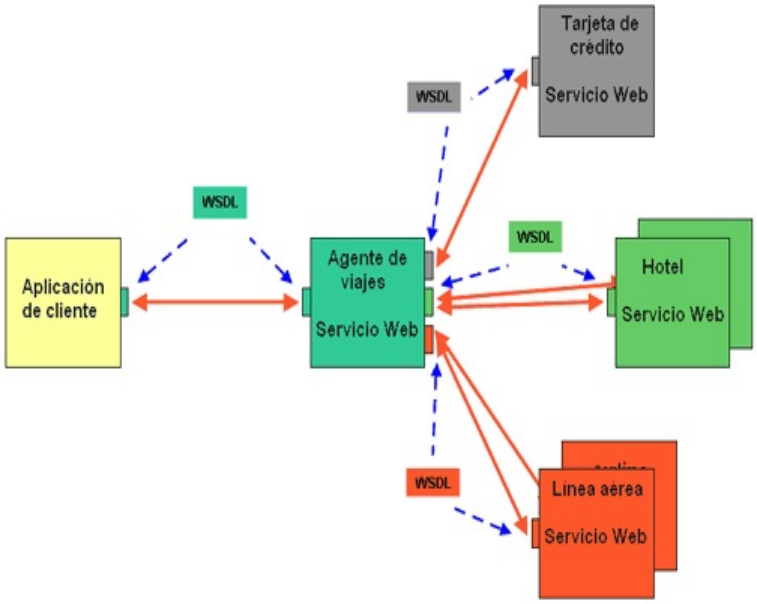


Figura 1: Interacción del cliente con varios web services
Fuente: Recuperado de: https://goo.gl/fv03ZY
Elaboración: (Morales, 2014)

Web Services (WS) ofrece un significado estándar para interoperar entre diferentes aplicaciones de software corriendo en diferentes plataformas y/o marcos de trabajo. El W3C pretende diseñar la arquitectura, definirla y crear el núcleo de tecnologías que hagan posible los Servicios Web. Esta arquitectura se basa en los siguientes componentes:

- Diseñar un marco de mensajería:

  Simple SOAP: *Simple Object Access Protocol* es un protocolo simple para intercambiar información estructurada en un ambiente descentralizado y distribuido. *"Messaging Framework"* define usando tecnologías XML, un marco extensible de mensajería que contiene una construcción del mensaje que se pueda intercambiar con una variedad de protocolos subyacentes

  Web Services Addressing (WS-Addressing): Direccionamiento de Servicios Web. La dirección de los servicios Web proporciona mecanismos neutrales para transportar los servicios web y los mensajes. Define un sistema de características abstractas y una representación de XML para referirse a servicios de la Web y para facilitar la dirección final de los mensajes. Esta especificación permite a los sistemas de mensajería soportar la transmisión del mensaje a través de redes que incluyen el procesado de nodos tales como gestión final, cortafuegos y pasarelas mediante una forma de transporte neutro.

  SOAP Message Transmission Optimization (MTOM) Descripción de la Optimización de la Transmisión del Mensaje. Describe una característica abstracta y una puesta en práctica concreta para optimizar el formato de la transmisión y/o de la vía de los mensajes SOAP.

- Descripción de los Servicios:

  Web Services Description Language (WSDL): Lenguaje de Descripción de los Servicios Web. Se trata de un lenguaje para describir Servicios Web. La especificación define el lenguaje básico que puede usarse para describir servicios Web basados en un modelo abstracto de lo que ofrece el servicio. También define los criterios de conformidad de los documentos en relación a este lenguaje.

  Web Services Choreography Description Language (WS-CDL): Lenguaje de Descripción de la Coreografía de los Servicios Web. Es un lenguaje basado en XML que describe colaboraciones *peer to peer* de los participantes, definiendo desde un punto de vista global un comportamiento observable común y complementario; donde ordenado el mensaje, intercambia el resultado de acuerdo a un objetivo de negocios común.

Los servicios web que se basan en XML permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado dichas aplicaciones e independientemente del sistemas operativo o plataforma en que se ejecuten y de los dispositivos utilizados en el acceso.

1.8. REST

REpresentational State Transfer es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.

REST permite la creación de servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP, por lo que es increíblemente más simple y convencional que otras alternativas que se han usado en los últimos diez años como SOAP y XML-RPC.

REST se definió en el 2000 por Roy Fielding, coautor principal también de la especificación HTTP. Podríamos considerar REST como un *framework* para construir aplicaciones web respetando HTTP.

Por lo tanto REST es el tipo de arquitectura más natural y estándar para crear APIs para servicios orientados a Internet.

Existen tres niveles de calidad a la hora de aplicar REST en el desarrollo de una aplicación web y más concretamente una API que se recogen en un modelo llamado Richardson Maturity Model en honor al tipo que lo estableció, Leonard Richardson padre de la arquitectura orientada a recursos. Estos niveles son:

1. Uso correcto de URIs.
2. Uso correcto de HTTP.
3. Mensajes Autodescriptivos.

Además de estas tres reglas, nunca se debe guardar estado en el servidor, toda la información que se requiere para mostrar la información que se solicita debe estar en la consulta por parte del cliente.

REST al ser un protocolo sin estado brinda altas prestaciones, eliminando problemas con temas como el almacenamiento de variables de sesión e incluso podemos jugar con distintas tecnologías para servir determinadas partes o recursos de una misma API.

1.8.1. Nivel 1: Uso correcto de URIs

Las URL, (*Uniform Resource Locator*) , son un tipo de URI, (*Uniform Resource Identifier*), que además de permitir identificar de forma única el recurso, permite localizarlo para poder acceder a él o compartir su ubicación.

Una URL se estructura de la siguiente forma:

(protocolo):///(dominio o hostname)[:puerto (opcional)]/ruta del recurso?(consulta de filtrado)

REST plantea un serie de reglas básicas para nombrar a la URI de un recurso:

- Los nombres de URI no deben implicar una acción, por lo tanto debe evitarse usar verbos en ellos.
- Deben ser únicas, no debemos tener más de una URI para identificar un mismo recurso.
- Deben ser independiente de formato.
- Deben mantener una jerarquía lógica.
- Los filtrados de información de un recurso no se hacen en la URI.

1.8.2. Nivel 2: HTTP

Al adoptar REST se deben tener claros los siguientes aspectos:

- Métodos HTTP.
  - GET: Para consultar y leer recursos
  - POST: Para crear recursos
  - PUT: Para editar recursos
  - DELETE: Para eliminar recursos.
  - PATCH: Para editar partes concretas de un recurso.
- Códigos de estado.
- Deben mantener una jerarquía lógica.
- Aceptación de tipos de contenido.

1.8.3. Mensajes Autodescriptivos

Los mensajes que se intercambian entre el cliente y el servidor pueden ser personalizados a través de MIME type, es decir se especifica en el MIME type el tipo de respuesta y que tipo se desea obtener.

1.9. Linked data

1.9.1. Principios de Linked data.

La importancia y facilidad que Tim Berners-Lee da a los principios de *Linked data* permiten e invitan al usuario a contribuir y publicar sus datos, aspecto tan relevante para ir construyendo la Web de datos, conforme a nuestras necesidades y en prioridad a la información que se

maneja. Aplicar estos principios esenciales y obligatorios permite impulsar el crecimiento de la Web.

Utilizando como referencia el artículo publicado por Tim Berners-Lee en julio del 2006, es conveniente complementar estos principios o reglas básicas con un mayor detalle:

Utilice URIs como nombre para las cosas: Al nombrar los conceptos mediante URIs, estamos ofreciendo una abstracción del lenguaje natural para así evitar ambigüedades y brindar una forma estándar y univoca para referirnos a cualquier recurso.

Usar HTTP URIs para que la gente pueda buscar los nombres: Se debe usar URIs sobre HTTP para asegurar que cualquier recurso pueda ser buscado y accedido en la Web. Se debe tener en cuenta que los URIs no son sólo direcciones, son identificadores de los recursos.

Proveer información útil utilizando las normas (RDF. SPARQL): Cuando se busca y accede a un recurso identificado mediante una URI HTTP, se debe obtener información útil sobre dicho recurso representada mediante descripciones estándares en RDF. Se pretende que para cualquier conjunto de datos o vocabulario se ofrezca información relativa a la información que representa.

Incluye enlaces a otros URI, para que puedan descubrir más cosas: Esta regla es necesaria para enlazar datos que se encuentran en la Web, de tal manera que no se queden aislados y así poder compartir la información con otras fuentes externas y que otros sitios puedan enlazar sus propios datos de la misma forma que se hace con los enlaces en HTML.

Generalmente se otorga una igualdad en conceptos entre URL y URI, es necesario establecer la diferencia entre URI (Uniform Resource Identifier) es una secuencia compacta de caracteres que identifica un recurso abstracto o físico teniendo una sintaxis y un proceso para resolver éstas. (Berners-Lee, 2005); Y URL (Uniform Resource Locator) el mismo que sirve como medio de localización y acceso a los recursos en el mundo de Internet.

1.9.2. Beneficios de *Linked data*.

Luego de haber revisado historia, estructura y principios sobre *Linked data* llegamos al punto en donde la pregunta es ¿cuales son los beneficios que otorga a sus usuarios como: investigadores, desarrolladores y usuarios finales?. En relación a esto y usando como referencia lo que menciona (W3C Oficina Española, 2009) podemos citar algunos de los beneficios de los datos enlazados:

- La web de datos puede ser rastreada por los enlaces RDF.

- Contiene mecanismos de acceso único y estandarizado.
- Facilita el trabajo de los motores de búsqueda.
- Accede mediante navegadores de datos genéricos.
- Vinculación de los datos de diferentes fuentes.

1.9.3. Proceso de publicación de datos

Publicar lo datos implica seguir un proceso que permita de una forma organizada aportar con información valiosa y estructurada a la Web Semántica, para tomar en consideración la publicación de datos es necesario plantearse las siguientes preguntas ¿Con qué cantidad de Datos se cuenta? ¿Cómo se almacenaran los Datos? ¿Cómo actualmente se encuentran los datos a vincular?

Mediante estos enfoques adoptamos un panorama mas claro de que vamos a publicar conociendo el ámbito de trabajo, hacia quienes van dirigidos estos datos y quienes van a reutilizar la información. Luego de pensar en estos ambientes se puede tomar en consideración las respuestas a las preguntas citadas anteriormente. (W3C Oficina Española, 2010)

Boris Villazón menciona en el artículo *"Methodological Guidelines for Govermment Linked data"* que es necesario conocer que el proceso de publicación de Datos tiene un ciclo de vida. La figura 2 muestra el esquema gráfico del ciclo de publicación de datos enlazados:

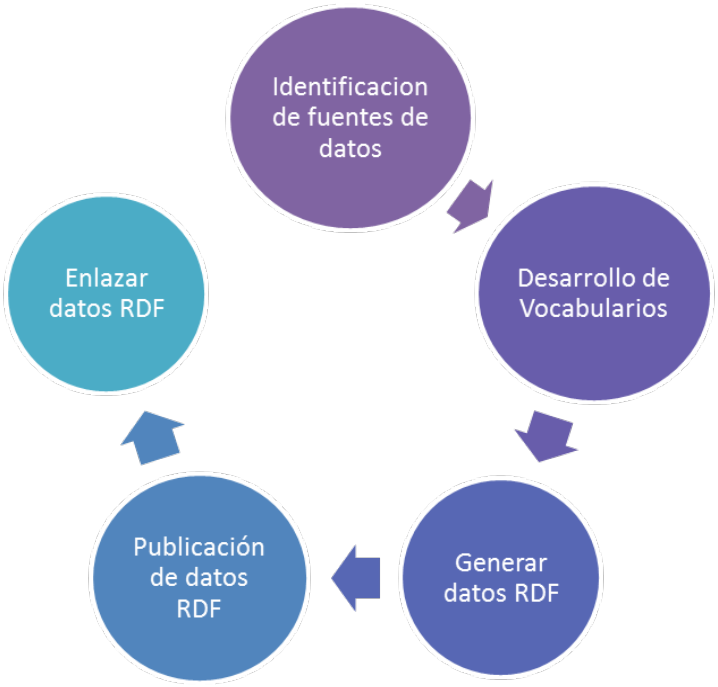


Figura 2: Proceso de publicación de datos.
Fuente: Recuperado de: https://goo.gl/oW9uuW
Elaboración: (Villazón-Terrazas et al., 2014)

Identificación de fuentes de datos: Identificar y seleccionar las fuentes de datos que queremos publicar. Aportar a un proyecto que ya tiene su data publicada o tomar data particular para posteriormente ser publicada. Se responde preguntas como: ¿Dónde están los datos?, ¿en qué formato?, ¿qué tipo de repositorio?, es decir se necesita conocer el entorno de trabajo a utilizar.

Desarrollo de vocabularios: se recomienda la reutilización de vocabularios existentes para un rápido desarrollo de ontologías, esto permite una reducción de tiempo, esfuerzo y recursos.

Generar datos RDF: se toman las fuentes de datos identificadas previamente y se la transforma a formato RDF utilizando los vocabularios establecidos, de esta forma se cumple con los principios de *Linked data*.

Publicación de datos RDF: Se necesita almacenar y publicar la data en *triplestore* (base de datos para el almacenamiento en tripletas RDF).

Limpieza de datos: es importante encontrar los posibles errores. Según(Hogan et al., 2014) identificar los errores comunes como por ejemplo; errores de accesibilidad, tipos de datos malformados o incompatibles, o términos de vocabularios mal definidos.

Enlazar datos RDF: Se aplica el cuarto principio de *Linked data* "incluir enlaces con otras URIS" para lograr interconectar con datasets externos.

Habilitar un descubrimiento efectivo: Habilitar el efectivo descubrimiento y sincronización de los dataset publicados, mantener la data actualizada y en lo posible que buscadores como Google o Sindice sean capaces de localizarla. (Villazón-Terrazas et al., 2014)

Para la creación de aplicaciones de software es necesario el uso o aplicación de metodologías de desarrollo ya sean estas tradicionales o ágiles, que permitan alcanzar los objetivos propuestos del proyecto, la función de estas metodologías es brindar una guía o marco de trabajo a todos los involucrados en el proyecto y de esta forma establecer los pasos que deben seguir para obtener un proyecto final de calidad; de igual forma la Web Semántica establece una serie de metodologías que señalan cuales son los pasos a seguir para la publicación de información en la nube de datos enlazados siguiendo una serie de pasos establecidos dentro de un ciclo.

1.9.4. Tecnologías de la web semántica

Para poseer una correcta operación de los datos se necesita de tecnologías que hagan posible la web semántica, como los lenguajes para la representación de ontologías, lenguajes de consulta, entornos de desarrollo, módulos de gestión para ontologías (almacenamiento, acceso, actualización), módulos que permitan la visualización; es importante mencionar cada una de ellas y los conceptos que manejarán para tener un panorama claro de su funcionamiento.

1.9.5. Representación la información

Siguiendo los conceptos establecidos por la (W3C Oficina Española, 2009) y menciona que la representación de la información en la web se hace posible gracias al lenguaje RDF *(Resource Description Framework)*, es una base para procesar metadatos; proporciona interoperabilidad entre aplicaciones que intercambian información legible por máquinas en la Web. RDF destaca por la facilidad para habilitar el procesamiento automatizado de los recursos Web.

El objetivo general de RDF es definir un mecanismo para describir recursos que no cree ninguna asunción sobre un dominio de aplicación particular ni defina (a priori) la semántica de algún dominio de aplicación. La definición del mecanismo debe ser neutral con respecto al dominio, sin embargo el mecanismo debe ser adecuado para describir información sobre cualquier dominio. El modelo de datos básico consiste en tres tipos de objetos:

Recursos: todas las cosas descritas por expresiones RDF se denominan recursos. Un recurso puede ser una parte de una página Web, una colección completa de páginas web, un sitio Web completo, un libro impreso. Los recursos se designan siempre por URIs.

Propiedades: una propiedad es un aspecto específico, característica, atributo, o relación utilizado para describir un recurso. Cada propiedad tiene un significado específico, define sus valores permitidos, los tipos de recursos que puede describir y sus relaciones con otras propiedades.

Sentencias: Un recurso específico junto con una propiedad denominada, más el valor de dicha propiedad para ese recurso es una sentencia RDF.

Estas tres partes se denominan respectivamente sujeto, predicado y objeto. El objeto de una sentencia (el valor de la propiedad) puede ser otro recurso o puede ser un literal; es decir, un recurso (especificado por una URI) o una cadena simple de caracteres u otros tipos de datos primitivos definidos por XML.

Consideremos como ejemplo la sentencia: Efren Narvaez es el creador (autor) del recurso http://www.w3.org/Home/Efren.

La tabla 1 muestra las partes que comprenden la sentencia anterior:

Tabla 1: Sentencia de Ejemplo

Sujeto(Recurso)	http://www.w3.org/Home/Efren
Predicado(Propiedad)	Creator
Objeto (literal)	“Efren”

Fuente: El Autor
Elaboración: El Autor

Para representar gráficamente esta sentencia se usa diagrama de nodos y arcos. En estos gráficos; los nodos (óvalos) representan recursos y los arcos representan propiedades denominadas. Los nodos que representan cadenas de literales pueden dibujarse como rectángulos. La sentencia citada anteriormente se representaría gráficamente como lo muestra la figura 3.

Figura 3: Representación en diagrama de nodos de la sentencia.
Fuente: El Autor
Elaboración: El Autor

Nota: La dirección de la flecha es importante. El arco siempre empieza en el sujeto y apunta hacia el objeto de la sentencia.

RDF necesita también una sintaxis concreta para crear e intercambiar metadatos. Esta especificación de RDF utiliza XML [Lenguaje de Marcado extensible].

La figura 4 muestra una representación en XML/RDF de la sentencia anterior.

```
<rdf:RDF>
 <rdf:Description about="http://www.w3.org/Home/Efren">
  <s:Creator>Efren</s:Creator>
 </rdf:Description>
</rdf:RDF>
```

Figura 4: Sentencia representada en XML/RDF
Fuente: El Autor
Elaboración: El Autor

1.9.6. Explotación de datos RDF

Como menciona en su presentación (Qaissi, 2009) al hablar de un formato de socialización no tan solo es hablar de poder enlazar los datos en la web y poder ser hallados por medio de buscadores, es hablar de como recuperar los datos en la web tomando en consideración necesidades específicas como:

- Los datos en RDF no servirían de nada si no se pueden utilizar.
- Los lenguajes de la Web Semántica necesitan interactuar con los datos almacenados en la “base de datos” RDF.
- Necesidad parecida al lenguaje SQL de bases de datos relacionales.

SPARQL *(Simple Protocol and RDF Query Language)* es una lenguaje de consulta del ámbito de la Web Semántica de W3C. En otras palabras define la sintaxis y la semántica necesarias para una expresión de consulta sobre un grafo RDF y las diferentes formas de resultados obtenidos.

Su misión es devolver todas las tripletas o componentes solicitados basándose en la comparación de una tripleta pasada como parámetro de la consulta (grafo básico) con todas las tripletas que componen el grafo RDF. Cabe recalcar que sus sintaxis son similares al estándar SQL de bases de datos relacionales. Las consultas SPARQL cubren tres objetivos:

- Extraer información en forma de URIs y literales.
- Extraer sub-estructuras RDF
- Construir nuevas estructuras RDF partiendo de resultados de consultas. (Corcho & Gómez, 2010)

Al tener la información almacenada en formato RDF es posible realizar consultas utilizando SPARQL. La Figura 5 muestra un ejemplo de una consulta SPARQL “Se desea buscar el nombre de grupos de música heavy de los años 80” y la tabla 2 muestra el resultado obtenido.

```
PREFIX esdbpp: <http://es.dbpedia.org/property/>
PREFIX esdbpr: <http://es.dbpedia.org/resource/>
SELECT ?grupo
WHERE{
  ?grupo  rdf:type  dbpedia-owl:MusicalArtist .
  ?grupo  dbpedia-owl:activeYearsStartYear ?inicio .
  ?grupo  dbpedia-owl:activeYearsEndYear   ?fin .
  ?grupo  esdbpp:estilo  esdbpr:Heavy_metal .
  FILTER ((?inicio > "1980-01-01T00:00:00Z" ^^ xsd:dateTime
  && ?inicio < "1990-01-01T00:00:00Z"^^xsd:dateTime) || (?fin >
 "1980-01-01T00:00:00Z"^^xsd:dateTime && ?fin
 < "1990-01-01T00:00:00Z"^^xsd:dateTime ) || (?inicio < "1980-01-01T00:00:00Z"
 ^^xsd:dateTime && ?fin > "1990-01-01T00:00:00Z"^^xsd:dateTime ) )
}ORDER BY DESC(?inicio) LIMIT 10
```

Figura 5: Interacción del cliente con varios web services
Fuente: El Autor
Elaboración: El Autor

Tabla 2: Resultado de la consulta hacia DBpedia

GRUPO
http://es.dbpedia.org/resource/Bonham
http://es.dbpedia.org/resource/Argus (banda)
http://es.dbpedia.org/resource/Logos (banda)
http://es.dbpedia.org/resource/Shotgun Messiah
http://es.dbpedia.org/resource/Reverend
http://es.dbpedia.org/resource/Vago (banda argentina)
http://es.dbpedia.org/resource/Seo Taiji
http://es.dbpedia.org/resource/Mother Love Bone
http://es.dbpedia.org/resource/Saigon Kick
http://es.dbpedia.org/resource/Tad

Fuente: (DBpedia, 2016)
Elaboración: El Autor

Gracias a la Web y de una manera progresiva se ha logrado eliminar barreras de comunicación, comercio, y acceso a la información; beneficiando directamente a los usuarios en cada una de las actividades que realizan, otorgándole el poder de descubrir nuevas cosas. Si remontamos hacia una década atrás en el entorno una moda era tener una cuenta de Email, o ser parte de una sala de chat; se denomina un lujo tener servicio de Internet; actualmente y gracias al avance tecnológico tener estos servicios se ha convertido en una necesidad, debido a diversos factores sociales, económicos, educativos, entre otros.

Linked data aporta una perspectiva completamente nueva y estructurada de como manejar la web usando como línea base el conocimiento, mejorar la capacidad de búsqueda de contenido, brindarle al usuario información útil y referente a su necesidad, y no podemos dejar de lado la importancia de mantener un nivel de estructura formal, de igual forma cumplir con los principios establecidos, seguir la metodología de publicación de datos, todo esto en beneficio del contenido que será agregado para finalmente ser parte de esta nueva ventana de conocimiento.

1.10. Apache Marmotta

Apache Marmotta es un servidor de *Linked data* (LD), servidor de SPARQL y un entorno de desarrollo para *Linked data*. Marmotta ofrece una Plataforma de *Linked data* (PLD) para el acceso a los datos de lectura-escritura legible por la máquina a través de la negociación de contenido HTTP. Marmotta cuenta con módulos y bibliotecas para el desarrollo de aplicaciones *Linked data*. La arquitectura de servidor modular hace posible la implementación de las funcionalidades requeridas solamente. Marmotta también ofrece una colección de bibliotecas de *Linked data* para las tareas comunes tales como el acceso a los recursos *Linked data* y consulta *Linked data* (a través de LIBPATH, un simple lenguaje de consulta *Linked data*). El triplestore está segregado desde el servidor, por lo que se puede utilizar de forma independiente. Se trata de una arquitectura orientada a servicios utilizando Contextos y la inyección de dependencias (CDI), un conjunto de servicios de desarrollo de aplicaciones web Java. Marmotta Core, un componente fundamental de Apache Marmotta, proporciona acceso a *Linked data*, la funcionalidad para realizar la importación y exportación de RDF y una interfaz de administración. Marmotta Core une el servicio y la dependencia de la inyección, el triplestore, la configuración del sistema y el registro.

1.10.1. Módulo SPARQL

Este módulo proporciona en la plataforma apoyo a SPARQL 1.1. Más precisamente, la Recomendación 11 que componen SPARQL 1.1, Marmotta cubre 9 de ellos (sólo Entailment Regimes y las consultas federadas aún no son compatibles).
El módulo SPARQL ofrece una interfaz unificada tanto para la consulta y actualización basada en Squebi y visualización de datos utilizando Sgvizler.

La figura 6 muestra la interfaz web de usuario que el modulo de Apache Marmotta posee.

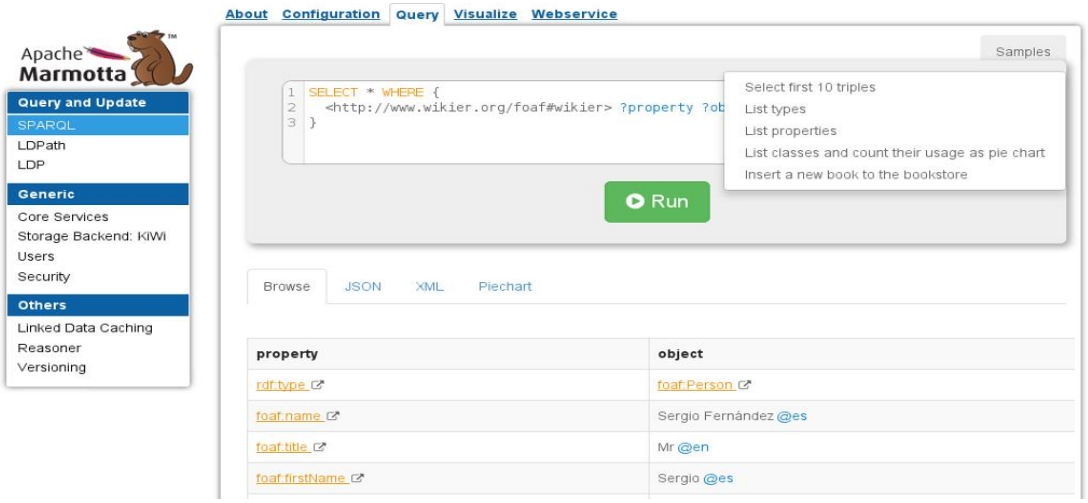


Figura 6: Interfaz gráfica del módulo SPARQL
Fuente: Recuperado de: `http://marmotta.apache.org`
Elaboración: El Autor

1.10.2. Módulo LDP

1.10.2.1. ¿Qué es LDP?

Linked Data Platform Es un conjunto de mejores prácticas y un enfoque simple para una arquitectura de *Linked data* de lectura-escritura, basados en el acceso HTTP a los recursos web que describen su estado utilizando el modelo de datos RDF.

Linked Data Platform se construye sobre una jerarquía o niveles en los cuales se manejan un tipo de recurso en especifico pudiendo ser estos RDF o NonRDF los cuales son almacenados en contenedores. La figura 7 muestra una ilustración de como cada nivel gestiona o maneja un tipo de recurso en especifico.

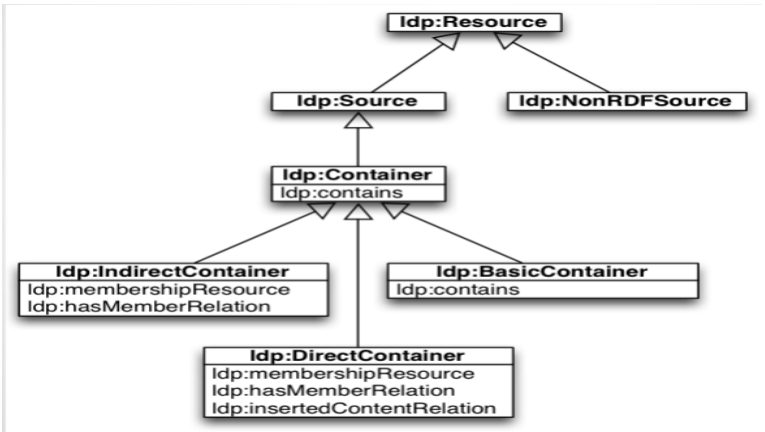


Figura 7: Jerarquía del LDP Marmotta
Fuente: Recuperado de: `https://goo.gl/WtQtGO`
Elaboración: (Fernández, 2014)

Un servidor de LDP gestiona dos tipos de los recursos:

Aquellos recursos cuyo estado se representa mediante RDF (LDP-RSS) y los que utilizan otros formatos (LDP-NR).

La figura 8 muestra una ilustración de los tipos de datos que un LDP maneja.

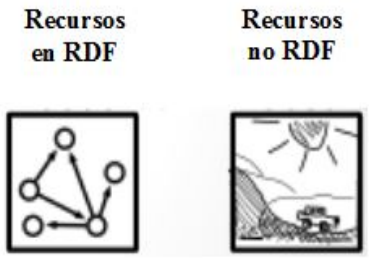


Figura 8: Tipos de Datos
Fuente: Recuperado de:
https://goo.gl/WtQtGO
Elaboración: (Fernández, 2014)

Estos servidores no imponen ninguna restricción a LDPRs y generalmente actúan como sistemas de almacenamiento sin ninguna lógica de la aplicación específica de dominio y vocabularios.

1.10.3. Módulo LDPath

LDPath es un lenguaje simple de consulta basado en path-based, similar a XPath o SPARQL, que está particularmente bien adaptado para realizar consultas y recuperación de recursos provenientes de la nube de datos enlazados, siguiendo los enlaces entre servidores y recursos RDF.

1.10.3.1. Protocolo

La petición al LDPATH se expone a través de un protocolo REST simple, utilizando JSON para la descripción de los resultados. Los programas pueden ser evaluados utilizando cualquiera de estos dos métodos (GET o POST), donde el contexto de URI debe ser siempre enviada como parámetro de consulta.

1.10.3.2. Módulos

LDPath integra una serie de submódulos que se agrupan según su función como se detallara a continuación.

1.10.3.3. Mulos del Core

Los siguientes módulos son los módulos básicos del LDPATH y se necesarios en cada situación:

ldpath-api: contiene interfaces utilizadas por el lenguaje; necesario para extensiones personalizadas

ldpath-core: contiene el core, el Backend independiente del lenguaje de LDPath, incluyendo un analizador y un evaluador; está a la espera de alguna aplicación backend para estar presente.

1.10.3.4. Módulos de backend

Apache Marmotta ofrece una serie de backends que están listos para su uso en cualquier proyecto en el que se desee implementar. La implementación de un backend suele ser sencillo e implica principalmente la interfaz RDFBackend. RDFBackend hace uso de Java Generics, por lo que son capaces de utilizar siempre el modelo de datos de forma directa. Los siguientes backends son proporcionados por la distribución:

ldpath-backend-sesame: Es una implementación genérica para los repositorios Sesamo.

ldpath-backend-jena: Es una implementación genérica para los modelos de Jena.

ldpath-backend-file: Backend basado en archivos permitiéndole consultar el contenido de un archivo RDF.

ldpath-backend-linkeddata: Una aplicación backend sofisticada que realiza consultas sobre recursos en la Nube de *Linked data* y luego las cachea.

1.10.3.5. Módulos de extensión

Apache Marmotta una serie de módulos de extinción en donde alguno de ellos son parte de otros proyectos como es el caso de: Linked Media Framework o Apache Stanbol.

Parte del proyecto en sí son actualmente los siguientes módulos:

ldpath-template: Implementa una extensión del motor de plantillas FreeMarker que permite la construcción de las plantillas con declaraciones LDPATH para insertar e iterar sobre los valores este módulo se caracteriza por ser independiente.

ldpath-template-linkeddata: Es una implementación de backend para las plantillas de LDPATH lo que permite realizar consultas sobre la Nube de datos enlazados; proporciona solo una herramienta de línea de comandos para el procesamiento de las plantillas.

1.10.4. Módulo de seguridad

Este módulo proporciona mecanismos de seguridad para Apache Marmotta, implementa su propio mecanismo de autenticación y autorización.

1.10.4.1. Usuarios y Roles

Existen dos tipos de usuarios predeterminados en Marmotta: anónimo y admin. El primero no es un usuario real, pero este usuario puede acceder a todas las solicitudes anónimas. El segundo es el usuario con derechos de administración del sistema.

Al mismo tiempo, se aplican un grupo de reglas para simplificar la administración de permisos de los usuarios. El sistema viene con tres grupos (director, editor y usuario) de forma predeterminada, pero esto puede ser personalizado según se prefiera.

1.10.4.2. Perfiles

Dentro de Marmotta existen tres perfiles predefinidos:

Simple: permite el acceso de lectura de todas partes y el acceso de escritura sólo de interfaces locales *localhost* u otros.

Estándar: permite el acceso de lectura de todas partes y el acceso de escritura sólo para usuarios autenticados con el rol de ”gerente”.

Por defecto: Marmotta utilizará el perfil simple, permitiendo sólo el acceso desde *localhost*. Si desea cambiar el modo, se puede establecer el security.profile propiedad de configuración estándar.

1.10.4.3. Reglas

Las reglas o configuraciones se basa en listas de control de acceso (ACL) bajo una serie de parámetros que las determinan.

La figura 9 muestra un ejemplo de esta configuración.

```
security.{TYPE}.{NAME}.pattern = {PATTERN}
security.{TYPE}.{NAME}.methods = {METHOD}
security.{TYPE}.{NAME}.priority = {PRIORITY}
```

Figura 9: ACL Marmotta
Fuente: Recuperado de: https://goo.gl/xdXzVq
Elaboración: El Autor

La siguiente lista describe cada uno de los parámetros que componen las reglas.

TYPE es el tipo de control que puede ser el permiso para la concesión de la autorización de las solicitudes que cumplan esa norma o restricción para restringir.

NAME es una etiqueta arbitraria para nombrar la regla que debe ser única en combinación con el tipo.

PATTERN es el patrón de expresión regular que coincide con esta regla.

METHOD es el método HTTP esta regla se aplica (HEAD, OPTIONS, GET, POST, PUT o DELETE). Si las reglas se aplican a más de un método, puede agregar todos los métodos separadas por comas o añadir líneas de propiedad adicionales para cada método, las dos opciones son válidas.

PRIORITY es la prioridad de esta norma de la lista de control de acceso.

El sistema evalúa las reglas ordenadas por prioridad, permitiendo o rechazando el acceso cada vez que una regla coincide con cada solicitud en el sistema. La interfaz de usuario de administración proporciona una página de perfiles para el estado de las actuales normas aplicadas al sistema.

1.10.5. Módulo de versionamiento

Marmotta ofrece un servicio de control de versiones que realiza un seguimiento de cambios triples en gráficos configurables. Versionamiento basado en transacción, es decir, cada vez que se confirma una transacción con éxito, una nueva entrada de versión es creada por los servicios de control de versiones. Marmotta actualmente soporta versiones del listado (TimeMap), así como la inspección de instantáneas (mementos) (es decir, que se remonta en el tiempo) por los recursos. Para dar un acceso web sin problemas a timemaps y snapshots. El módulo de control de versiones Marmotta implementa el protocolo Memento (RFC 7089).

Para hacer uso del servicio de control de versiones que tienes que insertar la dependencia al módulo marmotta-versionado en el archivo pom.xml del plattform Marmotta.

La figura 10 muestra la porción de código que se debe adicionar al archivo pom.xml.

```
<dependency>
    <groupId>org.apache.marmotta</groupId>
    <artifactId>marmotta-versioning</artifactId>
    <version>3.3.0</version>
</dependency>
```

Figura 10: Código de Inserción de dependencias del módulo
Fuente: Recuperado de: https://goo.gl/xdXzVq
Elaboración: El Autor

Los recursos son los recursos originales. Al solicitar esos recursos, la respuesta HTTP incluye enlaces a la TimeMap (una lista de todas las versiones de los recursos) y a timegate (donde

se puede solicitar el recurso con cabecera Accept-Datetime)

.

La figura 11 muestra como el LDP gestiona el flujo de Peticiones HTTP.

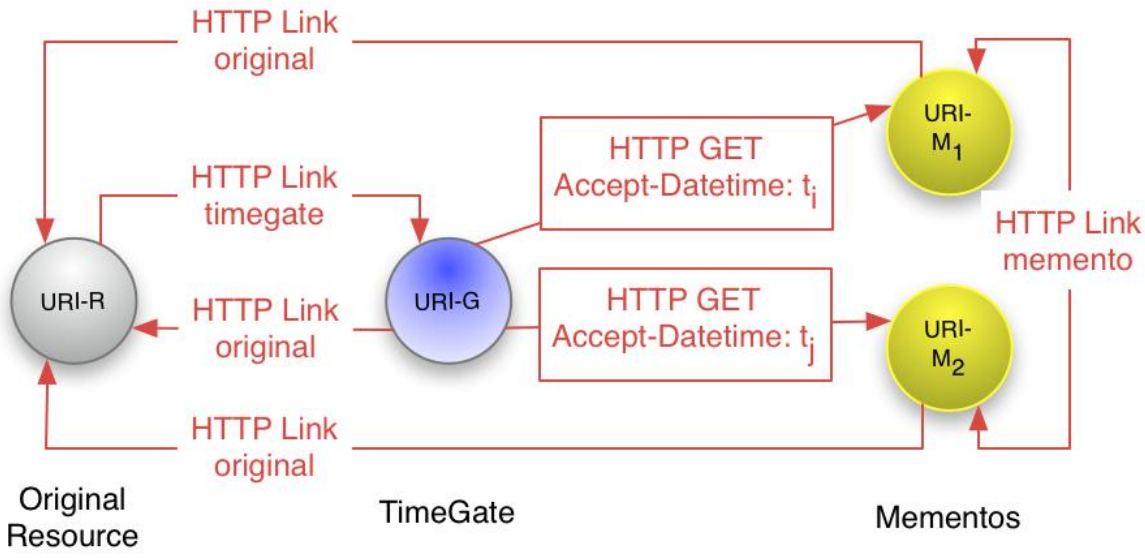


Figura 11: Flujo de Peticiones
Fuente: Recuperado de: https://goo.gl/xdXzVq
Elaboración: El Autor

TimeGate soporta negociación de contenido en la dimensión de fecha y hora. Al negociar con el TimeGate, el cliente HTTP utiliza una cabecera Accept-Datetime para enviar un enlace de redirección para la versión solicitada. Versiones de recursos en Memento tiene URI bookmarkable, para que puedas conectar directamente a un Memento.

Memento ofrece algunos eslabones de la cabecera de la respuesta HTTP. Incluye enlaces a las primeras versiones (rel = primera memento), la última versión (rel = último memento), el memento actual (rel = memento), la versión anterior (rel = prev recuerdo) y la próxima versión (rel = próximo recuerdo). Todos los enlaces a Recuerdos también incluyen información de fecha y hora.

TimeMap proporciona una lista de enlaces a todos los recuerdos de un determinado recurso. Por el momento, TimeMaps son capaces de entregar 2 formatos diferentes (application/link-format and text/html) en función de la cabecera HTTP Accept petición.

1.11. Arquitectura de software

De acuerdo al Software Engineering Institute (SEI), la Arquitectura de Software se refiere a las estructuras de un sistema, compuesta de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos según el autor (Bass et al., 2013). El término “elementos” dentro de la definición del SEI, puede referirse a distintas entidades relacionadas con el sistema; los elementos pueden ser entidades que existen en tiempo de ejecución

(objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Por otro lado, las relaciones entre elementos dependen de propiedades visibles de los elementos, quedando ocultos detalles de implementación. Finalmente el conjunto de elementos relacionados de un tipo particular corresponde a una estructura distinta, de ahí que la arquitectura está compuesta por distintas estructuras.

(Klein et al., 1999) señala que la arquitectura de software es importante como una disciplina debido a que los sistemas de software crecen, de tal forma que resulta muy complicado que sean diseñados, especificados y entendidos por un solo individuo. Uno de los aspectos que motivan el estudio en este campo es el factor humano en términos de inspecciones de diseño, comunicación a alto nivel, desarrollo, reutilización de componentes y comprobación a alto nivel de diseños alternativos.

La arquitectura de software, es el proceso de definición de una solución estructurada que cumple con todos los atributos de calidad, tales como el rendimiento, seguridad y facilidad administración. Se trata de una serie de decisiones basadas en una amplia gama de factores en que cada una de estas decisiones puede tener impacto considerable en la calidad, el funcionamiento, el mantenimiento y el éxito en general de la aplicación.

1.11.1. Importancia de la arquitectura de software

Al igual que cualquier otra estructura compleja, el software debe ser construido sobre una base sólida y estructurada, el no considerar escenarios claves por la falta de diseño o en su defecto despreciar consecuencias a largo plazo, pueden poner en riesgos las aplicaciones de software. Los riesgos expuestos por la mala arquitectura incluyen software inestable e incapaz de ser apoyo a las soluciones de software en la empresa, de abordar futuros requisitos o es difícil implementar y gestionar en un entorno de producción.

Según el autor (Bass et al., 2013) “la arquitectura de software es de especial importancia ya que la manera en que se estructura un sistema tiene un impacto directo sobre la capacidad de este para satisfacer lo que se conoce como atributos de calidad del sistema”. Atributos de calidad como el desempeño, que tiene que ver con el tiempo de respuesta del sistema en las peticiones que se hacen, usabilidad, que tiene que ver con qué tan sencillo les resulta a los usuarios realizar operaciones en el sistema, o bien la modificabilidad, que tiene que ver con qué tan simple resulta introducir cambios en el sistema. Los atributos de calidad son parte de los requerimientos no funcionales del sistema y representan características que deben expresarse de forma cuantitativa.

En opinión de estos autores, se puede concluir que la arquitectura de software permitirá comprender, organizar y comunicar las estructuras que componen un sistema, con el propósito de satisfacer los atributos de calidad como: desempeño, usabilidad, seguridad, además de servir de guía para el desarrollo de la solución software. Con la arquitectura de software se define de

manera abstracta los componentes de un sistema, los procesos de información, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser implementada en una arquitectura física, que consiste en determinar en qué ordenador o componente físico tendrá asignada cada tarea de ejecución. Consideraciones Claves para el diseño de una arquitectura de software Estilos de arquitectura de Software Comparativa de arquitecturas.

1.11.2. Selección de arquitectura

Tomando en cuenta la Tabla 5, donde se realiza la comparación entre los estilos de arquitectura de software seleccionados como: cliente-servidor, basado en componentes y en capas, a continuación se realiza una valoración por cada estilo arquitectónico de acuerdo a los atributos de calidad detallados a continuación.

Escalabilidad: Es la capacidad de expandirse en usuarios o incrementar la capacidad del sistema sin realizar cambios.

Rendimiento: Se mide en término de la respuesta del sistema a ciertas funcionalidades como pueden ser la velocidad de respuesta al recibir una petición o procesar una información.

Seguridad: Es la característica que evita el acceso no autorizado o accidental de usuarios. Es medido en función de la probabilidad de daño o riesgo a la seguridad, número o porcentaje de daños clasificados por tipo y severidad.

Eficiencia: Es una medida de la eficiencia en el uso de los recursos del sistema y se mide en términos de uso de memoria, ancho de banda, espacio en disco o disponibilidad de capacidad del procesar durante las operaciones.

Disponibilidad: Está relacionada con la habilidad de acceder al sistema bajo factores que lo afectan durante el respaldo, recuperación o reinicio y se mide como el porcentaje del tiempo en que el sistema puede estar disponible.

Flexibilidad: Es la capacidad de adaptación para aumentar, extender o expandirse con usuarios adicionales. Es medido en función del esfuerzo, duración o costo de agregar o modificar componentes específicos.

La tabla 3 muestra el cumplimientos de las características descritas anteriormente en cada estilo de arquitectura de software.

Tabla 3: Valoración de los estilos de arquitectura de software.

Estilos Arquitectonicos	Disponibilidad	Escalabilidad	Flexibilidad	Rendimiento	Seguridad	Eficacia
Cliente - Servidor	X	-	X	-	X	-
Basadas En Componentes	X	X	X	-	-	X
Arquitectura En Capas	X	X	X	X	X	X

Fuente: (Conery, 2009)
Elaboración: El Autor

1.12. Arquitectura 3 capas web

La arquitectura en capas es en realidad un estilo de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio, mecanismos de almacenamiento.

La arquitectura por capas es una de las técnicas más comunes que los arquitectos de software que utilizan para dividir sistemas de software complicados. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior.

Es por esto que la programación por capas es una arquitectura que separa en tres niveles la programación de un sistema en capa de presentación, capa de negocios y capa de datos. Una de las ventajas es que el desarrollo se puede llevar a cabo en varios niveles y en caso de ocurrir algún cambio en los requerimientos del usuario, solo se realiza el mantenimiento al nivel correspondiente, según el autor (Arenas Paredes, 2011), como lo muestra la Figura 12.

Figura 12: Arquitectura en tres capas
Fuente: Basado en: https://goo.gl/ShMKIo)
Elaboración: (Paredes, 2011)

Capa de presentación: En la capa de presentación se encuentra todo el diseño del sistema, es decir la parte que el usuario visualiza, la cual sirve de medio de comu-

nicación para receptar y capturar la información.

En esta capa no se desarrolla ninguna validación sobre las funcionalidades que presente la aplicación, ya que se encargará únicamente de la presentación de la información, la cual debe tener la característica de ser amigable, entendible y fácil de uso para el usuario.

Capa de negocio: En esta capa se ubica la parte lógica, donde se ejecutan las peticiones del usuario y se envían las respuestas tras el proceso; por lo que se podría decir que es una vía de comunicación entre la IU y la lógica de negocios.

Capa de datos: En esta capa residen los datos y es la encargada de acceder a los mismos, es decir que se reciben los argumentos y se devuelven los resultados por medio de la capa correspondiente hacia los usuarios finales.

1.13. Modelo de 4+1 Vistas

La arquitectura del software se trata de abstracciones, de descomposición y composición, de estilos y estética. También tiene relación con el diseño y la implementación de la estructura de alto nivel del software.

Los diseñadores construyen la arquitectura usando varios elementos arquitectónicos elegidos apropiadamente. Estos elementos satisfacen la mayor parte de los requisitos de funcionalidad y performance del sistema, así como también otros requisitos no funcionales tales como confiabilidad, escalabilidad, portabilidad y disponibilidad del sistema. La figura 13 muestra una representación gráfica del Modelo de 4+1 Vistas.

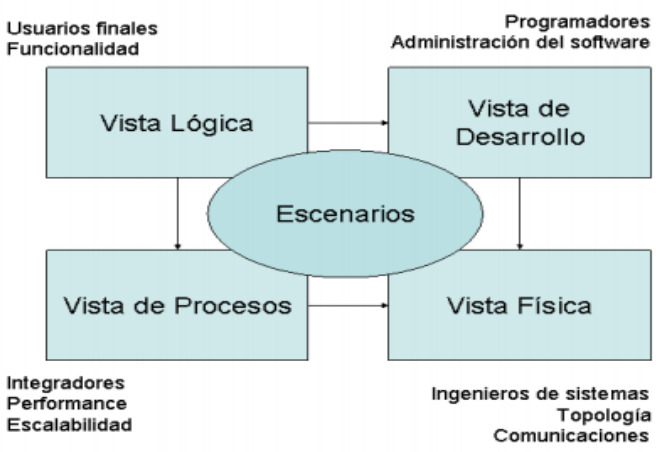


Figura 13: Modelo de 4+1 Vistas
Fuente: Recuperado de: https://goo.gl/vkY93n
Elaboración: (Kruchten, 1995)

1.13.1. La Vista lógica

La arquitectura lógica apoya principalmente los requisitos funcionales –lo que el sistema debe brindar en términos de servicios a sus usuarios. El sistema se descompone en una serie de abstracciones clave, tomadas (principalmente) del dominio del problema en la forma de objetos o clases de objetos. Aquí se aplican los principios de abstracción, encapsulamiento y herencia. Esta descomposición no solo se hace para potenciar el análisis funcional, sino también sirve para identificar mecanismos y elementos de diseño comunes a diversas partes del sistema

1.13.2. La Vista de procesos

La arquitectura de procesos toma en cuenta algunos requisitos no funcionales tales como la performance y la disponibilidad. Se enfoca en asuntos de concurrencia y distribución, integridad del sistema, de tolerancia a fallas. La vista de procesos también especifica en cual hilo de control se ejecuta efectivamente una operación de una clase identificada en la vista lógica

La arquitectura de procesos se describe en varios niveles de abstracción, donde cada nivel se refiere a distintos intereses. El nivel más alto de la arquitectura de procesos puede verse como un conjunto de redes lógicas de programas comunicantes (llamados ”procesos”) ejecutándose en forma independiente, y distribuidos a lo largo de un conjunto de recursos de hardware conectados mediante un bus, una LAN o WAN. Múltiples redes lógicas pueden usarse para apoyar la separación de la operación del sistema en línea del sistema fuera de línea, así como también para apoyar la coexistencia de versiones de software de simulación o de prueba.

Un proceso es una agrupación de tareas que forman una unidad ejecutable. Los procesos representan el nivel al que la arquitectura de procesos puede ser controlada tácticamente (i.e., comenzar, recuperar, reconfigurar, y detener). Además, los procesos pueden replicarse para aumentar la distribución de la carga de procesamiento, o para mejorar la disponibilidad.

1.13.3. La Vista de desarrollo

La vista de desarrollo se centra en la organización real de los módulos de software en el ambiente de desarrollo del software. El software se empaqueta en partes pequeñas –bibliotecas de programas o subsistemas– que pueden ser desarrollados por uno o un grupo pequeño de desarrolladores. Los subsistemas se organizan en una jerarquía de capas, cada una de las cuales brinda una interfaz estrecha y bien definida hacia las capas superiores

La vista de desarrolla tiene en cuenta los requisitos internos relativos a la facilidad de desarrollo, administración del software, reutilización y elementos comunes, y restricciones impuestas por las herramientas o el lenguaje de programación que se use. La vista de desarrollo apoya

la asignación de requisitos y trabajo al equipo de desarrollo, y apoya la evaluación de costos, la planificación, el monitoreo de progreso del proyecto, y también como base para analizar reuso, portabilidad y seguridad. Es la base para establecer una linea de productos.

La vista de desarrollo de un sistema se representa en diagramas de módulos o subsistemas que muestran las relaciones exporta e importa. La arquitectura de desarrollo completa solo puede describirse completamente cuando todos los elementos del software han sido identificados. Sin embargo, es posible listar las reglas que rigen la arquitectura de desarrollo –partición, agrupamiento, visibilidad– antes de conocer todos los elementos.

1.13.4. La Vista física

La arquitectura física toma en cuenta primeramente los requisitos no funcionales del sistema tales como la disponibilidad, confiabilidad (tolerancia a fallas), performance (throughput), y escalabilidad. El software ejecuta sobre una red de computadores o nodos de procesamiento (o tan solo nodos). Los variados elementos identificados –redes, procesos, tareas y objetos– requieren ser mapeados sobre los variados nodos. Esperamos que diferentes configuraciones puedan usarse: algunas para desarrollo y pruebas, otras para emplazar el sistema en varios sitios para distintos usuarios. Por lo tanto, el mapeo del software en los nodos requiere ser altamente flexible y tener un impacto mínimo sobre el código fuente en si.

1.13.5. Escenarios

Los elementos de las cuatro vistas trabajan conjuntamente en forma natural mediante el uso de un conjunto pequeño de escenarios relevantes –instancias de casos de uso más generales– para los cuales describimos sus scripts correspondientes (secuencias de interacciones entre objetos y entre procesos) tal como lo describen (Rubin & Goldberg, 1992). Los escenarios son de alguna manera una abstracción de los requisitos más importantes. Su diseño se expresa mediante el uso de diagramas de escenarios y diagramas de interacción de objetos (Booch, 1994). Esta vista es redundante con las otras (y por lo tanto “+1”), pero sirve a dos propósitos principales:

- como una guía para descubrir elementos arquitectónicos durante el diseño de arquitectura tal como lo describiremos más adelante.
- como un rol de validación e ilustración después de completar el diseño de arquitectura, en el papel y como punto de partido de las pruebas de un prototipo de la arquitectura.

1.14. Comentarios Finales

Al terminar este primer capítulo, se puede concluir que la industria de la tecnología ha venido persiguen continuamente la integración de sus grandes fuentes de datos de forma en la que la información ya existente de los sistemas sea uno de los insumos principales para los nuevos

desarrollos. De igual forma se ven las aproximaciones más relevantes para conseguir esta integración y se identifica claramente como la integración semántica de los datos brinda una amplia solución a los problemas que las aproximaciones anteriores no logran resolver.

También se puede observar como iniciativas Open Source apuntan a la creación de plataformas que permitan a las empresas establecer un punto de interoperabilidad de datos entre aplicaciones.

Algo muy importante que se debe tener en cuenta es que para entender lo que la integración semántica de datos puede lograr se debe estudiar y comprender sus conceptos con la finalidad de evitar confusiones.

CAPÍTULO II: DEFINICIÓN DEL MARCO DE TRABAJO

2.1. Introducción

Dentro de este capítulo se explican las principales motivaciones que impulsaron el desarrollo del presente trabajo de fin de titulación. Abordando así la problemática que el presente trabajo pretende resolver con la solución planteada. De igual forma se identifican las principales causas que dan origen al problema identificado. Se presenta una solución puntual a cada una de estas causas y de esta forma poder dar solución al problema.

2.2. El problema

Grandes volúmenes de datos, diversidad, dispersión y ambigüedad, son calificativos que describen perfectamente a las fuentes de datos existentes en la Universidad Técnica Particular de Loja (UTPL). Actualmente se emplean bases de datos relacionales: Oracle, My SQL, My SQL Enterprise, Posgres y SQL Server; las mismas que utilizan diferentes estándares y protocolos de comunicación, ésta heterogeneidad de tecnologías convierte en una tarea bastante compleja la interoperabilidad entre estas fuentes de datos.

La duplicidad de registros existentes en cada una de las diferentes bases de datos Institucionales (Financiero, Académico, Recursos Humanos, Biblioteca, Editorial, etc.), es una clara evidencia que la ambigüedad y carencia de semántica en los datos institucionales crea el ambiente idóneo para la existencia de grandes problemas de integridad e interoperabilidad de los datos.

2.2.1. Causas principales

2.2.1.1. Carencia de modelos y diccionarios de datos actualizados de los sistemas: NSGA, UTE, Distributivo, EVA, SIAC, SIEC, FileMaker y Exalumnos.

La inexistencia de modelos y diccionarios de datos actualizados de los sistemas satélite de la Universidad Técnica Particular de Loja indican una madurez mínima o nula con respecto a un gobierno de datos. El no contar con modelos y diccionarios de datos actualizados impiden al departamento de tecnología desarrollar soluciones de software que conviertan los datos en insumos de alto valor para la toma de decisiones estratégicas en la organización.

Este problema principalmente se origina como producto de desarrollos de software aislados que promueven la formación de silos de información dentro de la organización.

2.2.1.2. Heterogeneidad de tecnologías de almacenamiento de datos.

La UTPL cuenta con una serie de sistemas satélite que son clave para lograr una correcta gestión y administración de sus recursos, los mismos que se encuentran desarrollados en diferentes lenguajes de programación y hacen uso de distintas tecnologías de base de datos.

La tabla 4 muestra un listado de los sistemas que posee la UTPL detallando cada uno de los lenguajes de programación y base de datos que estos utilizan:

Tabla 4: Diversidad de tecnologías en los sistemas de la UTPL

Sistema	Lenguaje de Desarrollo	Motor de Base de datos	versión del motor de base de datos	Tipo de Información que se almacena
NSGA	C#.Net	Oracle	11g	Estudiantes Docentes Horarios de Clase Becas Matrículas Distributivo
Financiero	C#.Net	Orcale	11g	cobros pagos
EVA	php	My SQL	5.6 Enterprise Edition	Tareas Foros Materias Evaluaciones
Biblioteca	php	My SQL	5.1 Standard Edition	Libros Revistas Tesis
SIAC	Java	My SQL	5.1 Standard Edition	Producción científica de los docentes de la UTPL
RRHH	Objective-C	File Maker	13	Información del personal que labora en la UTPL

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

2.2.1.3. Redundancia e Inconsistencia de Datos.

La universidad cuenta con un total de 35 aplicaciones, las mimas que poseen su propia base de datos donde se aloja información de varias entidades. Entidades a las cuales cada sistema tiene una forma particular de identificar de manera única dentro de su base de datos. Esto conlleva a la implementación de soluciones rudimentarias como: tablas de equivalencias para poder mapear una entidad del sistema A con la representación de la misma entidad en el sistema B.

En bases de datos, la redundancia hace referencia al almacenamiento de los mismos datos varias veces en diferentes lugares. La redundancia de datos puede provocar problemas como:

Incremento del trabajo: Como un mismo dato está almacenado en dos o más lugares, ésto hace que cuando se graben o actualicen los datos, deban hacerse en todos los lugares a la vez.

Desperdicio de espacio de almacenamiento: Ya que los mismos datos están almacenados en varios lugares distintos, ocupando así más bytes del medio de almacenamiento. Éste problema es más evidente en grandes empresas que manejan bases de datos con altos volúmenes de datos.

Puesto que las bases de datos que mantienen almacenada la información institucional son pobladas por diferentes tipos de sistemas existe la posibilidad de que, si no se controla detalladamente el almacenamiento, se pueda originar un duplicado de información. Esto aumenta los costos de almacenamiento y acceso a los datos, además puede originar la inconsistencia de los datos - es decir diversas copias de un mismo dato no concuerdan entre sí -, por ejemplo: Cuando se actualiza la dirección de un estudiante dentro de un sistema en otros permanecerá la anterior.

Notoriamente este es el caso de la UTPL, la misma que maneja varios sistemas para mantener una correcta administración y control de sus recursos. Esta inconsistencia producto de una redundancia en los datos almacenados por dichos sistemas, es uno de los problemas más críticos presentes en la UTPL.

Esta redundancia e inconsistencia se ve claramente en los registros de información que involucran al capital humano y estudiantes de la institución, en donde varios sistemas almacenan la misma información (Nombre, Apellido, Cédula, Dirección, Correo electrónico, etc.), lo cual obviamente genera una inconsistencia.

En la tabla 5 se muestran las fuentes en las que se encuentra registrada la información del capital humano y estudiantes de la UTPL.

Tabla 5: Redundancia de Datos en los Sistemas de la UTPL

Sistema	Docentes	Estructura Institucional	Componente Educativo	Estudiantes
Nómina	x	x		
RRHH	x	x		
SIEC	x	x	x	x
SIAC	x			
NSGA	x		x	x
Distributivo	x		x	x

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

La tabla 6 muestra un cuadro resumen de las principales causas al problema de interoperabilidad de datos que la UTPL presenta.

Tabla 6: Cuadro Resumen de las Principales Causas del Problema.

CAUSAS	CONSECUENCIAS	RELACIÓN CON LA FALTA DE INTEROPERABILIDAD
REDUNDANCIA E INCONSISTENCIA DE LOS,DATOS.	* Múltiples bases de datos que contiene la misma información. * Bases de datos desactualizadas	Al contar con múltiples fuentes que contiene la misma información no se puede definir una sola confiable, puesto que cada una cierta porción de la información actualizada.
CARENCIA DE MODELOS Y DICCIONARIOS DE DATOS ACTUALIZADOS.	* Creación de nuevas bases de datos con información redundante. * Formación de Silos de Información.	El carecer de modelos y diccionarios de datos actualizados promueven el desarrollo de software aislado que tiene a crear sus propias fuentes de datos y de esta forma se forman silos de información dentro de la Universidad.
DIVERSIDAD Y DISPERSIÓN EN LAS TECNOLOGÍAS DE ALMACENAMIENTO.	* Múltiples Formas de acceder a los datos. * Diferentes modelos y esquemas de datos. * Acceso a múltiples fuentes de información para validar datos. * Inexistencia de estándares de acceso a los datos.	La inexistencia de un estándar definido para el acceso a los datos de estas diferentes fuentes de información impide que los nuevos desarrollos tomen como insumo principal las fuentes de datos ya existentes.

Fuente: El Autor
Elaboración: El Autor

2.3. Solución propuesta

Frente al gran volumen de datos que se genera diariamente en la UTPL y los problemas que se identifican en la sección anterior, la interoperabilidad semántica de datos pretende ser una solución a dichos problemas atacando de forma puntual cada uno de éstos.

Dos problemas principales son identificados en la sección anterior, problemas a los cuales se presenta una solución detallada y funcional.

A continuación, se detalla la solución a tomar para cada uno de estos problemas.

2.3.1. Carencia de modelos y diccionarios de datos actualizados de los sistemas: NSGA, UTE, Distributivo, EVA, SIAC, FileMaker, Buxis y NSFA.

Definir políticas y protocolos de actualización de los modelos de datos existentes actualmente en la UTPL. Actividad que se la realizaría con cada uno de los custodios o responsables de estas fuentes de información, con la finalidad de establecer la frecuencia de actualización y definir un repositorio central de estos modelos y diccionarios. Esto se realizaría con la finalidad de evitar el desarrollo de soluciones de software aisladas que redunden información ya existente.

2.3.2. Diversidad en las tecnologías de almacenamiento de datos.

Al ser varias las aplicaciones con las que se trabaja en la UTPL, es común encontrar diversidad en las tecnologías utilizadas para realizar dichas aplicaciones. Las tecnologías que utilizan las aplicaciones para su correcto funcionamiento se pueden dividir en:

Lenguaje de programación. - Un lenguaje de programación es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

Motor de base de datos. - El motor de base de datos es el servicio principal para almacenar, procesar y proteger los datos. El Motor de base de datos proporciona acceso controlado y procesamiento de transacciones.

Protocolos de comunicación. - Un protocolo de comunicaciones es un sistema de reglas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellas para transmitir información por medio de cualquier tipo de variación de una magnitud física. Se trata de las reglas o el estándar que define la sintaxis, semántica y sincronización de la comunicación, así como también los posibles métodos de recuperación de errores. Los protocolos pueden ser implementados por hardware, por software, o por una combinación de ambos.

Formatos de Intercambio de datos. - Se usan para transferir documentos electrónicos o datos de negocios de un sistema computacional a otro. Este intercambio puede realizarse en distintos formatos: YAML, XML, ANSI ASC X12, TXT, JSON, etc.

Al ser notoria la utilización de todas las variantes de tecnologías mencionadas anteriormente en cada una de las aplicaciones que posee la UTPL, es necesario establecer protocolos, estándares y formatos de intercambio de comunicación e intercambio de datos entre las aplicaciones y levantar un solo repositorio con la información básica de contacto de las personas vinculadas a la UTPL.

El presente trabajo presenta como una solución viable a la diversidad de tecnologías la utilización de:

HTTP 2.0: Como el protocolo de comunicación entre aplicaciones. Al poseer un amplio catálogo de métodos y códigos de respuesta ofrece las características necesarias para lograr la interoperabilidad entre las aplicaciones que proveen y consumen datos.

Web Services: Como el medio de intercambio de información entre los sistemas.

RESTful: Como el estándar utilizado por los web services.

JSON, JSONLD, RDF, XML: Como los formatos de intercambio de datos utilizados por las aplicaciones y web services.

2.3.3. Redundancia e inconsistencia de datos.

Al Eliminar o controlar la redundancia de datos se reduce en gran medida el riesgo inconsistencias. Si un dato está almacenado una sola vez cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios y sistemas inmediatamente.
Como una forma de eliminar y controlar la redundancia de datos el presente trabajo se plantea crear un repositorio único de datos, al cual todas las aplicaciones y servicios puedan acceder y de esta forma poseer una sola instancia de esos datos.
Para eliminar la inconsistencia en los datos dentro de este repositorio único se pretende la utilización de tecnologías de web semántica para crear ontologías y vocabularios que permitan la representación de esta información de manera conceptual.

Al optar por tecnologías semánticas para la representación de la información también es necesario la selección de una plataforma que permita la trasformación de datos estructurados a grafos RDF, la carga y publicación de esos datos.

Al haber establecido las tecnologías con las que se debería trabajar para lograr una interoperabilidad de datos entre las aplicaciones, es preciso que la plataforma a seleccionar posea todas estas tecnologías integradas y es por eso que siguiendo la recomendación de la W3C, la solución que platea el presente trabajo opta por una alternativa Open Source denominada “Apache Marmotta” desarrollada en Java la misma que está construida sobre una arquitectura SOA y que posee una capa de servicios REST permite el acceso a todas las funcionalidades de la plataforma y de la misma forma el desarrollo de aplicaciones basadas en Linked Data. La información se almacenara en RDF dentro de un triplestore propio de Apache Marmotta.

Para convertir toda la data en RDF es necesario procesos automatizados que permitan la extracción, trasformación y carga de esos datos depurados a la plataforma de Linked Data. Estos procesos forman parte de un RESTful API que permita la interoperabilidad entre las

aplicaciones.

Este RESTful API seguirá una arquitectura de 3 niveles implementando MVC como patrón de diseño.

La figura 14 muestra una ilustración general de como se encuentra estructurada la arquitectura diseñada para la solución.

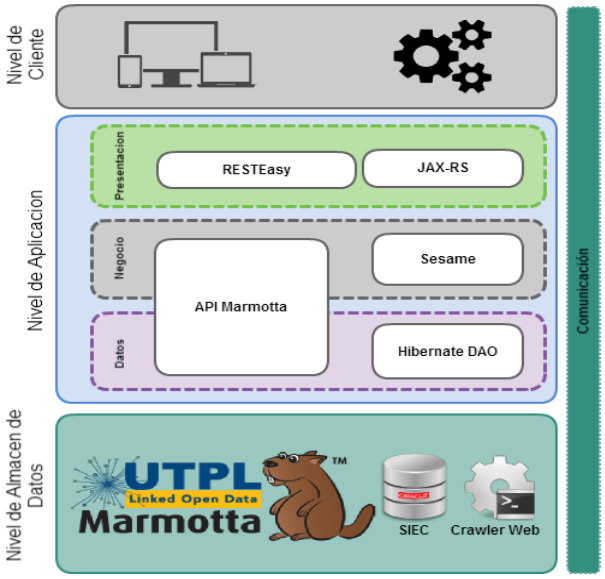


Figura 14: Arquitectura propuesta por la Solución
Fuente: El Autor
Elaboración: El Autor

La información a ser extraída, transformada y cargada en Apache Marmotta será la siguiente:

- Docentes.
- Estudiantes.
- Componentes Educativos.
- Estructura Institucional.

Conforme a las recomendaciones establecidas por la W3C se debe seguir un ciclo de publicación de datos enlazados o Linked data, razón por la cual esta solución opta por el ciclo de vida propuesto por (Piedra et al., 2014) el mismo que se encuentra dentro del domino de datos relacionados con educación; este ciclo propone un marco de trabajo para la publicación de datos enlazados, que inicia con una fase de visionamiento del modelo de negocio, identificación y selección de las fuentes de datos, modelamiento, generación de datos RDF, publicación, consumo y visualización.

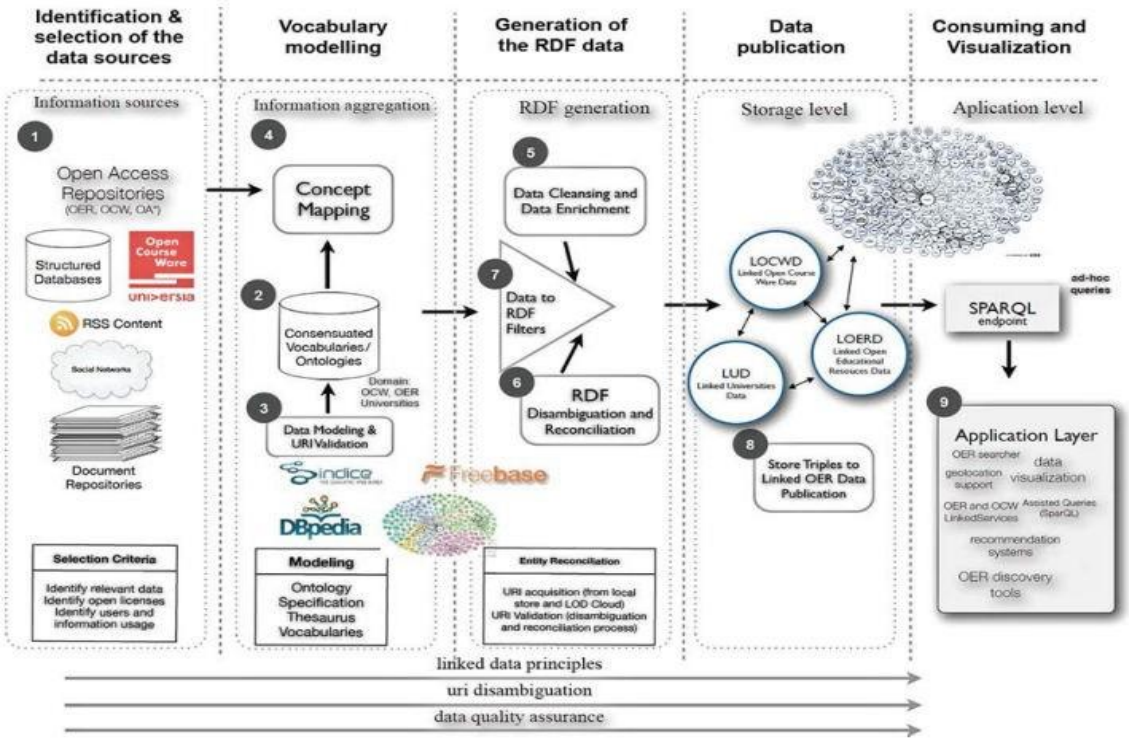


Figura 15: General framework for publishing open educational contents as linked data
Fuente: Recuperado de: https://goo.gl/WX9WS6
Elaboración: (Piedra et al., 2014)

La solución propuesta se desarrolla siguiendo las fases mencionadas a continuación. Siendo la primera el diseño de la arquitectura y las fases siguientes seguirán el orden establecido por el ciclo de publicación de datos enlazados que muestra la figura 15.

- Fase 1: Diseño y Documentación de la Arquitectura Propuesta.
- Fase 2: Identificación y Selección de las Fuentes de Datos.
- Fase 3: Modelamiento del Vocabulario.
- Fase 4: Generación de Datos RDF.
- Fase 6: Publicación de datos RDF.
- Fase 6: Consumo y visualización.

2.3.4. Preguntas que resuelven esta aproximación

La solución propuesta está dirigida a los desarrolladores que deseen hacer uso del RESTful API, los mismos que podrán obtener respuestas a preguntas como:

¿Cuáles son los datos a los que se puede tener acceso mediante el API?

La solución contempla su dominio respecto a la información básica de contacto de Docentes, Estudiantes y miembros del Gobierno General UTPL, como también a la información de los componentes Académicos.

¿Qué tipo de consultas se podrán realizar?

Existen tres tipos de consultas que se pueden realizar utilizando el RESTful API:

- **Listar por criterio** Este tipo de consulta permite obtener un listado al ingresar un criterio o filtro, ejemplo (listado de los docentes o estudiantes nacidos en Loja).
- **Buscar por número de cédula o código de componente** Esta consulta utilizando un identificador permite obtener la información de determinado Estudiante, Docente o Autoridad y de la misma forma toda la información de un Componente Académico.
- **Contabilizar por criterio** Este tipo de consulta devuelve un número entero el cual será el total de registros.

¿En que formatos se presentan los datos?

El desarrollador tendrá la opción de escoger en que formato desea que se presente el resultado estando como formatos disponibles JSON, RDF, XML, JSON-LD.

Las preguntas anteriores son las que el API REST sera capaz de responder de forma predeterminada. Para realizar consultas más complejas se utiliza el SPARQL endpoint de la plataforma para Linked Data Apache Marmotta.

2.4. Comentarios Finales

Al finalizar este capítulo se pude notar claramente que el problema de interoperabilidad de datos en las aplicaciones que posee la Universidad Técnica Particular (UTPL) de Loja son serio y que necesitan se les otorgue el grado de importancia que merecen, siendo así que al lograr una interoperabilidad semántica de datos en las aplicaciones representaría un paso agigantado al momento de recopilar datos para la toma de daciones estratégicas en la UTPL.

CAPÍTULO III: ANÁLISIS Y ESPECIFICACIÓN DE REQUERIMIENTOS

3.1. Introducción

Este tercer capítulo se enfoca en la perspectiva sobre la cual se desarrollará la solución y de igual forma se identifican los requerimientos funcionales y no funcionales que debe poseer las soluciones. De esta forma se describen cada uno de los requerimientos en utilizando una plantilla para la especificación de requerimientos.

Mediante el análisis de requerimientos buscamos capturar y describir detalladamente los requerimientos funcionales y no funcionales que deberán ser parte de la solución que plantea el presente trabajo de investigación.

La selección o clasificación de estos requerimientos entre funcionales y no funcionales se presenta a continuación:

3.2. Perspectiva de la solución

La solución establece el diseño e implementación de una arquitectura que garantice la integridad e interoperabilidad semántica de los datos académicos en la Universidad Técnica Particular de Loja (UTPL).

La figura 14 en el capitulo anterior muestra la arquitectura propuesta como parte de la solución define tres niveles (Cliente – Aplicación - Almacén de Datos) implementado en el nivel de la aplicación el patrón de diseño Modelo-Vista-Controlador (MVC) el mismo que expone una capa se servicios web tipo RESTful. Se adopta este modelo de arquitectura con la finalidad de poder conectar la capa de servicios de la solución con la capa de servicios que ofrece la plataforma para Likend Data seleccionada como lo es Apache Marmotta. Al conectar estas dos capas de servicios las tareas de interoperabilidad entre la plataforma y las aplicaciones se facilita significativamente.

El desarrollo de un API REST sobre la arquitectura propuesta permite la implementación de la misma en un entorno real. Este API se conecta a al triplestore de Apache Marmotta y permite realizar consultas complejas mediante una serie de servicios web RESTful. De igual forma el API se conecta a la base de datos SQL que posee datos actualizados de las cuatro entidades de datos seleccionadas las cuales se mencionan en el capítulo 3 de este trabajo. El resultado de todas las consultas que se realizan al API pueden ser serializadas en diferentes formatos de salida. Ofrecer diferentes formatos de salida se realiza con la finalidad de ofrecer una mayor flexibilidad a los desarrolladores que utilicen la solución como fuente de información para sus desarrollos.

3.3. Requerimientos funcionales

Se identifica a los requerimientos funcionales (RF) como los servicios que deben proporcionar la solución de la manera en que ésta debe reaccionar a entradas particulares y de como se debe comportar en situaciones especificas. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.

Las especificación de requerimientos se la realiza a través de una plantilla la misma que se detalla a continuación:

Código Especifica un código único asignado a un requerimiento en especifico compuesto de dos partes la primera una abreviatura (RF) Requerimiento Funcional y la segunda un dígito secuencial.

Nombre: Nombre asignado al Requerimiento.

Fecha: Indica la fecha en la que se elaboro.

Grado Necesidad: indica la importancia del requerimiento.

Descripción: Describe de forma breve y puntual el requerimiento en si.

Entradas: Son los insumos necesarios para la correcta ejecución del requerimiento.

Fuente: De donde provienen los insumos necesarios.

Salida: La información resultante luego de ejecutar el requerimiento.

Destino: Indica hacia donde se dirige la salida producto de la ejecución del requerimiento.

Restricciones: Señala reglas que condicionan la ejecución del requerimiento.

Proceso: Descripción detallada de los pasos que ejecuta el requerimiento.

Efecto Colateral: Es el efecto que tiene el requerimiento en el sistema luego de ejecutarse.

3.3.1. Búsquedas

La tabla 7 detalla el requerimiento funcional para la búsqueda de un estudiantes mediante su número de cédula. Se retorna un objeto JSON como resultado de dicha consulta.

Tabla 7: Requerimiento estudiantes

Especificación de Requerimientos Funcionales				
Código	**Nombre**		**Fecha**	**Grado Necesidad**
RF 01	Búsqueda de Estudiantes		10/05/2016	Esencial
Descripción	El Api debe permitir la búsqueda de estudiantes ingresando el número de cédula.			
Entradas	**Fuente**	**Salida**	**Destino**	**Restricciones**
Número de cédula	Base de Dato SIEC	Información básica de contacto del estudiante	Cliente RESTful	Solo debe retornar un registro
Proceso	El cliente que consulta este servicio deberá ingresar como parte de la URL del servicio web el número de cédula del estudiante al que desea acceder, en caso de existir algún registro de este estudiante se retornará la información básica de contacto en un JSON y de no existir registro alguno de este estudiante se enviará un mensaje notificando ésto dentro del JSON.			
Efecto Colateral	No Aplica			

Fuente: El Autor
Elaboración: El Autor

La tabla 8 detalla el requerimiento funcional para la búsqueda de un docente mediante su número de cédula. Se retorna un objeto JSON como resultado de dicha consulta.

Tabla 8: Requerimiento docentes

Especificación de Requerimientos Funcionales				
Código	**Nombre**		**Fecha**	**Grado Necesidad**
RF 02	Búsqueda de Docentes		10/05/2016	Esencial
Descripción	El Api debe permitir la búsqueda de docentes ingresando el número de cédula.			
Entradas	**Fuente**	**Salida**	**Destino**	**Restricciones**
Número de cédula	Base de Dato SIEC	Información básica de contacto del docente	Cliente RESTful	Solo debe retornar un registro único
Proceso	El cliente que consulta este servicio deberá ingresar como parte de la URL del servicio web el número de cédula del docente al que desea acceder, en caso de existir algún registro de este docente se retornará la información básica de contacto en un JSON y de no existir registro alguno de este docente se enviará un mensaje notificando ésto dentro del JSON.			
Efecto Colateral	No Aplica			

Fuente: El Autor
Elaboración: El Autor

La tabla 9 detalla el requerimiento funcional para la búsqueda de un miembro del gobierno UTPL mediante su número de cédula. Se retorna un objeto JSON como resultado de dicha consulta.

Tabla 9: Requerimiento Autoridades

Especificación de Requerimientos Funcionales				
Código	**Nombre**		**Fecha**	**Grado Necesidad**
RF 03	Búsqueda de Autoridades		10/05/2016	Esencial
Descripción	El Api debe permitir la búsqueda de Autoridades ingresando el Código del cargo.			
Entradas	**Fuente**	**Salida**	**Destino**	**Restricciones**
Número de Cédula	Base de Dato SIEC	Información básica de la autoridad	Cliente RESTful	Solo debe retornar un registro único
Proceso	El cliente que consulta este servicio deberá ingresar como parte de la URL del servicio web el código del cargo al que desea acceder, en caso de existir algún registro de esta autoridad se retornará la información básica de contacto en un JSON y de no existir registro alguno de esta autoridad se enviará un mensaje notificando ésto dentro del JSON.			
Efecto Colateral	No Aplica			

Fuente: El Autor
Elaboración: El Autor

La tabla 10 detalla el requerimiento funcional para la búsqueda de un componente académico mediante su código. Se retorna un objeto JSON como resultado de dicha consulta.

Tabla 10: Requerimiento componente académico

Especificación de Requerimientos Funcionales				
Código	**Nombre**		**Fecha**	**Grado Necesidad**
RF 04	Búsqueda de Componente Académico		10/05/2016	Esencial
Descripción	El Api debe permitir la búsqueda de un componente académico ingresando el código.			
Entradas	**Fuente**	**Salida**	**Destino**	**Restricciones**
Número de cédula	Base de Dato SIEC	Información básica de del componente Académico	Cliente RESTful	Solo debe retornar un registro único
Proceso	El cliente que consulta este servicio deberá ingresar como parte de la URL del servicio web el código del componente académico al que desea acceder, en caso de existir algún registro de este Componente Académico se retornará la información básica de contacto en un JSON y de no existir registro alguno de este Componente Académico se retornará un mensaje notificando ésto dentro del JSON.			
Efecto Colateral	No Aplica			

Fuente: El Autor
Elaboración: El Autor

3.4. Requerimientos no funcionales

3.4.1. Eficiencia

- Toda funcionalidad del Api y transacción de negocio debe responder al usuario en menos de 5 segundos.
- El sistema debe ser capaz de operar adecuadamente con hasta 1000 transacciones.
- Los datos modificados en la base de datos deben ser actualizados para todos los usuarios que acceden en menos de 2 segundos.

3.4.2. Usabilidad

- El Api debe contar con manuales de usuario estructurados adecuadamente.
- El Api debe proporcionar mensajes de error que sean informativos y orientados al usuario final.
- El Api debe proporcionar URLs amigables.
- El sistema estará dirigido a desarrolladores.

3.4.3. Seguridad

- El acceso a los datos será solamente de lectura para los desarrolladores.
- Las operaciones de CRUD quedan estrictamente dirigidas a la Unidad de Gestión de Datos Académicos (UGDA).

3.4.4. Rendimiento

- El Api debe soportar el manejo de gran cantidad de información durante el proceso.
- El Api debe correr sobre un servidor Tomcat de versión superior a 7.
- El sistema deberá responder en el mínimo de tiempo posible ante las solicitudes de información por parte de otros sistemas y en el procesamiento de la información. La eficiencia de la aplicación estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo de 3 capas y la velocidad de las consultas a la base de datos.

3.5. Comentarios Finales

Al finalizar este capítulo se pude tener una idea clara de cuáles serán las funcionalidades que poseerá la solución y se puede apreciar una visión anticipada de como pretende solucionar el problema de interoperabilidad.

CAPÍTULO IV: DISEÑO DE LA SOLUCIÓN

4.1. Introducción

Dentro de este capítulo se presenta el diseño de la solución en base a los requerimientos que fueron identificados en el capítulo anterior. Para diseñar esta solución es necesario seguir el modelo planteado por (Kruchten, 1995). Al seguir este modelo la solución se pude observar desde 5 vistas diferentes y de esta forma tener una idea más clara de los componentes que conformaran la arquitectura propuesta como solución.

Este capítulo posee una serie de diagramas propios del modelo de Kruthen los cuales cuentan con una serie de descripciones que ayudaran a comprender de mejor manera su contenido.

4.2. Presentación de la arquitectura propuesta.

Diseñar una arquitectura robusta que garantice la interoperabilidad semántica e integridad de datos entre aplicaciones es el objetivo principal del presente trabajo de fin de titulación, razón por la cual se utiliza el modelo de vistas 4+1 propuesto por (Kruchten, 1995) que encaja con el estándar “IEEE 1471-2000” (Recommended Practice for Architecture Description of Software-Intensive Systems (Hilliard, 2000)) que se utiliza para describir la arquitectura de un sistema software intensivo basado en el uso de múltiples puntos de vista.

La arquitectura que se presenta a continuación en una serie de vistas a continuación es una arquitectura de 3 niveles implementando el patrón de diseño Modelo-Vista-Controlador (MVC) en el nivel de la aplicación.

4.3. Vista de casos de uso

La figura 16 muestra la vista 4+1 o también conocida como vista de casos de uso, figura en la cual se puede apreciar cada uno de los casos de uso que posee la solución.

La representación de casos de uso se la realiza utilizando diagramas UML en el cual se puede definir de forma más técnica la interacción que tiene el usuario con la solución y como cada una de las funciones que posee la solución se relacionan entre sí.

En particular dentro de la solución planteada se definen tres casos de uso padres (Buscar, Ejecutar SPAQRL y CRUD) de los cuales se extiendes o derivan 10 casos de uso hijos los cuales heredan características del caso de uso parare para ejecutar una tarea en particular.

El API expuesto a los sistemas solamente tiene accesos de lectura sobre los datos siendo la UGDA la única con la posibilidad de ejecutar creaciones, eliminaciones, actualizaciones de los datos sobre las entidades.

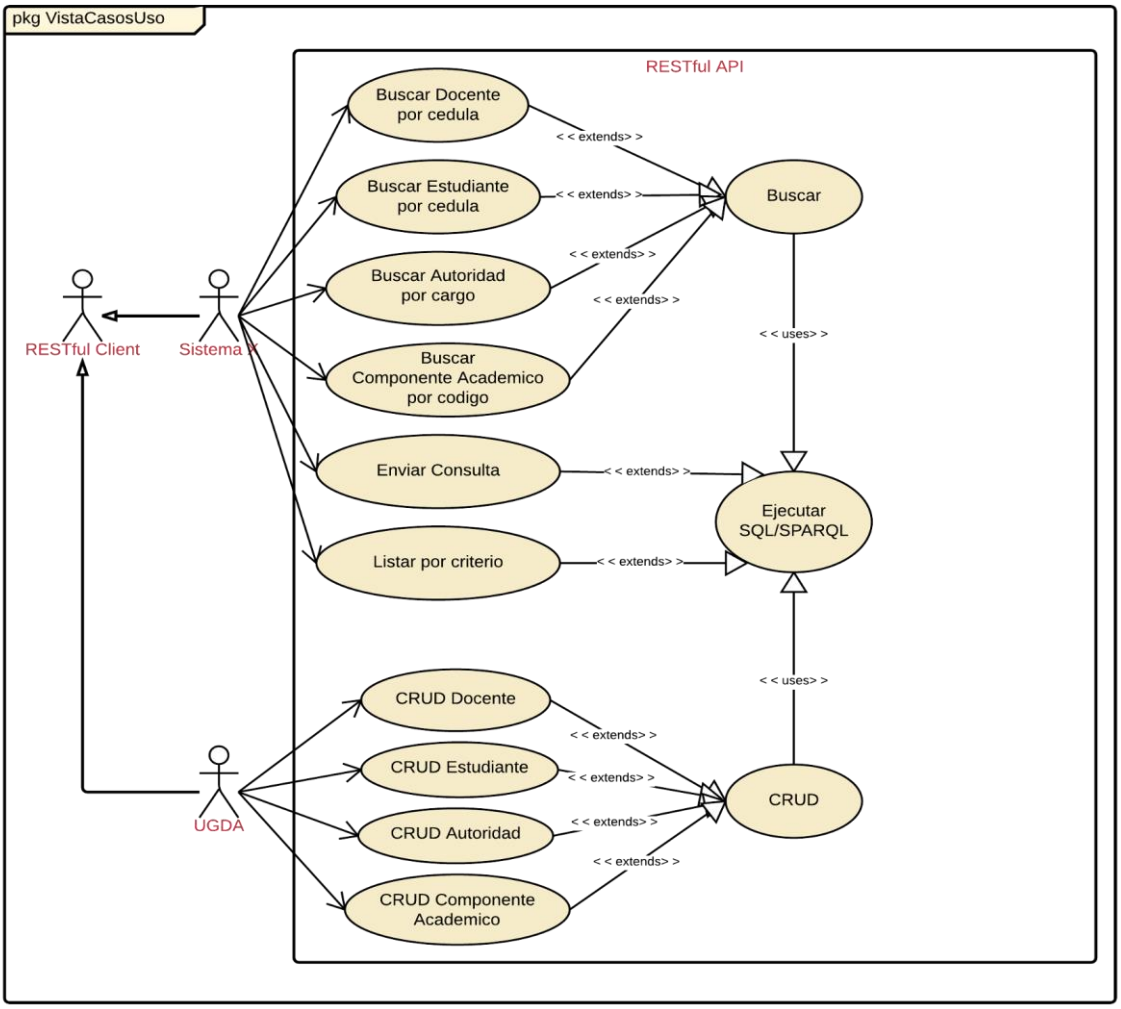


Figura 16: Casos De Uso
Fuente: El Autor
Elaboración: El Autor

4.3.1. Especificación de casos de uso

En la figura 16 se muestra los casos de uso enfocados en cada una de las entidades previamente seleccionadas. Para realizar las descripción de cada uno de estos casos de uso se utiliza una pantilla de especificación de casos de uso.

Las especificación de casos de uso se la realiza a través de una plantilla que contiene 9 atributos que detallan con claridad a los casos de uso y la misma que se detalla a continuación para mejor compresión.

Nombre: Nombre asignado al Requerimiento.

Codigo: Indica la fecha en la que se elaboró.

Descripción: Describe de forma detallada la información del caso de uso.

Actor(es): Son los entes que intervine dentro del caso de uso.

Precondición: Menciona las condiciones que se debe existir antes del caso de uso.

Poscondición: Indica las condiciones luego de haberse ejecutado el caso de uso.

Flujo Normal: Detalla el orden normal de los eventos de un caso de uso.

Excepciones: Indica que podría salir mal durante la ejecución del caso de uso.

Anotaciones: Observaciones sobre el caso de uso.

La tabla 11 muestra la especificación del caso de uso Buscar Docente Por Cédula en base a la platilla descrita anteriormente. Dentro de esta descripción se puede observar el flujo que se ejecuta al realizar la búsqueda de un docente por medio de API.

Tabla 11: Especificación búsqueda docente por cédula

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	Buscar Docente por Cédula
CÓDIGO	1
DESCRIPCIÓN	Búsqueda de un Docente en particular utilizando su número de cédula como identificador.
ACTOR(ES)	RESTful Client
PRECONDICIÓN	Se debe pasar el número de cédula del docente como parámetro de búsqueda.
POSTCONDICIÓN	El número de cédula enviado debe ser válido para obtener algún registro de retorno.
FLUJO NORMAL	1. Establecer el número de cédula como parámetro de búsqueda en el URL del Servicio Web de Docentes. 2. Seleccionar el formato en el que se desea presentar los datos. 3. Presentar los datos de la búsqueda en el formato seleccionado. 4. Procesar los datos obtenidos.
EXCEPCIONES	Si el número de cédula enviado como parámetro no es válido se informará de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 12 muestra la especificación del caso de uso Buscar Estudiante Por Cédula. Dentro de esta descripción o especificación se puede observar el flujo que se ejecuta al realizar la búsqueda de un estudiante por medio de API, siendo un cliente REST el actor sobre este caso de uso.

Tabla 12: Especificación búsqueda estudiante por cédula.

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	Buscar Estudiante por Cédula
CÓDIGO	2
DESCRIPCIÓN	Búsqueda de un Estudiante en particular utilizando su número de cédula como identificador.
ACTOR(ES)	RESTful Client
PRECONDICIÓN	Se debe pasar el número de cédula del Estudiante como parámetro de búsqueda.
POSTCONDICIÓN	El número de cédula enviado debe ser válido para obtener algún registro de retorno.
FLUJO NORMAL	1. Establecer el número de cédula como parámetro de búsqueda en el URL del Servicio Web de Estudiantes. 2. Seleccionar el formato en el que se desea presentar los datos. 3. Presentar los datos de la búsqueda en el formato seleccionado. 4. Procesar los datos obtenidos.
EXCEPCIONES	Si el número de cédula enviado como parámetro no es válido se informará de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 13 muestra la especificación del caso de uso Buscar Autoridad Por Cargo. Dentro de esta descripción o especificación se puede observar el flujo que se ejecuta al realizar la búsqueda de un estudiante por medio de API, siendo un cliente REST el actor sobre este caso de uso.

Se debe definir como parámetro en la URL el cargo de la Autoridad que se desee buscar.

Tabla 13: Especificación búsqueda autoridad por cargo.

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	Buscar Autoridad por Cargo
CÓDIGO	3
DESCRIPCIÓN	Búsqueda de una Autoridad que integre el Gobierno General UTPL utilizando su cargo como filtro de búsqueda.
ACTOR(ES)	RESTful Client
PRECONDICIÓN	Se debe pasar el cargo de la Autoridad como parámetro de búsqueda.
POSTCONDICIÓN	El cargo enviado debe estar registrado como parte del Gobierno General UTPL para obtener algún registro de retorno.
FLUJO NORMAL	1. Establecer el cargo de la autoridad como filtro de búsqueda en el URL del Servicio Web de Gobierno UTPL. 2. Seleccionar el formato en el que se desea presentar los datos. 3. Presentar los datos de la búsqueda en el formato seleccionado. 4. Procesar los datos obtenidos.
EXCEPCIONES	Si el cargo enviado como parámetro no es válido se informará de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 14 muestra la especificación del caso de uso Buscar Componente Académico por Código. Dentro de esta descripción o especificación se puede observar el flujo que se ejecuta al realizar la búsqueda de un componente académico por medio de API, siendo un cliente REST el actor sobre este caso de uso.

Tabla 14: Especificación búsqueda componente académico por código.

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	Buscar Componente Académico por Código.
CÓDIGO	4
DESCRIPCIÓN	Búsqueda de un Componente Académico utilizando código de identificación como filtro de búsqueda.
ACTOR(ES)	RESTful Client
PRECONDICIÓN	Se debe pasar el código del componente como parámetro de búsqueda.
POSTCONDICIÓN	El código debe estar asignado a un componente académico para obtener algún registro de retorno.
FLUJO NORMAL	1. Establecer el código del componente como filtro de búsqueda en el URL del Servicio Web de Gobierno UTPL. 2. Seleccionar el formato en el que se desea presentar los datos. 3. Presentar los datos de la búsqueda en el formato seleccionado. 4. Procesar los datos obtenidos.
EXCEPCIONES	Si el código enviado como parámetro no es válido se informará de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 15 muestra la especificación del caso de uso Enviar Consulta. Dentro de esta descripción o especificación se puede observar el flujo que se ejecuta al enviar una consulta por medio de API.

Tabla 15: Especificación Enviar Consulta.

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	Enviar Consultas
CÓDIGO	5
DESCRIPCIÓN	El desarrollador que utilice el API podría enviar consultas personalizadas para obtener los datos conforme a sus necesidades y en el orden que él los desee.
ACTOR(ES)	RESTful Client
PRECONDICIÓN	Se debe enviar una consulta SQL o SPARQL personalizada
POSTCONDICIÓN	La consulta debe seguir los parámetros establecidos por el API
FLUJO NORMAL	1.Construir consulta personalizada 2. Seleccionar el tipo de consulta 3. Seleccionar el formato en el que se desea presentar los datos. 3. Presentar los datos de la búsqueda en el formato seleccionado. 4. Procesar los datos obtenidos.
EXCEPCIONES	Si la consulta enviada no es valida se informará de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 16 muestra la especificación del caso de uso CRUD Docente. Dentro de esta especificación se observa el flujo que la solución debe seguir al iniciarse esta tarea.

Tabla 16: Especificación CRUD Docente.

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	CRUD Docente
CÓDIGO	6
DESCRIPCIÓN	La UDGA administra el API y tiene acceso a tareas de Creación, Lectura, Actualización y Borrado de registros sobre Docentes con los que trabaja el API.
ACTOR(ES)	RESTful API
PRECONDICIÓN	Se debe enviar el número de cédula y tipo de tarea para poder realizar cualquiera de las tareas de CRUD en un registro.
POSTCONDICIÓN	El número de cédula y la tarea deben ser válidas para que la tarea sea realizada con éxito.
FLUJO NORMAL	1. Enviar número de cédula y tipo de tarea como parámetros dentro de la URL del Web service CRUD Docente. 2. Presentar mensaje de confirmación de tarea.
EXCEPCIONES	Si la tarea no se pudo realizar se informará la razón de este fallo dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 17 muestra la especificación del caso de uso CRUD Estudiante. Docente. Dentro de esta especificación se observa el flujo que la solución debe seguir al iniciarse esta tarea.

Tabla 17: Especificación CRUD Estudante.

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	CRUD Estudiante
CÓDIGO	7
DESCRIPCIÓN	La UDGA administra el API y tiene acceso a tareas de Creación, Lectura, Actualización y Borrado de registros sobre Estudiantes con los que trabaja el API.
ACTOR(ES)	RESTful API
PRECONDICIÓN	Se debe enviar el número de cédula y tipo de tarea para poder realizar cualquiera de las tareas de CRUD sobre un registro.
POSTCONDICIÓN	El número de cédula enviado y la tarea deben ser válidas para que la tarea sea realizada con éxito.
FLUJO NORMAL	1.Enviar número de cédula y tipo de tarea como parámetros dentro de la URL del Web service CRUD Estudiante. 2. Presentar mensaje de confirmación de tarea.
EXCEPCIONES	Si la tarea no se pudo realizar se informará la razón de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 18 muestra la especificación del caso de uso CRUD Autoridad. Docente. Dentro de esta especificación se observa el flujo que la solución debe seguir al iniciarse esta tarea.

Tabla 18: Especificacion CRUD Autoridad

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	CRUD Autoridad
CÓDIGO	8
DESCRIPCIÓN	La UDGA administra el API y tiene acceso a tareas de Creación, Lectura, Actualización y Borrado de registros sobre las autoridades que integran el Gobierno General UTPL con los que trabaja el API.
ACTOR(ES)	RESTful API
PRECONDICIÓN	Se debe enviar el número de cédula y tipo de tarea para poder realizar cualquiera de las tareas de CRUD sobre un registro.
POSTCONDICIÓN	El número de cédula enviado y la tarea deben ser válidas para que la tarea sea realizada con éxito.
FLUJO NORMAL	1.Enviar número de cédula y tipo de tarea como parámetros dentro de la URL del Web service CRUD Gobierno UTPL. 2. Presentar mensaje de confirmación de tarea.
EXCEPCIONES	Si la tarea no se pudo realizar se informará la razón de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

La tabla 19 muestra la especificación del caso de uso CRUD Componente Académico. Dentro de esta especificación se observa el flujo que la solución debe seguir al iniciarse esta tarea.

Tabla 19: Especificación CRUD Componente Académico

ESPECIFICACIÓN DE CASO DE USO	
NOMBRE	CRUD Componente Académico
CÓDIGO	9
DESCRIPCIÓN	La UDGA administra el API y tiene acceso a tareas de Creación, Lectura, Actualización y Borrado de registros sobre Componentes Académicos con los que trabaja el API.
ACTOR(ES)	RESTful API
PRECONDICIÓN	Se debe enviar el código del componente y tipo de tarea para poder realizar cualquiera de las tareas de CRUD sobre un registro.
POSTCONDICIÓN	El número de cédula enviado y la tarea deben ser válidas para que la tarea sea realizada con éxito.
FLUJO NORMAL	1.Enviar código del componente y tipo de tarea como parámetros dentro de la URL del Web service CRUD Componente Académico. 2. Presentar mensaje de confirmación de tarea.
EXCEPCIONES	Si la tarea no se pudo realizar se informará la razón de este incidente dentro de una estructura JSON.
ANOTACIONES	Ninguna.

Fuente: El Autor
Elaboración: El Autor

4.4. Vista Lógica

Está enfocada en describir la estructura y funcionalidad del sistema. Los diagramas UML que se utilizan para representar esta vista son los Diagrama de Clase, Diagrama de Comunicación y Diagrama de Secuencia.

La figura 17 muestra la vista lógica de la arquitectura propuesta en la solución.

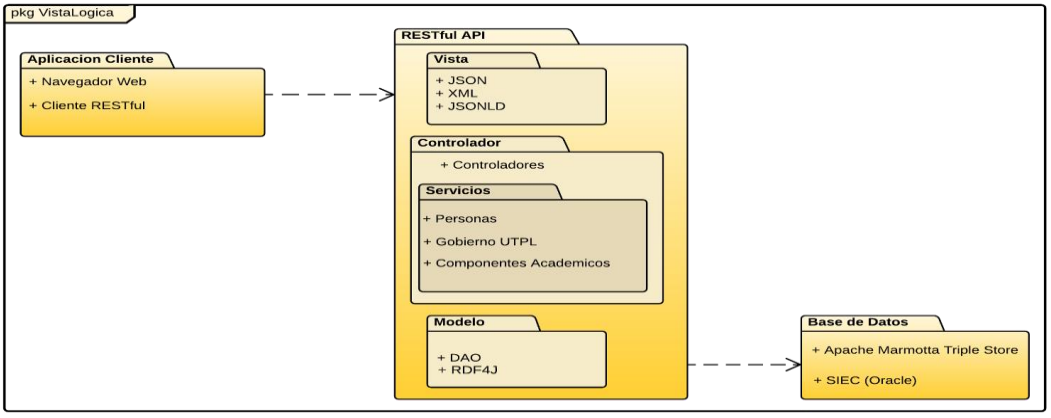


Figura 17: Vista Lógica
Fuente: El Autor
Elaboración: El Autor

4.5. Vista de procesos

Trata los aspectos dinámicos del sistema, explica los procesos de sistema y cómo se comunican. se enfoca en el comportamiento del sistema en tiempo de ejecución. La vista considera aspectos de concurrencia, distribución, rendimiento, escalabilidad, etc. En UML se utiliza el Diagrama de Actividad para representar esta vista

La figura 18 muestra el diagrama perteneciente a la vista lógica de la arquitectura propuesta.

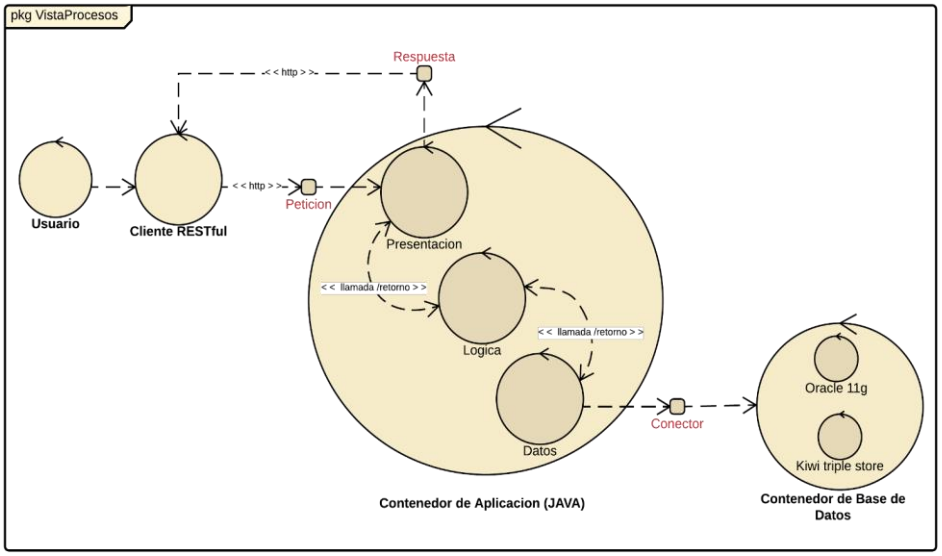


Figura 18: Vista de Procesos
Fuente: El Autor
Elaboración: El Autor

4.6. Vista de Desarrollo

Ilustra el sistema de la perspectiva del programador y está enfocado en la administración de los artefactos de software. Esta vista también se conoce como vista de implementación. Utiliza el Diagrama de Componentes UML para describir los componentes de sistema. Otro diagrama UML que se utiliza en la vista de desarrollo es el Diagrama de Paquetes

La figura 19 muestra el diagrama perteneciente a la vista de Desarrollo de la arquitectura propuesta.

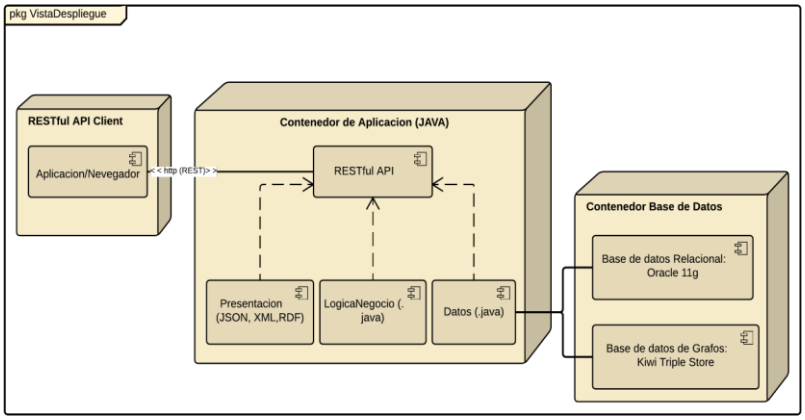


Figura 19: Vista de Desarrollo
Fuente: El Autor
Elaboración: El Autor

4.7. Vista de Despliegue

Describe el sistema desde el punto de vista de un ingeniero de sistemas. Está relacionada con la topología de componentes de software en la capa física, así como las conexiones físicas entre estos componentes.

La figura 20 muestra el diagrama perteneciente a la vista de Despliegue de la arquitectura propuesta.

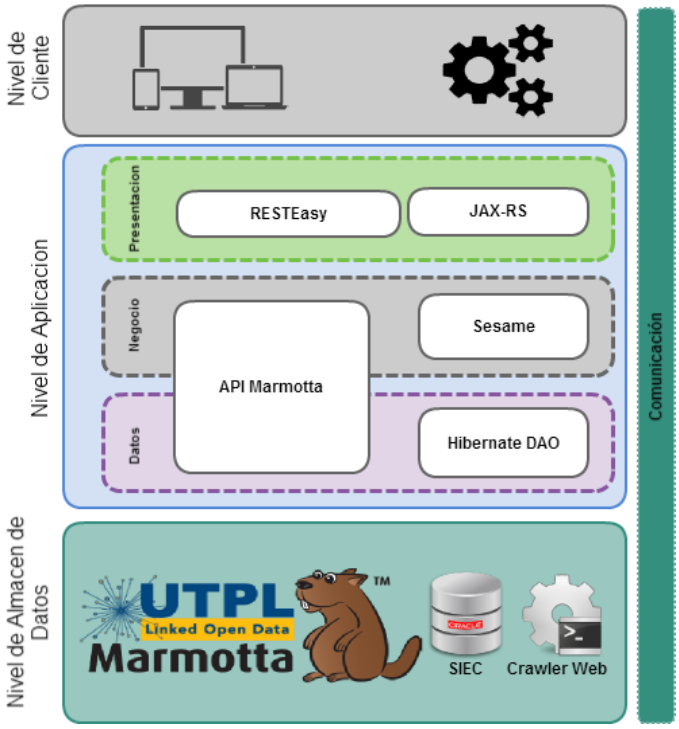


Figura 20: Arquitectura propuesta para el RESTful API
Fuente: El Autor
Elaboración: El Autor

4.8. Comentarios finales.

Al finalizar este capítulo claramente se pude identificar como cada una de las vistas encaja con otra. La vista de casos de uso se relaciona directamente con el capítulo 3 en donde se definen los requerimientos funcionales que debe poseer la solución.

Al utilizar este modelo para diseñar la arquitectura se identifica la finalidad de cada componente que la integra y de esa forma facilita la comprensión para el desarrollador que sera quien la implemente.

CAPÍTULO V: CASO DE APLICACIÓN

5.1. Introducción

La solución que se plantea y se Diseña en capítulos anteriores es una solución genérica que puede ser aplicada en varios dominios o escenarios que la UTPL lo desee. Uno de esos escenarios es la creación de catálogos de información que pueden ser consultados por diferentes sistemas de manera concurrente y de esta forma evitar la creación de más instancias de una misma entidad en todos los sistemas.

Para demostrar que la solución propuesta en este trabajo de investigación sea favorable, se establece un caso de caso de aplicación de la solución, el cual define como dominio las siguientes entidades: Estudiantes, Docentes, Autoridades y Componente Académico, las mismas que serán publicadas utilizando el API y*Apache Marmotta* como LDP.

5.2. Situación actual

En la actualidad, la Universidad Técnica Particular de Loja cuenta con una serie de sistemas que le permiten realizar un manejo más organizado y meticuloso de las actividades que se efectúan en la institución.

Los sistemas de apoyo que posee la UTPL manejan sus propias bases de datos sobre las cuales crean, acceden, actualizan y eliminan registros de una o varias entidades, lo que implica la creación y existencia de diferentes variantes de una misma entidad en cada uno de los sistemas donde se las necesitan, como se expone en el capítulo 2 de este trabajo de investigación, este comportamiento de los sistemas genera una serie de problemas que inciden en la calidad de los datos que la institución posee.

Actualmente encontramos situaciones como: Ambigüedades, redundancia y dispersión de la información, situaciones que se detallan a continuación

- Campos únicos para nombres y apellidos, donde en un solo atributo se encuentran los dos nombres y dos apellidos de una persona, haciendo más compleja las tareas de búsqueda de registros.
- Dentro de la base de datos de exalumnos, específicamente en la relación de experiencia laboral se puede encontrar un mismo rol o cargo expresado de diferentes maneras. Ejemplo: Docente se encuentra expresado de la siguiente manera (Profesor, Profesora, Teacher, Profesor(a) o docente). En donde se hace imposible obtener un listado completo de los exalumnos que se dedican o se han dedicado a la docencia, esto como producto de la ambigüedad del lenguaje natural utilizado en los datos.
- Los sistemas: EVA, SIAC, NSGA, RRHH, Exalumnos, Distributivo y UTE cuentan con instancias propias de las entidades Docente, Estudiante, Componente Académicos y Au-

toridad UTPL. Estas instancias se administran de forma independiente por cada sistema siendo así que cuando se efectúa una actualización en cualquiera de éstos sistemas el cambio no se ve reflejado en los demás.

- Cada sistema tiene una forma particular de identificar de manera única a estas entidades creando identificadores distintos al número de cédula, RUC o pasaporte en el caso de las personas y código establecido por la Senescyt en el caso de los programas y componentes académicos.
- Estas fuentes de datos dispersas e incomunicadas dan lugar a la formación de silos de información.

A continuación, se detallan todas y cada una de las fases que el caso de aplicación siguió para conseguir datos sin ambigüedades y con una representación semántica.

5.3. Identificación y selección de las fuentes de datos.

La solución planteada identifica las cuatro entidades con las que se trabajará, estas entidades se encuentran alojadas en varios sistemas que maneja la UTPL. La tabla 20 detalla cada uno de los sistemas que alojan información correspondiente a estas entidades, de la misma forma muestra el motor de base de datos, la versión y los datos que estos sistemas almacenan y la frecuencia con la que son actualizados.

Tabla 20: Fuentes de datos identificadas

Sistema	Base de Datos	Versión de Base Datos	Actualización	Información que Maneja
NSGA	Oracle	11g	Semestral	Estudiantes Docentes Horarios de Clase Becas Matrículas
NÓMINA	Oracle	11g	Semestral	Personal UTPL
EVA	My SQL	5.6 Enterprise Edition	Semestral	Estudiantes Docentes Tareas Foros Componentes Evaluaciones
Biblioteca	My SQL	5.1 Standard Edition	Semestral	Libros Revistas Tesis
SIAC	My SQL	5.1 Standard Edition	Diaria	Producción científica de la UTPL
RRHH	File Maker	13	Semestral	Personal UTPL

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

Al contar con las fuentes de datos previamente identificadas es necesario conocer en que sistema se encuentra almacenada la información de cada una de las cuatro entidades que para esta solución se han seleccionado.

la tabla 21 muestra de forma particular cada entidad y dentro de que sistema se encuentra alojada su información.

Tabla 21: Entidades y Fuentes

Entidades y Fuentes				
Fuente (Sistema)	Docente	Gobierno UTPL	Componente Educativo	Estudiante
Nómina	x	x		
RRHH	x	x		X
SIAC	x			
SIEC	x		x	x
Distributivo	x		x	x
NSGA	x		x	x

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

5.3.1. Selección y descripción de los atributos de las entidades seleccionadas

Seleccionar los atributos que contengan la información básica de contacto de los Estudiantes, Docentes, Gobierno UTPL y de la misma forma la información básica de los componentes educativos ofertados por la UTPL. La descripción de estos atributos previamente seleccionados de cada entidad permitirá asignar una breve descripción e identificar que tipo de dato corresponde a cada atributo.

5.3.1.1. Estudiante

Se denomina como estudiantes aquellas personas matriculadas dentro de cualquiera de las modalidades de estudio que la UTPL posee y que aún no se han graduado.

La tabla 22 describe los atributos que se considera como información básica de contacto de los estudiantes.

Tabla 22: Descripción de los Atributos de la Entidad Estudiante

ESTUDIANTE			
Atributo	**Descripción**	**Privacidad**	**Tipo de Dato**
Nombres	Primer y Segundo Nombre	Público	Cadena
Apellidos	Apellido Paterno y Materno	Público	Cadena
Cédula	Númerode Cédula	Público	Cadena
Correo Electrónico Personal	Correo Electrónico Personal del Estudiante	Privado	Cadena
Correo Electrónico Institucional	Correo Electrónico Institucional del Estudiante	Público	Cadena
Teléfono	Número de Extensión UTPL	Público	Cadena
Celular	Número de Teléfono Celular	Privado	Cadena
Género	Género del Estudiante	Público	Cadena
Nacionalidad	Nacionalidad del Estudiante	Público	Cadena
Fecha de Nacimiento	Fecha de Nacimiento del Estudiante	Público	Fecha
Fecha de Inicio de Estudios	Fecha en la que inició los estudios en la UTPL	Privado	Fecha
Titulación	Titulación en la que se encuentra matriculado	Privado	Cadena
Estado	Estado del Estudiante (Activo - Inactivo)	Privado	Cadena
Modalidad	Modalidad en la que se encuentra matriculado	Privado	Cadena

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

5.3.1.2. Docente

Se denomina como Docente aquellas personas que poseen un contrato con la UTPL ya sea este de tipo “eventual” o “indefinido” y se encuentre como tutor principal o auxiliar de un Componente Académico.

La tabla 23 describe los atributos que se considera como información básica de contacto del Docente.

Tabla 23: Descripción de los Atributos de la Entidad Docente

DOCENTE			
Atributo	**Descripción**	**Privacidad**	**Tipo de Dato**
Nombres	Primer y Segundo Nombre	Público	Cadena
Apellidos	Apellido Paterno y Materno	Público	Cadena
Cédula	Número de Cédula	Público	Cadena
Correo Electrónico Personal	Correo Electrónico Personal del Docente	Privado	Cadena
Correo Electrónico Institucional	Correo Electrónico Institucional del Docente	Público	Cadena
Teléfono	Número de Extensión UTPL	Público	Cadena
Celular	Número de Teléfono Celular	Privado	Cadena
Género	Género del Docente	Público	Cadena
Nacionalidad	Nacionalidad del Docente	Público	Cadena
Dirección	Dirección del Domicilio del Docente	Privado	Cadena
Fecha de Nacimiento	Fecha de Nacimiento del Docente	Público	Fecha
Fecha de Inicio de Estudios	Fecha en la que Inició a Trabajar en la UTPL	Privado	Fecha
Titulación	Titulación a la que Pertenece	Privado	Cadena
Estado	Estado del Docente (Activo - Inactivo)	Privado	Cadena
Tipo	Tipo de Contrato que posee	Privado	Cadena

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

5.3.1.3. Componente académico

Se denomina como Componente Académico aquellas materias ofertadas dentro de la malla curricular de todas las modalidades de estudio en la UTPL y que para esta solución solo se tomarán en cuenta las de la modalidad presencial de pregrado.

La tabla 24 describe los atributos que se considera como información básica y de interés de cada Componente Académico.

Tabla 24: Descripción de los Atributos de la Entidad Componente Académico

COMPONENTE ACADÉMICO			
Atributo	**Descripción**	**Privacidad**	**Tipo de Dato**
Código del Componente	Código Unificado UTPL del Componente	Privado	Cadena
Nombre del Componente	Nombre Completo del Componente	Público	Cadena
Descripción del Componente	Breve Descripción del Componente Académico	Público	Cadena
Estado del Componente	Vigencia del Componente	Privado	Cadena
Número de Créditos	Número de Créditos del Componente Académico	Público	Cadena
Periodo Ofertado	Último Periodo en que se Ofertó el Componente	Privado	Cadena

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

5.3.1.4. Gobierno General UTPL

Se denomina como Gobierno General UTPL a un grupo limitado de personas integrado de forma general por la autoridad de cogobierno; la primera autoridad ejecutiva; los vicerrectores; las autoridades ejecutivas, académicas, administrativas, de gestión y de apoyo, y los órganos colegiados que no constituyen gobierno. La figura 21 muestra la ilustración jerárquica de la UTPL.

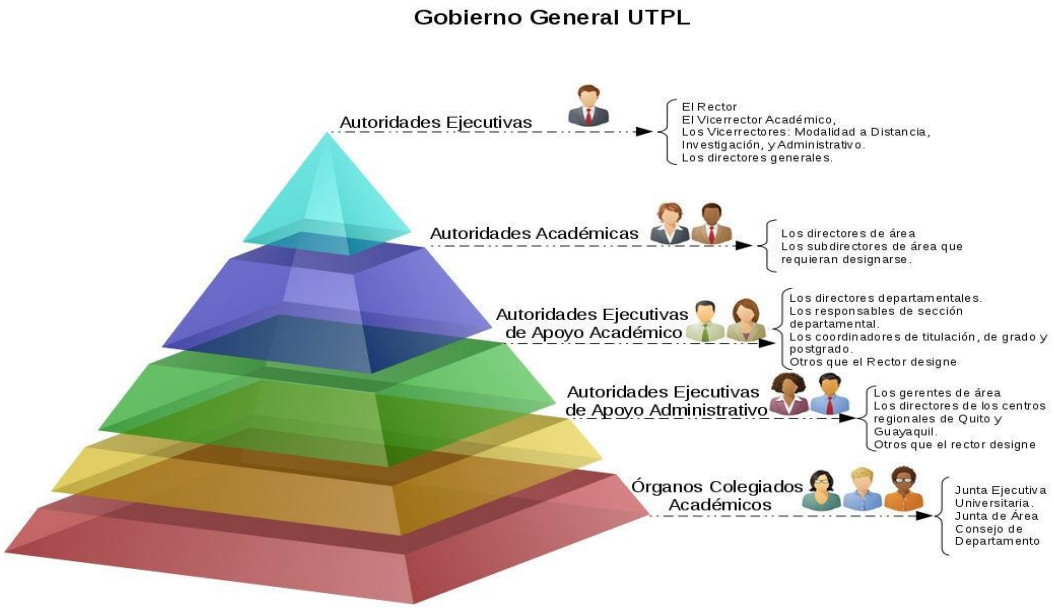


Figura 21: Estructura Jerárquica de la UTPL
Fuente: Portal UTPL
Elaboración: El Autor

La tabla 25 describe los atributos que se considera como información básica de contacto de los integrantes del Gobierno General UTPL.

Tabla 25: Descripción de los Atributos del Gobierno General UTPL

AUTORIDAD DEL GOBIERNO GENERAL UTPL			
Atributo	**Descripción**	**Privacidad**	**Tipo de Dato**
Nombres	Primer y Segundo Nombre	Público	Cadena
Apellidos	Apellido Paterno y Materno	Público	Cadena
Cédula	Número de Cédula	Público	Cadena
Correo Electrónico Personal	Correo Electrónico Personal de la Autoridad	Privado	Cadena
Correo Electrónico Institucional	Correo Electrónico Institucional de la Autoridad	Público	Cadena
Teléfono	Número de Extensión UTPL	Público	Cadena
Celular	Número de Teléfono Celular	Privado	Cadena
Género	Genero de la Autoridad	Público	Cadena
Nacionalidad	Nacionalidad de la Autoridad	Público	Cadena
Dirección	Dirección del Domicilio de la Autoridad	Privado	Cadena
Fecha de Nacimiento	Fecha de Nacimiento de la Autoridad	Público	Fecha
Fecha de Posesión	Fecha en la que Fue Otorgado el Cargo Como Autoridad de la UTPL	Privado	Fecha
Título	Nivel de Formación Académica	Privado	Cadena
Tipo de Autoridad	Grupo al que Pertenece su Cargo	Público	Cadena
Cargo	Nombre Completo del Cargo que Posee Como Autoridad	Público	Cadena

Fuente: recuperado de: https://goo.gl/51CFI5
Elaboración: El Autor

5.4. Modelamiento del Vocabulario

El presente trabajo sigue la metodología planteada en (Piedra et al., 2014) para el correcto modelamiento de un Vocabulario.

Realizar un correcto análisis y descripción de las fuentes de datos, facilitará la construcción y definición de un modelo ontológico que resulta en el vocabulario RDF u ontología adecuada para ser utilizada en los datos enlazados que publicaremos.

5.4.1. Mapeo entre conceptos y entidades de almacenamiento físico

Una vez identificadas las tablas de las bases de datos que anteriormente se analizan y que contienen información sobre las entidades seleccionadas y los atributos que corresponden a cada entidad. Se realiza un mapeo entre atributos, entidades y vocabularios. En la mayoría de los casos estos vocabularios son reutilizados de los ya existentes. El resultado final de esta etapa es contar con un listado de todos los atributos, entidades y vocabularios a emplear en la fase de modelado.

5.4.1.1. Personas

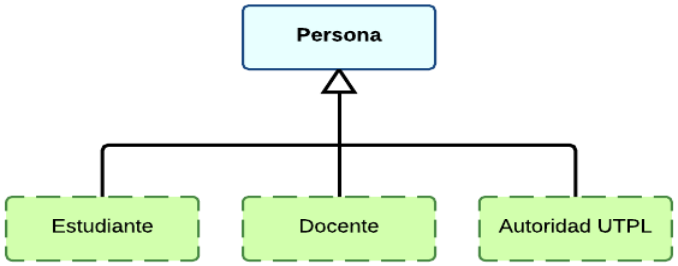


Figura 22: Concepto Persona y Subentidades
Fuente: El Autor
Elaboración: El Autor

La figura 22 ilustra el concepto “PERSONA” el mismo que abarca tres de las cuatro entidades seleccionadas en esta solución, entidades como: Estudiante, Docente, Gobierno General UTPL de las cuales ya se obtuvo un listado de atributos con los cuales se procede a mapear los orígenes de cada uno de éstos. De esta forma se identifican Sistemas y Tablas en los que residan estas entidades. A cada uno de los atributos identificados se le asigna una equivalencia con un vocabulario RDF para posteriormente poder a construir la Ontología de la entidad.

La tabla 26 muestra claramente el mapeo entre los atributos de la entidad persona y las propiedades RDF que las representan.

Tabla 26: Mapeo entre atributos que describen a Personas y propiedades en RDF

PERSONA			
Fuente de Datos		**Equivalencia Vocabulario RDF**	
Sistema	**Tabla**	**Atributo**	**Vocabulario:Propiedad**
SIEC	TDI_ENTIDAD	Nombres	foaf:name
		Apellidos	foaf:FamilyName
		Cédula	dcterms:identifier
		Edad	foaf:Age
		Fecha de Nacimiento	foaf:birthday
		Estado Civil	lodutpl:MaritalStatus
		Lugar de Nacimiento	schema:birthPlace
		Género	foaf:gender
		Email	foaf:mbox
		Teléfono	foaf:phone

Fuente: Unidad de Gestión de Datos Académicos
Elaboración: El Autor

5.4.1.2. Subentidad Estudiante

Estudiante es una subentidad que se desprende de persona y que posee características propias y las heredadas desde su entidad padre.

La tabla 27 muestra claramente el mapeo entre los atributos de la subentidad Estudiante y las propiedades RDF que las representan.

Tabla 27: Mapeo entre atributos que describen a Estudiantes y propiedades en RDF

ESTUDIANTE			
Fuente de Datos		**Equivalencia Vocabulario RDF**	
Sistema	**Tabla**	**Atributo**	**Vocabulario:Propiedad**
SIEC	TDI_ESTUDIANTE	Fecha Inicio de Estudios UTPL	schema:startDate
		Fecha Fin de Estudios UTPL	schema:endDate
		Estado Académico	lodutpl:AcademicStatus
		Nivel Académico	vivo:AcademicDegree

Fuente: El Autor
Elaboración: El Autor

5.4.1.3. Subentidad Docente

Docente es una subentidad que se desprende de persona y que posee características propias y las heredadas desde su entidad padre.

La tabla 28 muestra claramente el mapeo entre los atributos de la subentidad Docente y las propiedades RDF que las representan.

Tabla 28: Mapeo entre atributos que describen a Docentes y propiedades en RDF

DOCENTE			
Fuente de Datos		**Equivalencia Vocabulario RDF**	
Sistema	**Tabla**	**Atributo**	**Vocabulario:Propiedad**
SIEC	TDI_DOCENTE	Fecha Inicio de Estudios UTPL	schema:startDate
		Fecha Fin de Estudios UTPL	schema:endDate
		Estado Laboral	lodutpl:LaboralStatus
		Nivel Académico	vivo:AcademicDegree
		Extensión Telefónica	mads:extension

Fuente: El Autor
Elaboración: El Autor

5.4.1.4. Mapeo entre atributos para Gobierno UTPL y propiedades RDF

El gobierno UTPL define a las autoridades de la institución y estas se convierten en subentidadades de persona y que posee características propias y las heredadas desde su entidad padre.

La tabla 29 muestra claramente el mapeo entre los atributos de la subentidad Docente y las propiedades RDF que las representan.

Tabla 29: Atributos de la Subentidad Autoridad UTPL

AUTORIDAD UTPL (GOBIERNO GENERAL)			
Fuente de Datos		**Equivalencia Vocabulario RDF**	
Sistema	**Tabla**	**Atributo**	**Vocabulario:Propiedad**
SIEC	TDI_CARGO_DOCENTE TDI_CARGO_PERFIL TDI_CARGO TDI_PERFIL	Fecha Inicio de Estudios UTPL	schema:startDate
		Fecha Fin de Estudios UTPL	schema:endDate
		Estado Laboral	lodutpl:LaboralStatus
		Nivel Académico	vivo:AcademicDegree
		Extinción Telefónica	mads:extension
		Cargo	org:Role

Fuente: El Autor
Elaboración: El Autor

5.4.1.5. Componente Educativo

Componente Educativo, entidad de la cual se obtuvo anteriormente un listado de atributos con los cuales se procede a mapear los orígenes de cada uno de éstos, de esta forma se identifican Sistemas y Tablas en los que residan estas entidades. A cada uno de los atributos identificados se les asigna una equivalencia con un vocabulario RDF para posteriormente poder a construir la Ontología de la entidad.
La tabla 30 muestra claramente el mapeo entre los atributos de la entidad Componente Educativo y las propiedades RDF que las representan.

Tabla 30: Atributos y Equivalencia de la Entidad Componente Educativo

COMPONENTE EDUCATIVO			
Fuente de Datos		**Equivalencia Vocabulario RDF**	
Sistema	**Tabla**	**Atributo**	**Vocabulario:Propiedad**
SIEC	TDI_COMPONENTE_EDUCATIVO	Código del Componente	aiiso:Code
		Nombre del Componente	aiiso:name
		Descripción del Componente	aiiso:Description
		Vigencia del Componente	lodutpl:ValidityStatus
		Ciclo Correspondiente del Componente	aiiso:Module
		Número de Créditos del Componente	lodutpl:AcademicCredit
		Departamento Responsable del Componente	aiiso:Department
		Titulación que Oferta el Componente	aiiso:Faculty
		Tipo de Componente	lodutpl:SubjectType

Fuente: El Autor
Elaboración: El Autor

5.5. Diseño del modelo lógico de los datos

Las figuras siguientes muestran el gráfico del vocabulario utilizado para describir cada una de las entidades.

Descripción Personas y Subentidades El gráfico 23 muestra los vocabularios utilizados para describir y representar a las personas y las subentidades como: Estudiante, Docentes y Autoridad UTPL.

Descripción Componente Académico El gráfico 6.5 muestra los vocabularios utilizados para describir a un Componente Académico Ofertado en algún periodo Académico en la UTPL.

5.5.0.1. Vocabularios empleados

La reutilización de vocabularios es uno de los principios de *Linked Data* para lograr una interoperabilidad, es por esto que dentro de las dos ontologías que se muestran en las figuras siguientes se hace uso de vocabularios como:

*** org:** Esta ontología está diseñada para permitir la publicación de información sobre las organizaciones y estructuras organizativas incluidas las organizaciones gubernamentales. Se tiene la intención de proporcionar una

ontología núcleo genérico y reutilizable, que se puede extender o se especializada para su uso en situaciones particulares. **URI:** <https://www.w3.org/TR/vocab-org/>

*** AIISO:** Ofrece clases y propiedades para describir la estructura de la organización interna de una institució n acadé mica. **URI:** <http://vocab.org/aiiso/ schema>

*** FOAF:** Es una ontología legible para las máquinas que describe a las personas, sus actividades y sus relaciones con otras personas y objetos. **URI:** <http://xmlns.com/foaf/spec/>

*** Schema:** Este vocabulario cubre entidades y las relaciones entre entidades y acciones, y pueden ser fácilmente ampliados a través de un modelo de extensión bien documentado. **URI:** <http://schema.org/>

*** Lotutpl:** Linked Open Data UTPL Academia & Research Vocabulary - Este vocabulario representa los principales items curriculares relacionados con la actividad de investigación y academia de un investigador vinculado a una Universidad. **URI:** <http://data.utpl.edu.ec/utpl/lod>

5.5.1. Dominios y rangos de las propiedades

5.5.2. Componente educativo

La tabla 31 muestra el rango y dominio de cada una de las propiedades utilizadas para describir de forma básica un componente académico.

Tabla 31: Rangos y Dominios de la Propiedades Empleadas.

DESCRIPCION DE PROPIEDADES			
Vocabulario	**Propiedad**	**Dominio**	**Rango**
lodutpl	ValidityStatus	aiiso:Subject	xsd:string
lodutpl	SubjectType	aiiso:Subject	xsd:string
lodutpl	AcadeicCredit	aiiso:Subject	xsd:string
aiiso	Module	aiiso:Subject	xsd:string
aiiso	Code	aiiso:Subject	xsd:string
aiiso	name	aiiso:Subject	xsd:string
aiiso	Description	aiiso:Subject	xsd:string
aiiso	Department	aiiso:Subject	xsd:string
aiiso	Faculty	aiiso:Subject	xsd:string

Fuente: El Autor
Elaboracion: El Autor

5.5.3. Docentes, Estudiantes y Autoridades UTPL

La tabla 32 muestra el rango y dominio de cada una de las propiedades utilizadas para describir de forma básica al ente persona y sus subentidades Docente, Estudiante y Autoridad.

Tabla 32: Rangos y Dominios de la Propiedades Empleadas.

DESCRIPCION DE PROPIEDADES			
Vocabulario	**Propiedad**	**Dominio**	**Rango**
lodutpl	AcademicStatus	lodutpl:Estudiante	rdfs:Literal
lodutpl	LaboralStatus	lodutpl:Docente	rdfs:Literal
lodutpl	MaritalStatus	foaf:Person	rdfs:Literal
foaf	Age	foaf:Person	rdfs:Literal
foaf	birthday	foaf:Person	xsd:dateTime
foaf	FamilyName	foaf:Person	rdfs:Literal
foaf	Gender	foaf:Person	rdfs:Literal
foaf	mbox	foaf:Person	rdfs:Literal
foaf	phone	foaf:Person	rdfs:Literal
foaf	name	foaf:Person	rdfs:Literal
dcterms	identifier	foaf:Person	rdfs:Literal
mads	extension	lodutpl:Autoridad	rdfs:Literal
org	role	lodutpl:Autoridad	rdfs:Literal
schema	birthPlace	foaf:Person	rdfs:Literal
schema	endDate	lodutpl:Docente lodutpl:Autoridad	rdfs:Literal
schema	startDate	lodutpl:Docente lodutpl:Autoridad	rdfs:Literal
vivo	AcademicDegree	foaf:Person	rdfs:Literal

Fuente: El Autor
Elaboracion: El Autor

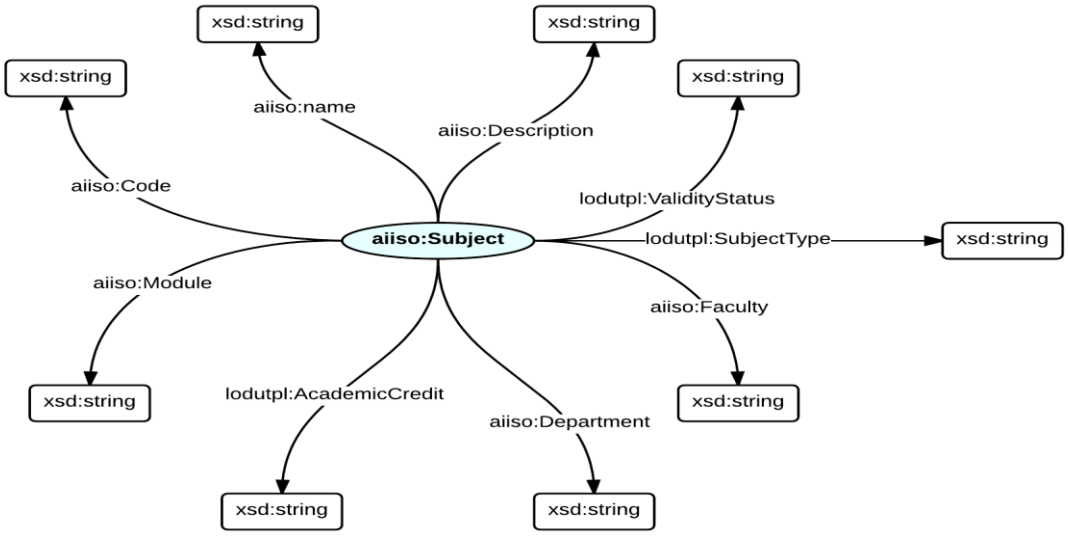


Figura 24: Gráfico del Vocabulario Utilizado Para Describir un Componente Académico
Fuente: El Autor
Elaboración: El Autor

Las figuras 23 y 24 muestra la representación grafía de las dos ontologías utilizadas en el caso de aplicación.

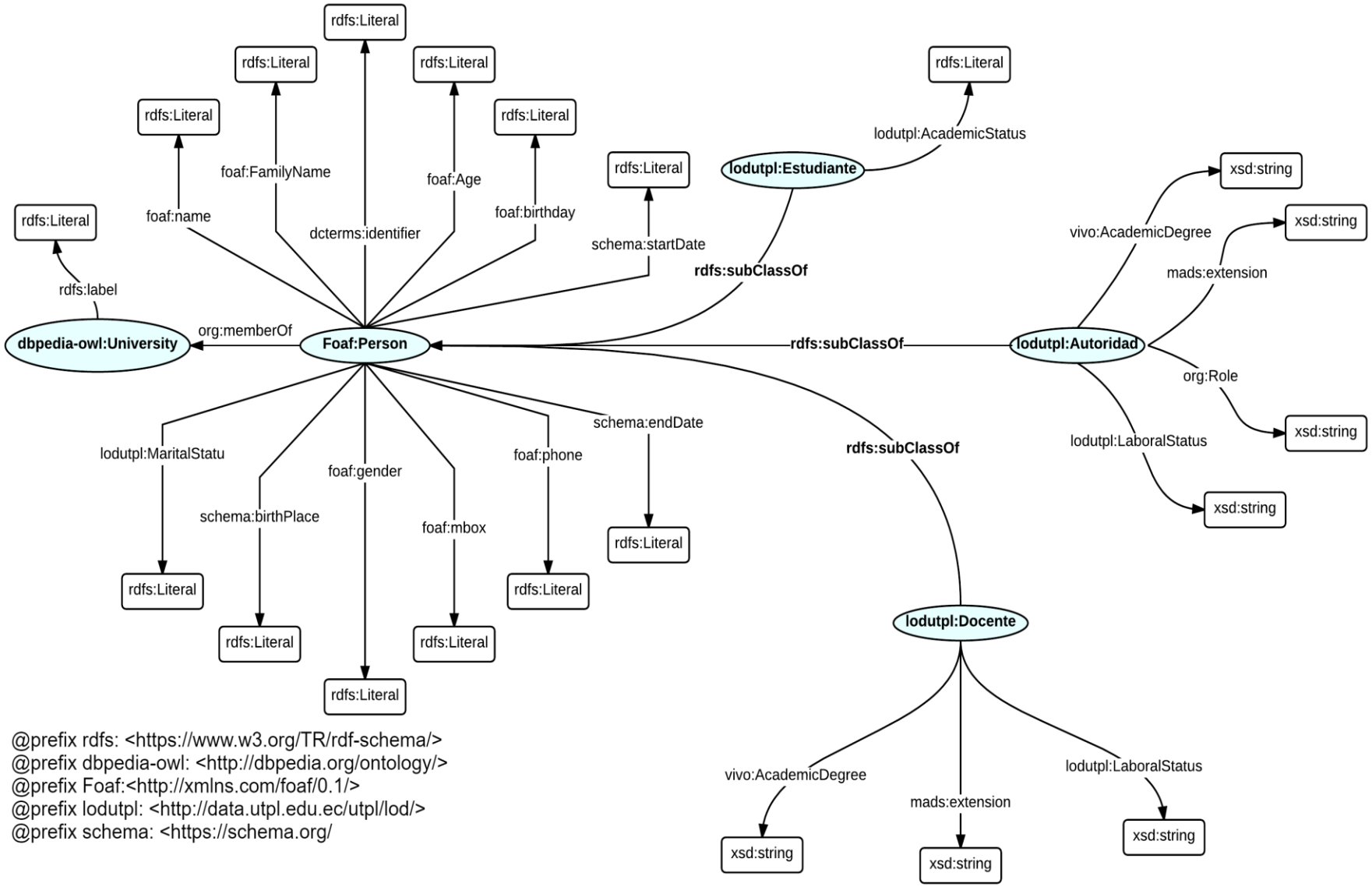


Figura 23: Gráfico del Vocabulario Utilizado Para Describir a las Personas y sus SubEnetidades
Fuente: El Autor
Elaboración: El Autor

5.6. Extracción.

La generación de datos RDF hace necesaria tareas intermedias de extracción y limpieza de datos para lo cual se define un modelo relacional de datos auxiliar como repositorio de los datos que ha pasado por una serie de procesos de depuración para finalmente pasar por un proceso de transformación hacia tripletas RDF.

La extracción de la información en el caso de aplicación se la realizó utilizando técnicas de *web Scrappy* para crear un *Crawler*, conexión directa a la base de datos empleando conectores JDBC y desarrollo de paquetes que incluyen *Script* y Cursores en PL/SQL.

Crawler: Éste se desarrolla en Java 1.8 e implementa la librería Jsoup para lograr el *web Scrappy* a la pagina del Senescyt, lo cual es totalmente legal puesto que la información que reposa en dicho sitio web es de acceso publico. La pagina del Senescyt provee de la información correspondiente a la formación académica de todos y cada uno de los docentes que conforman la UTPL, y así también verificar sus nombres y apellidos registrados correctamente. Se conecta a la tabla de docentes del SIEC y extrae los números de cédula de todos los docentes activos del último periodo académico, una vez extraída esta información el *Crawler* se procede a realizar la petición al servidor del Senescyt el mismo que provee una pagina con toda la información del docente. El *Crawler* procede a extraer toda esta información y posteriormente realiza el almacenamiento en la base de datos del SIEC.

Conexión JDBC: Es una interfase de acceso a bases de datos estándar SQL que proporciona un acceso uniforme a una gran variedad de bases de datos relacionales. Este JDBC permite realizar consultas directas a la base de datos de los sistemas que se señaló anteriormente, datos que serán almacenados en la base de datos del SIEC para su posterior limpieza y desambiguación con la utilización de PL/SQL. con esta tecnología se accedió a los datos de:

- Entorno Virtual de Aprendizaje
- Sistema de Información Académica Científica
- Nuevo Sistema de Gestión Académico
- Sistema de Exalumnos

Paquetes (Script, Cursores): Éstos se encuentran desarrollados en el lenguaje de desarrollo PL/SQL propio de Oracle el mismo que facilita las tareas de limpieza y depuración de los datos.

5.7. Desambiguación y limpieza.

Al contar con toda la data dentro de la base de datos del SIEC es altamente importante y necesario la desambiguación y limpieza de esta data.
PL/SQL *(Procedural Language/Structured Query Language)* es un lenguaje de programación incrustado en Oracle, que permite realizar tareas de extracción, transformación, carga y actualización de los datos almacenados en nuestra base de datos Oracle.
Para la Limpieza de datos del presente proyecto se desarrollaron un total de 3 paquetes que contienen *Scripts* que realizan tareas de extracción, transformación, carga y actualización. Se utilizaron:

Merge: esta función permite crear o actualizar un registro en nuestras tablas dada una condición

Cursores Los cursores permiten recorrer los registros que devuelve una consulta SQL y efectuar operaciones de INSERT, UPDATE Y DELETE, dependiendo del tipo de cursor que se utilice

5.8. Transformación.

La generación de datos RDF se la realizó mediante la utilización de un módulo desarrollado en java que hace uso del API de *Apache Marmotta*, el mismo que está construido con *Sesame*, un *framework* para *Linked Data*. Sesame permite la reutilización de vocabularios ya definidos y la creación de un modelo propio para cada proyecto, la escritura y lectura de RDF en diferentes formatos y muchas características más.

La figura 25 muestra una vista general de como se organizan los paquetes del proyecto en base a la arquitectura planteada.

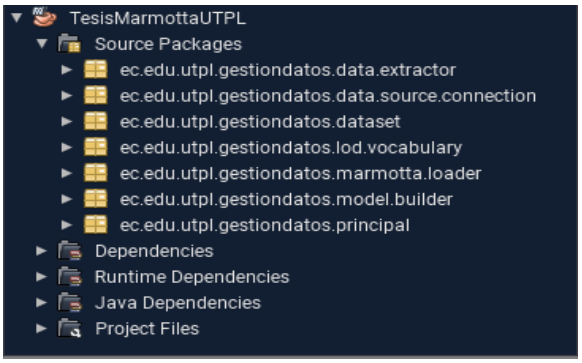


Figura 25: Estructura del módulo distribuida en paquetes
Fuente: El Autor
Elaboración: El Autor

Este módulo se conecta a la base de datos del SIEC y extrae la información ya depurada de las tablas, la misma que procesa y transforma agregado cada uno de los vocabularios y sus propiedades. Una vez finalizada la transformación de la data se seria liza en formato Turtle en un *ByteArrayOutputStream* para luego transformarlo en un *String* con codificación “UTF-8”

La figura 26 muestra la parte final del código Java donde se convierte todos los datos de un modelo relacional a RDF

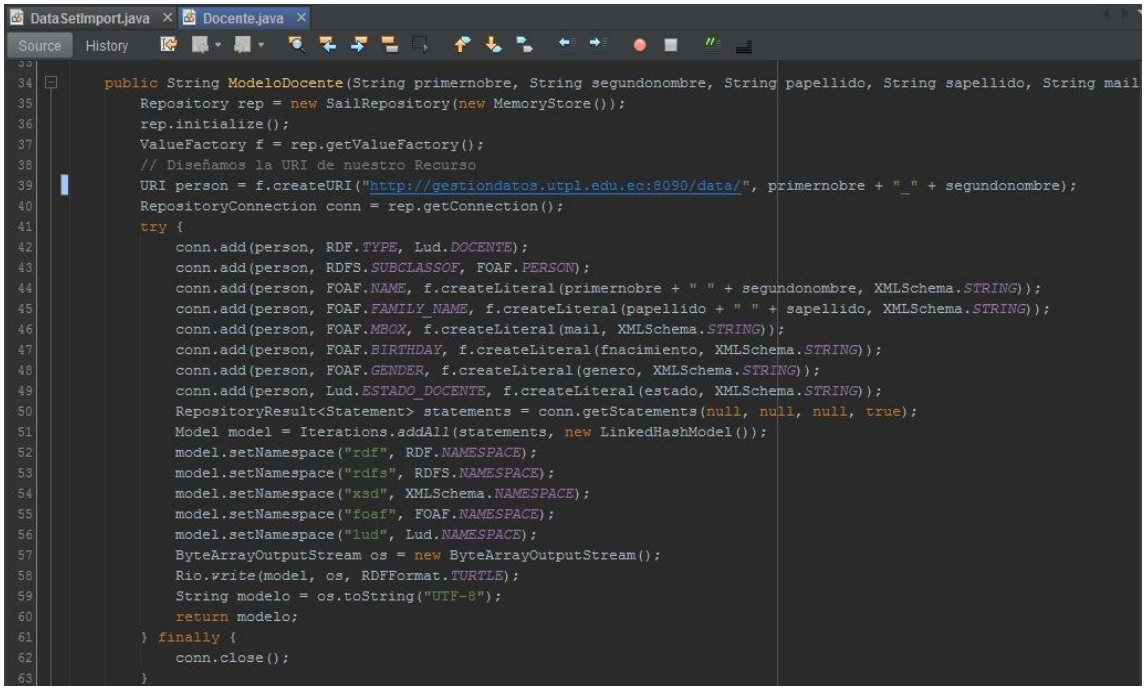


Figura 26: Parte del Código de la clase Docente
Fuente: El Autor
Elaboración: El Autor

5.9. Carga.

Una vez listo el modelo se envía para su carga en el *triplestore* de *Apache Marmota* (WIKI), para lo cual se define el URL del triple store, el contexto y las credenciales de autenticación. El proceso de carga se realiza utilizando un *POST Request*, medio por el cual viajan las credenciales y la data

La figura 27 muestra del código Java de la clase CargaDatos donde se envían los parámetros de seguridad y el dataset hacia el triplestore de Apache Marmotta.

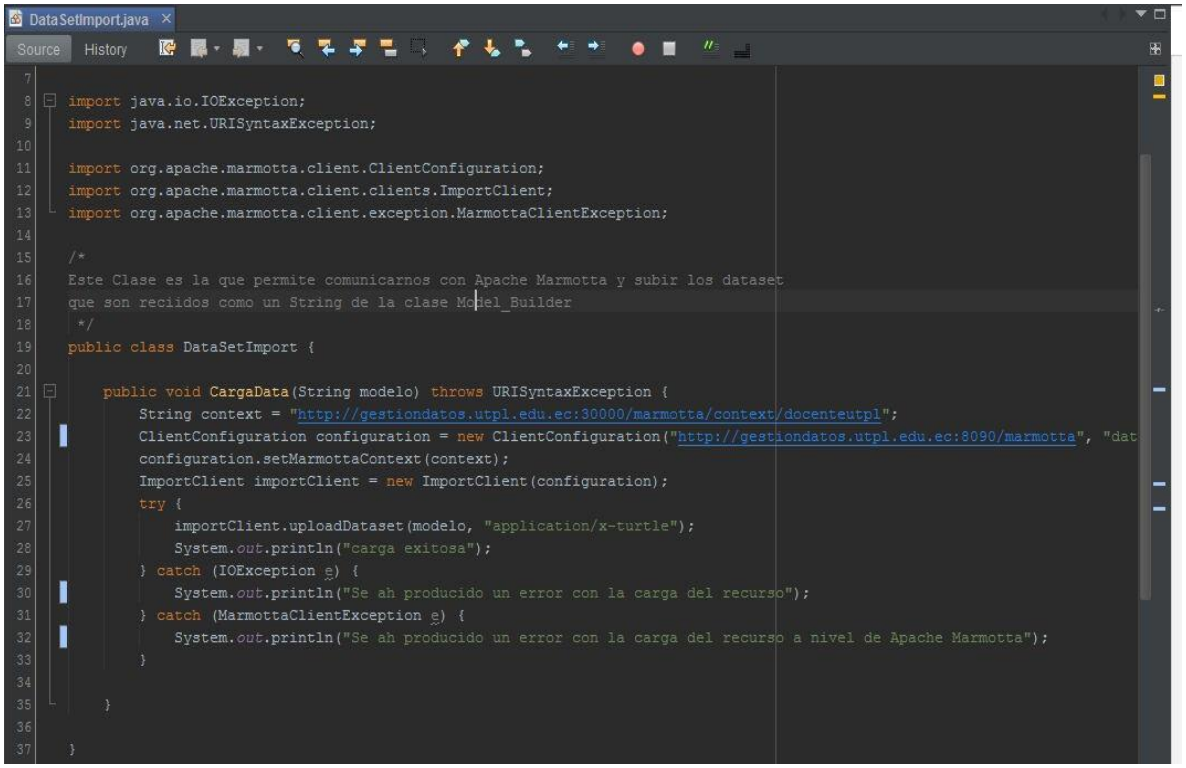


Figura 27: Parte del Código de la clase DataSetImport
Fuente: El Autor
Elaboración: El Autor

5.10. Publicación de datos RDF

La publicación de los datos se encuentran en un dominio Público que se puede acceder desde el *SPARQL endpoint de Marmotta*.

La figura 28 muestra la interfaz de usuario de Apache Marmotta para el SPAQRL endpoint y desde la cual se ejecuta un consulta sencilla cuyos resultados se muestran en la figura 29.

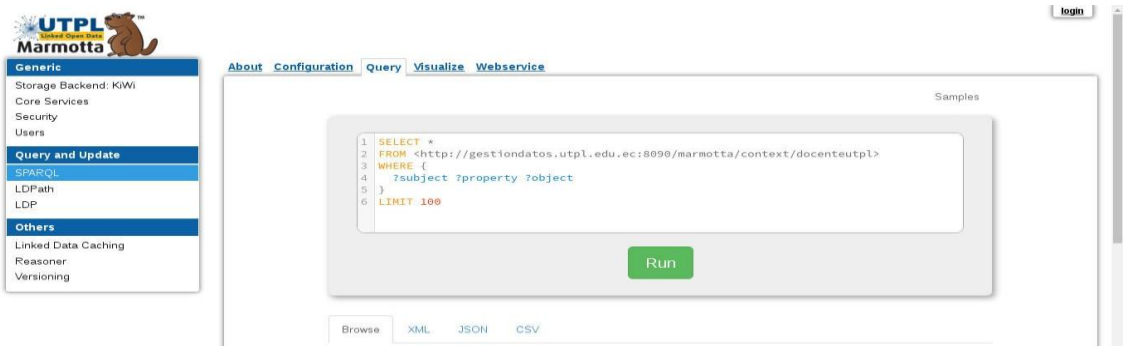


Figura 28: Consulta al*SPARQL endpoint*de Marmota sobre el contexto Docentes
Fuente: Recuperado de: https://goo.gl/GhOLt3
Elaboración: El Autor

subject	property	object
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	rdf:type	http://data.utpl.edu.ec/lud/0.1/Docente
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	rdfs:subClassOf	foaf:Person
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	foaf:name	NORA ELIZABETH ^^xsd:string
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	foaf:familyName	VEGA CHAMBA ^^xsd:string
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	foaf:mbox	nevega@utpl.edu.ec ^^xsd:string
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	foaf:birthday	1984-07-18 00:00:00.0 ^^xsd:string
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	foaf:gender	F ^^xsd:string
http://gestiondatos.utpl.edu.ec:8090/data/nora_elizabeth	http://data.utpl.edu.ec/lud/0.1/Estado_Docente	A ^^xsd:string
http://gestiondatos.utpl.edu.ec:8090/data/elsa_silvana	rdf:type	http://data.utpl.edu.ec/lud/0.1/Docente
http://gestiondatos.utpl.edu.ec:8090/data/elsa_silvana	rdfs:subClassOf	foaf:Person

< pref Rows 1 to 10 of overall 100 next >

Figura 29: Resultados de la consulta al*SPARQL endpoint de Marmota* sobre el contexto Docentes
Fuente: Recuperado de: https://goo.gl/GhOLt3
Elaboración: El Autor

5.11. Comentarios Finales

AL finalizar cada una de las actividades que este capítulo describe se puede concluir que la tarea de agrupar datos de varias fuentes heterogéneas de información resulta ser una de las tareas más complicadas durante el proceso de publicación de datos enlazados o Linked Data. Sin embargo, finalmente lo que resulta ser de gran importancia dentro de todo este capítulo es la capacidad de organización final de los datos recolectados, de forma que toda esa información redundante e inconsistente quede fuera de repositorio final de datos para que los procesos de transformación y carga se ejecuten sin mayores complicaciones.

CAPÍTULO VI: IMPLEMENTACIÓN Y PRUEBAS

6.1. Introducción

Este último capítulo mediante la implementación del caso de aplicación en un ambiente real de producción pretende realizar una serie de pruebas que permitan establecer de manera más clara la viabilidad de la solución planteada. Los resultados de estas pruebas son comparados con los métodos tradicionales de acceso a la información y con el API de servicio web que posee la UGTI.

Los resultados son producto de una serie de consultas realizadas en base a los casos de uso descritos en el sección 4.1.1 de este trabajo de fin de titulación.

Finalmente, el capítulo detalla dentro de una subsección de que forma la solución ataca a cada una de las causas del problema de interoperabilidad entre aplicaciones que la UTPL tiene.

6.2. Puesta en producción

La puesta en producción del prototipo que la solución plantea en la presente investigación se divide en dos partes. La primera es la centralización de la información descrita semánticamente y alojada sobre una plataforma para Linked Data. La misma que será la única fuente de datos que brinde información de las cuatro entidades seleccionadas en el caso de aplicación.

La segunda parte es el desarrollo de un RESTFul API el mismo que se encuentra programado en Java y que accede a la información almacenados en la Base de datos de la Unidad de Gestión de Datos Académicos (UGDA).

El API RESTFul es el encargado de presentar la información vía *RESTFul Web Services*. Este presenta los datos en diferentes formatos de serialización. Los formatos de serialización habilitados son los siguientes: JSON, JSON-LD, RDF, XML.

La serialización de salida de los resultados queda al criterio del desarrollador que haga uso de la solución como insumo de datos para sus aplicaciones. Lo que se pretende es ofrecer una amplia y flexible capacidad de adaptación de la solución a las diferentes tecnologías existentes y de esta forma motivar a los desarrolladores a utilizar la solución como mecanismo de interoperabilidad de datos entre las aplicaciones.

Para desplegar la arquitectura propuesta en la solución se utiliza los servidores asignados a la UGDA.

A continuación se detalla de forma técnica cada uno de los servidores sobre los que se implementa el prototipo de la solución.

6.2.1. Servidor de base datos SQL de la UGDA

El servidor de base de datos de la UGDA es en donde se encuentra almacenada toda información académica de la UTPL. Este servidor posee las siguientes características:

Sistema Operativo: Red Hat Enterprise Linux 5.5.

Memoria Ram: 12 MB.

Arquitectura del Sistema Operativo: x86_64.

Versión de kernel linux: 2.6.18.

Rack 1: 172.16.53.30.

Rack 2: 172.16.53.31.

Base de Datos: Oracle Express Edition.

Versión de Base de Datos: 11g.

6.2.2. Servidor para aplicaciones

El servidor de aplicaciones sobre el cual esta desplegado el API y la plataforma para Linked Data *Apache Marmotta* posee las siguientes características:

Sistema Operativo: Red Hat Enterprise Linux 5.5.

Memoria Ram: 12 MB.

Arquitectura del Sistema Operativo: x86_64.

Versión de kernel linux: 2.6.18.

Java Versión: 1.8.

Servidor web: Tomcat

Versión Servidor web: 8.5.5

Dominio: gestiondatos.utpl.edu.ec.

URL de Acceso Api: `http://gestiondatos.utpl.edu.ec/service/`

URL de Acceso a Apache Marmotta: `http://gestiondatos.utpl.edu.ec/marmotta/`

La figura 30 muestra un esquema general de los equipos de hardware utilizado.

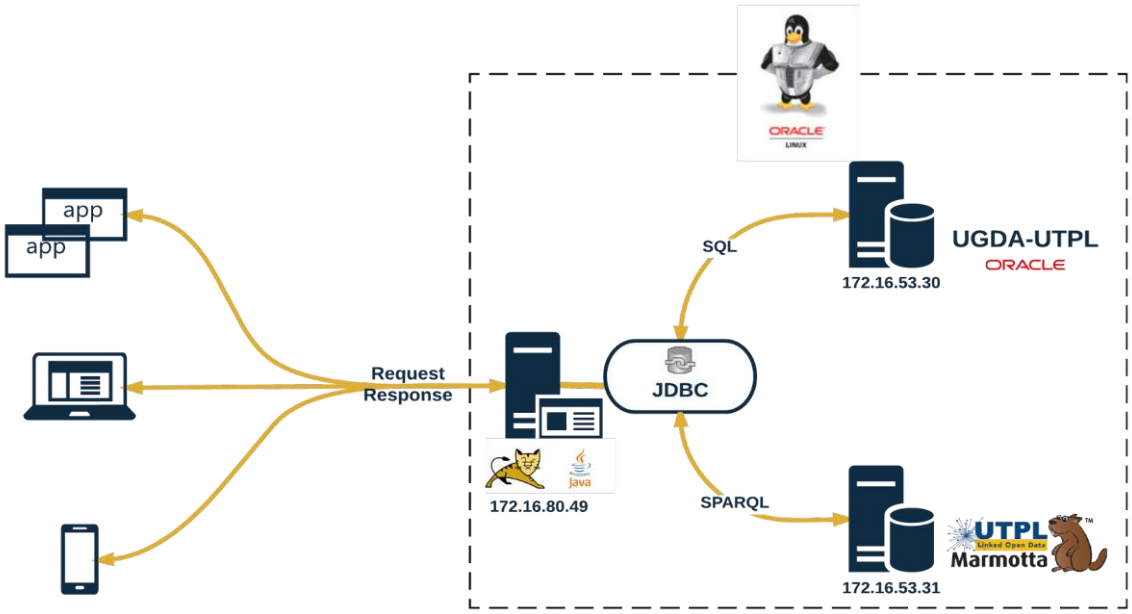


Figura 30: Arquitectura de hardware utilizada
Fuente: El Autor
Elaboración: El Autor

La figura 31 muestra como el RESTful API consulta información de los repositorios centralizados identificando la Base de datos del SIEC como el repositorio de los datos para consultas SQL y de Marmotta LDP específicamente Kiwi *triplestore* como el repositorio de los datos en formato de tripletas.

El anexo 1 detalla todo el proceso de instalación y las configuraciones necesarias para que *Apache Marmotta* LDP funcione correctamente dentro de un ambiente de producción.

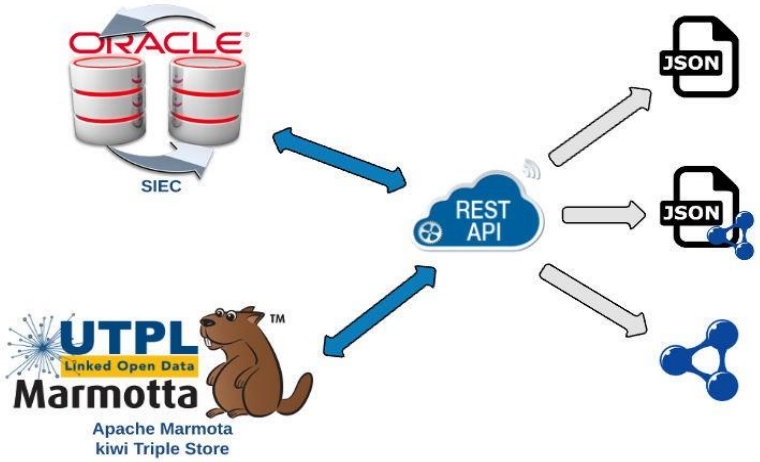


Figura 31: Interacción del API Rest
Fuente: El Autor
Elaboración: El Autor

6.2.3. Configuraciones.

Las configuraciones e instalaciones de software en los servidores descritos anteriormente se detallan a continuación.

Instalación Java: La arquitectura de software hace mención exclusiva a una serie de tecnologías que únicamente funcionan sobre el lenguaje de programación java. La versión que se utiliza es java 8 y su JDK 1.8 instalados en el servidor de aplicaciones.
La instalación empleó el archivo jre_8u91_linux_x64 el mismo que es un archivo de extensión RPM utilizado por esta distribución de Linux que posee el servidor.

Instalación Apache Tomcat El servidor de aplicaciones seleccionado es Apache Tomcat en su versión 8.0.24 y del cual se descargaron los archivos binarios y que se alojaron en el directorio opt del sistema operativo.
Existen dos instancias de Tomcat corriendo sobre el servidor una para el RESTful API y otra para *Apache Marmotta* de esta forma si por alguna razón se debe realizar actualizaciones o mantenimiento de alguna de estas aplicaciones únicamente deja de funcionar esa instancia y esa aplicación.

Apache Marmotta: Esta es la plataforma para *Linked Data* seleccionada, de la misma que se realiza una mención especial dentro del capítulo 1 en la sección 1.9 .

Apache Marmotta Platform o mayormente conocida como Marmotta es una plataforma abierta para *Linked Data* que su principal objetivo es proporcionar una implementación abierta de una plataforma de *Linked Data* que pueda ser usada, ampliada y desplegada fácilmente por organizaciones que desean publicar *Linked Data* o construir aplicaciones personalizadas en Linked Data.

Se opta por *Apache Marmotta* como la plataforma de publicación de datos por la arquitectura, siendo ésta una Arquitectura Orientada a Servicios (SOA) que utiliza *CDI/Weld service framework* para la inyección de contextos y dependencias.

Al estar desarrollada sobre una arquitectura SOA la plataforma provee una capa de servicios y de web services que pueden ser aprovechados por la organización que la implemente y de esta forma buscar la interoperabilidad entre la plataforma y sus aplicaciones.

La figura 32 muestra un esquema general de la arquitectura de Apache Marmotta.

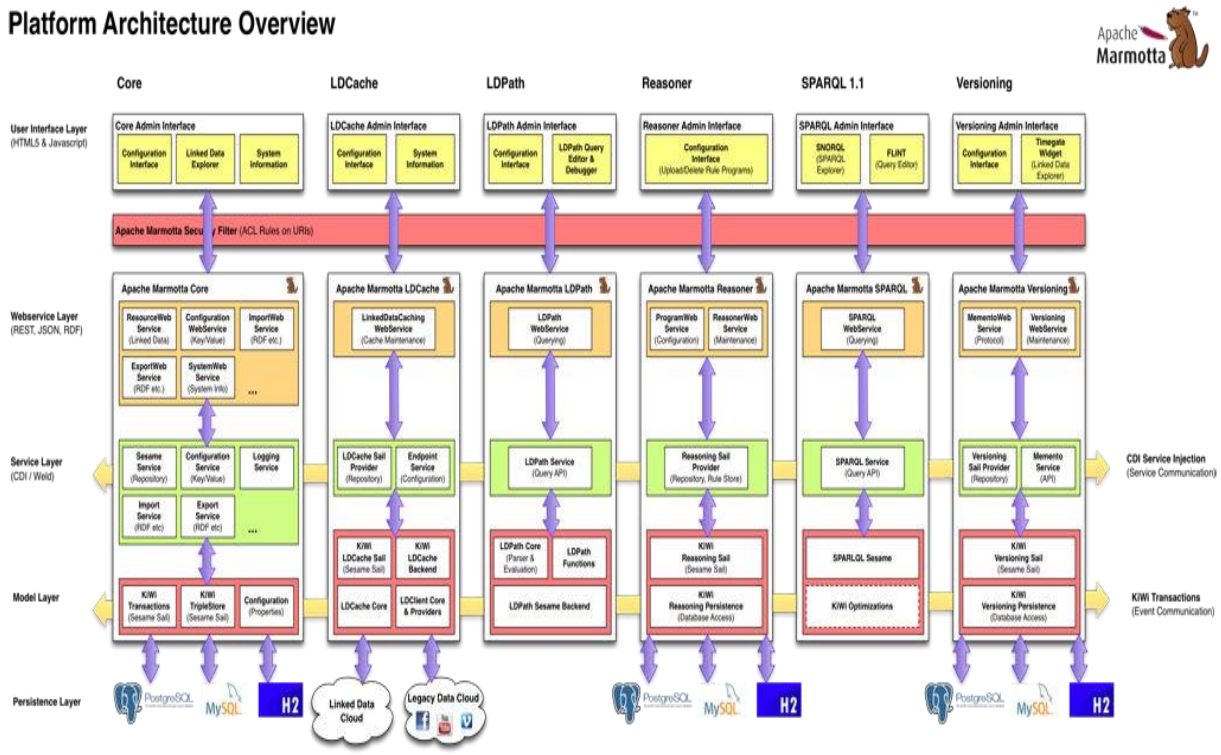


Figura 32: Visión general sobre la arquitectura de Apache Marmotta
Fuente: Recuperado de: https://goo.gl/GhOLt3
Elaboración: El Autor

Los módulos de *Apache Marmotta* se comunican en diferentes niveles utilizando diferentes medios: en el nivel de persistencia, el sistema utiliza las transacciones de kiwi para enviar notificaciones sobre las actualizaciones de triples de los almacenes; en el nivel de servicio, la notificación de inyección de servicio y eventos CDI se utiliza para enviar mensajes.

Para la configuración más básica *(Linked Data Server)*, solo se necesita el módulo principal. En la mayoría de configuraciones sin embargo, si se desea añadir módulos adicionales. Para activar / desactivar un módulo en la configuración, solo tiene que añadir el archivo .jar a la ruta de clases de la aplicación web (WEB–INF/lib)

6.3. Pruebas

Es de suma importancia realizar pruebas en el software que se desarrolla con la finalidad de encontrar de forma prematura errores que podrían convertirse en grandes problemas dentro de un entorno de producción real. Es por esto que la solución planteada en este trabajo de fin de titulación se ha sometido a pruebas de funcionalidad más conocidas como Prueba Unitaria y Pruebas de Rendimiento.

Las pruebas unitarias se las realiza para comprobar que todas las características descritas en los requerimientos funcionales se ejecuten correctamente. Y las pruebas de rendimiento

se efectúan para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo.

El objetivo principal de realizar esta serie de pruebas es demostrar como la solución propuesta puede realizar todas las funciones que realizaría una alternativa tradicional y que los tiempos en los que se ejecuta son mejores. Y de igual forma poder detectar a tiempo errores dentro del código.

Las pruebas se las ejecuto durante las etapas de: Transformación, Carga y Publicación de datos. Las siguientes subsecciones detallan los resultados en cada una de estas estepas.

6.3.1. Transformación y carga.

Para realizar la transformación de los datos a una estructura semántica se utilizó tecnología del lenguaje de programación Java. El *framework SESAMA* permite armar una descripción de un recurso en RDF para su posterior serialización en los diferentes formatos que la W3C determina para Linked Data.

Al finalizar toda la transformación de la data se procede a cargar el RDF sobre el *triplestore* de *Apache Marmotta* la plataforma para *Linked Data* que esta investigación selecciona como repositorio de datos.

La tabla 33 muestra los resultados producto de la generación y carga de un total de 271.140.00 tripletas que corresponden a los datos de las cuatro entidades seleccionadas en el caso de aplicación.

Tabla 33: Tiempos y recursos, transformación y carga

Proceso	Número de Tripletas	Formato de Serialización	Memoria Ocupada	Tiempo de Ejecución
Transformación y Carga	271.140.00	application/x-turtle	7 MB	1:88

Fuente: El Autor
Elaboración: El Autor

El convertir un total de tres mil registros a 38.035 tripletas y al mismo tiempo cargarlas sobre una plataforma de publicación de datos enlazados en 2.48 minutos es considerado un tiempo bastante bueno considerando el volumen de datos.

La figura 33 muestra el *output* de la aplicación el mismo que nos permite verificar la veracidad de la información que la tabla 33 muestra.

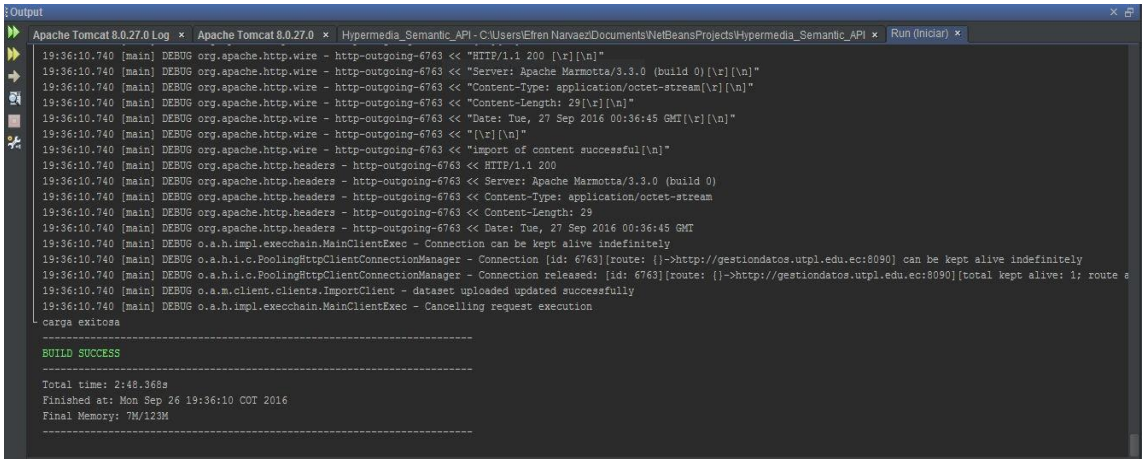


Figura 33: Información de salida producto del proceso de transformación y carga
Fuente: El Autor
Elaboración: El Autor

La figura que se muestra a continuación detalla toda información de un recurso accediendo directamente desde su URI.

Apache Marmotta realiza una negociación de contenido básica en la cual brinda la opción de presentar los datos un recurso como tripletas en una pagina HTML o un RDF. Esta negociación de contenido depende de como se acceda al recurso.

La figura 34 muestra la visualización de la información de un recurso en forma de una pagina web puesto que se accede al URI de dicho recurso desde un navegador.

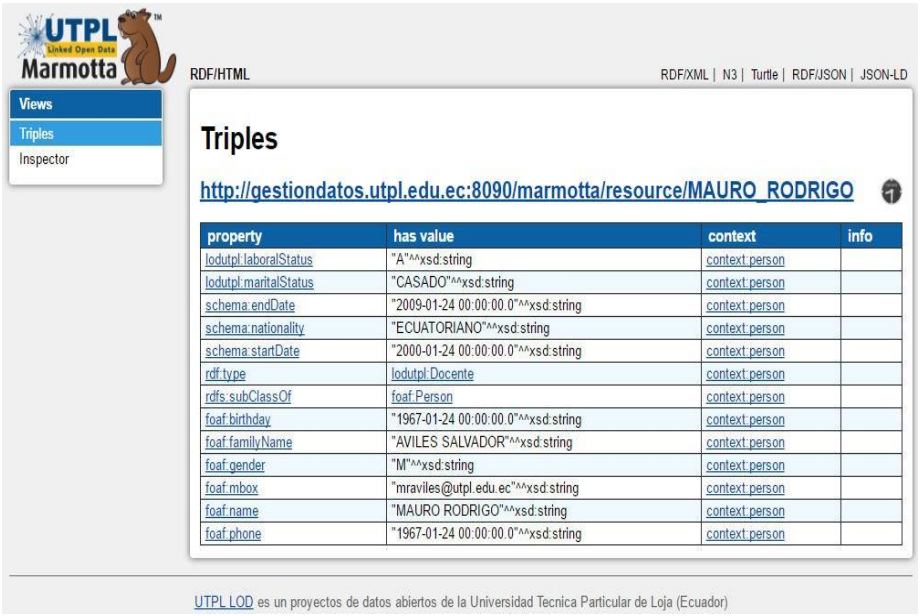


Figura 34: Visualización de la infracción de un recurso
Fuente: Recuperado de:https://goo.gl/dSIvPy
Elaboración: El Autor

Apache Marmotta dentro de su interfaz web de administración proyecta una serie de visualizaciones de los datos almacenados los cuales pueden ser accedidos por los usuarios registrados. Marmotta posee un *SPARQL endponit* al cual se accede desde la interfaz web en el cual se pueden escribir todo tipo de consultas SPARQL sin importar la complejidad.

La figura 35 brinda una perspectiva del tamaño de cada contexto de datos que *Apache Marmotta* está gestionando actualmente. Esto lo presenta con la finalidad de mantener un control de la información que la plataforma gestiona y presenta a través de su *SPARQL endponit*.

Figura 35: Visualización de Contextos Marmotta
Fuente: Recuperado de:https://goo.gl/dSIvPy
Elaboración: El Autor

La finalidad de los puntos mencionados anteriormente es comprobar el correcto funcionamiento del módulo de transformación y carga de la solución y adicionalmente la correcta instalación y configuración de la plataforma para *Linked Data Apache* Marmotta.

6.3.2. Consultas SPARQL endponit.

La consulta de esta información se la puede realizar de dos maneras diferentes, la primera accediendo directamente al *SPARQL endponit* que *Apache Marmotta* provee y ejecutar consultas más elaboradas y complejas y la segunda acceder al API de servicios para acceder a los catálogos que provee consultas predefinidas siento esta última la que contiene los datos actualizados en tiempo real sin importar el formato en el que se desee salga el resultado de la consulta.

Las consultas de los datos desde la primera opción se la realiza accediendo a la interfaz de usuario del módulo SPARQL de *Apache Marmotta* alojada en la siguiente dirección: http://

gestiondatos.utpl.edu.ec/marmotta/sparql/admin/squebi.html

Esta interfaz permite ejecutar consultas SPARQL de todo tipo de complejidad y obtener el resultado de la misma en diferentes formatos. Cabe mencionar que estos datos son producto de un proceso de carga masiva de información y transformación de datos realizado el día 27 de septiembre del 2016.

La figura 36 muestra la consola en donde de escriben las consultas y las cuales al momento tienen un tiempo de acceso igual 0.268 segundos.

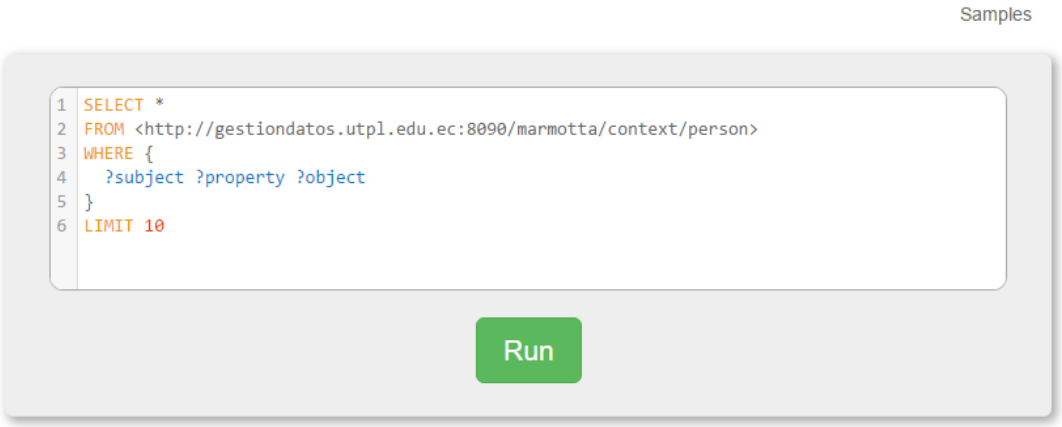


Figura 36: Consola de la interfaz gráfica para la creación de consultas SPARQL
Fuente: Recuperado de:https://goo.gl/dSIvPy
Elaboración: El Autor

6.3.3. Consultas API

La segunda opción con la que se cuenta para la consulta de estos datos es mediante la invocación de uno de los servicios que provee el API de Hipermedia desarrollado sobre la arquitectura planteada como parte de la solución en el presente trabajo de investigación.

Las consultas se realizaron sobre un registro en específico y definiendo el formato de salida de los datos producto de la consulta.

6.3.3.1. Tiempo de respuesta con JSON

Al presentar la información en un formato estándar de JSON se obtiene un tiempo de respuesta de 96 ms, quedando este registro en cache por un periodo de 15 minutos para que la próxima vez que se acceda a éste, el tiempo de respuesta sea más rápido.

La figura 37 muestra la llamada al API REST solicitando información de un recurso en especifico.

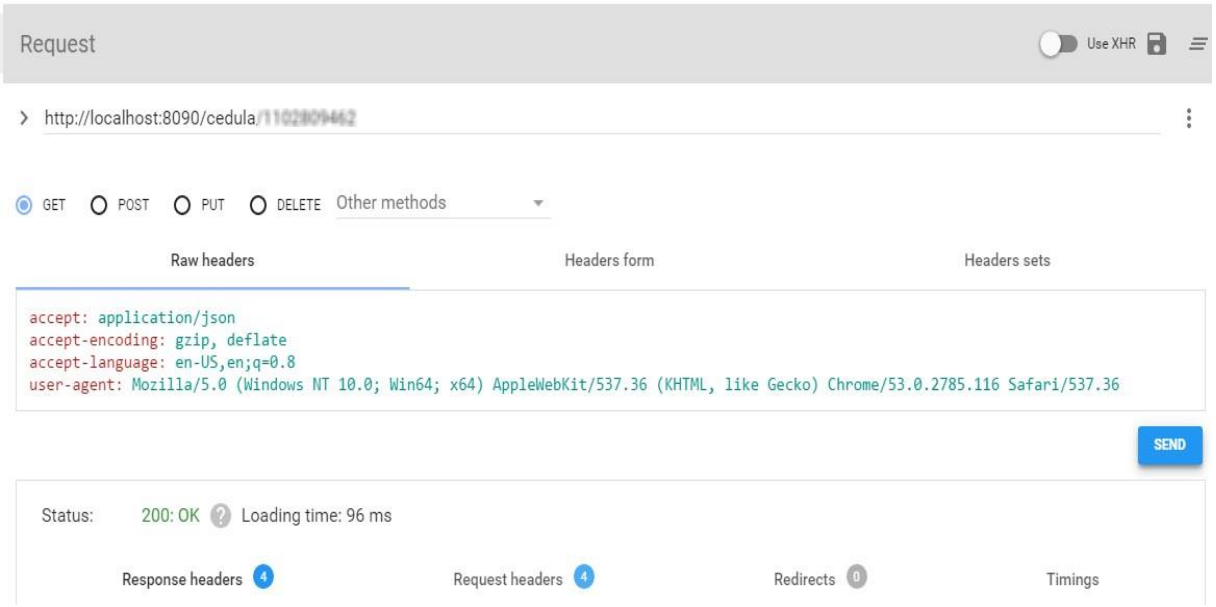


Figura 37: Tiempo de respuesta con un formato de salida JSON
Fuente: El Autor
Elaboración: El Autor

La figura 38 muestra el resultado en formato JSON de la llamada al API REST que se muestra en la figura 37.

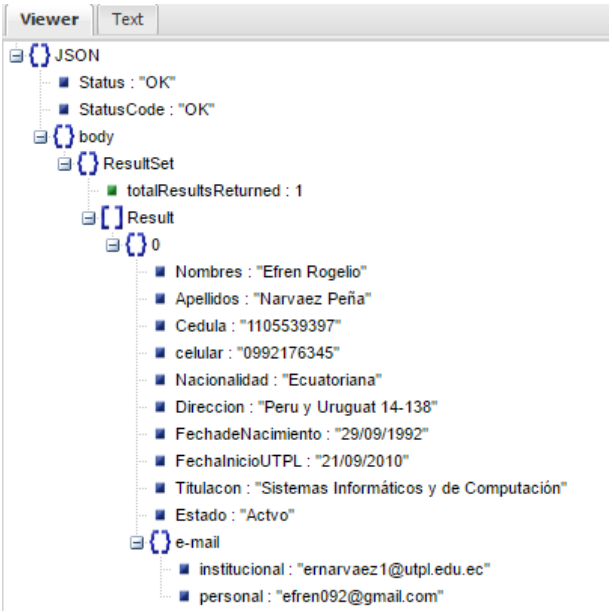


Figura 38: Datos Básicos de Contacto de un Estudiante en Formato JSON
Fuente: El Autor
Elaboración: El Autor

6.3.3.2. Tiempo de respuesta con JSON-LD

Al presentar la información en un formato de JSON-LD propio de *Linked Data* se obtiene un tiempo de respuesta de 90 ms, quedando este registro en cache por un periodo de 15 minutos

para que la próxima vez que se acceda a éste, el tiempo de respuesta sea más rápido.

La figura 39 muestra la llamada al API REST solicitando información de un recurso en específico.

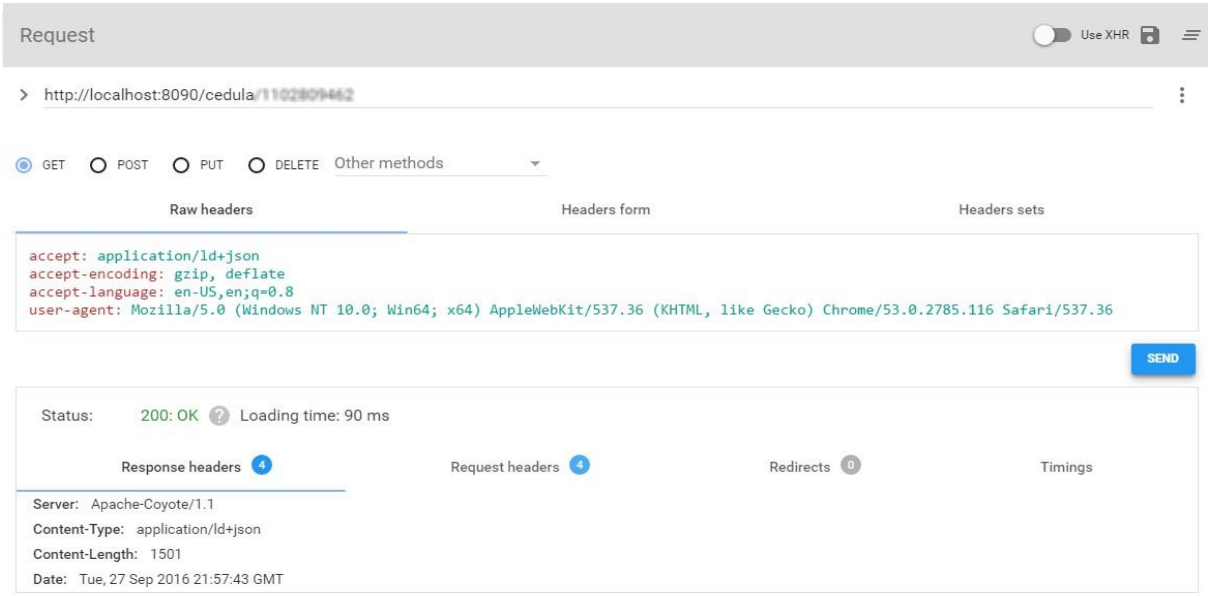


Figura 39: Tiempo de respuesta con un formato de salida JSON-LD
Fuente: El Autor
Elaboración: El Autor

La figura 40 muestra el resultado en formato JSON-LD de la llamada al API REST que se muestra en la figura 39.

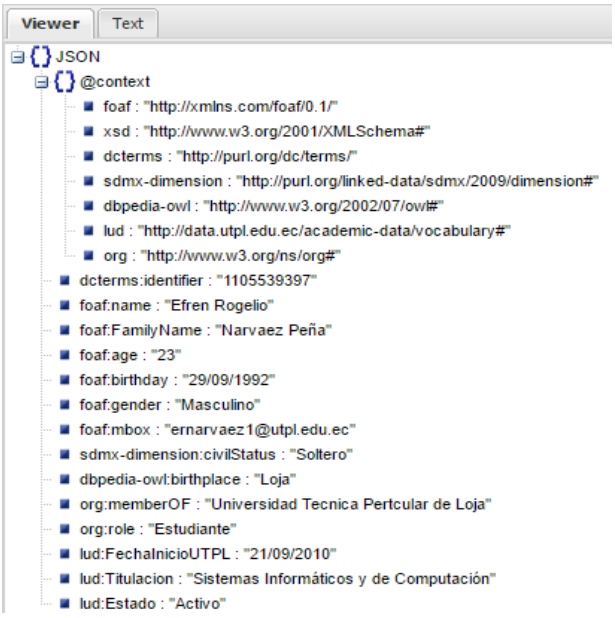


Figura 40: Datos Básicos de Contacto de un Estudiante en Formato JSON-LD
Fuente: El Autor
Elaboración: El Autor

6.3.3.3. Tiempo de respuesta con RDF

Al presentar la información en un formato de RDF propio de *Linked Data* se obtiene un tiempo de respuesta de 90 ms, quedando este registro en cache por un periodo de 15 minutos para que la próxima vez que se acceda a éste, el tiempo de respuesta sea más rápido.

La figura 41 muestra la llamada al API REST solicitando información de un recurso en especifico.

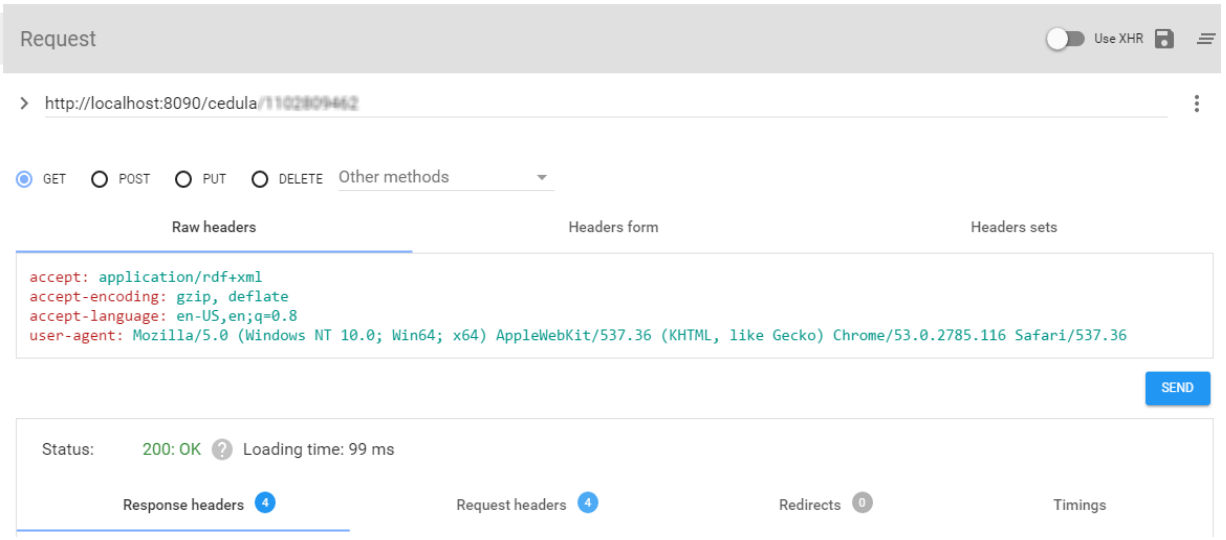


Figura 41: Tiempo de respuesta con un formato de salida RDF
Fuente: El Autor
Elaboración: El Autor

La figura 42 muestra el resultado en formato RDF de la llamada al API REST que se muestra en la figura 41.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:dbpedia-owl="http://www.w3.org/2002/07/owl#"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:lud="http://data.utpl.edu.ec/academic-data/vocabulary#"
   xmlns:org="http://www.w3.org/ns/org#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:sdmx-dimension="http://purl.org/linked-data/sdmx/2009/dimension#"
>
  <rdf:Description rdf:nodeID="Ne568abca1188492493df1b1aad680f4c">
    <foaf:birthday>29/09/1992</foaf:birthday>
    <foaf:FamilyName>Narvaez Peña</foaf:FamilyName>
    <dbpedia-owl:birthplace>Loja</dbpedia-owl:birthplace>
    <foaf:name>Efren Rogelio</foaf:name>
    <org:role>Estudiante</org:role>
    <sdmx-dimension:civilStatus>Soltero</sdmx-dimension:civilStatus>
    <org:memberOF>Universidad Tecnica Pertcular de Loja</org:memberOF>
    <foaf:age>23</foaf:age>
    <lud:FechaInicioUTPL>21/09/2010</lud:FechaInicioUTPL>
    <lud:Titulacion>Sistemas  Informáticos y de Computación</lud:Titulacion>
    <foaf:mbox>ernarvaez1@utpl.edu.ec</foaf:mbox>
    <lud:Estado>Activo</lud:Estado>
    <dcterms:identifier>1105539397</dcterms:identifier>
    <foaf:gender>Masculino</foaf:gender>
  </rdf:Description>
</rdf:RDF>
```

Figura 42: Datos Básicos de Contacto de un Estudiante en Formato RDF
Fuente: El Autor
Elaboración: El Autor

6.3.3.4. Tiempo de respuesta con XML

Al presentar la información en un formato de XML tradicional se obtiene un tiempo de respuesta de 128 ms, quedando este registro en cache por un periodo de 15 minutos para que la próxima vez que se acceda a éste, el tiempo de respuesta sea más rápido.

La figura 43 muestra la llamada al API REST solicitando información de un recurso en especifico.

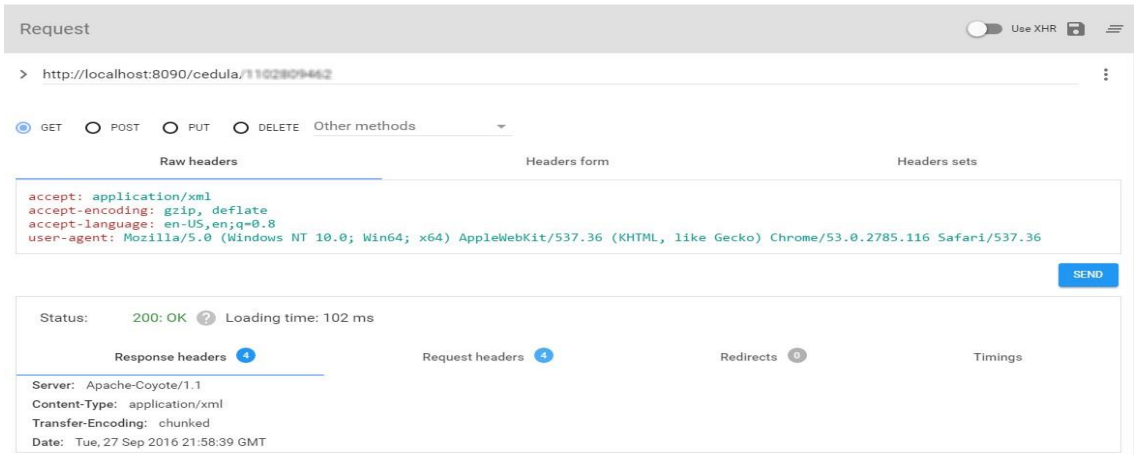


Figura 43: Tiempo de respuesta con un formato de salida XML
Fuente: El Autor
Elaboración: El Autor

La figura 44 muestra el resultado en formato XML de la llamada al API REST que se muestra en la figura 43.

```
<?xml version="undefined" encoding="undefined"?>
<Status>OK</Status>
<StatusCode>OK</StatusCode>
<body>
    <ResultSet>
        <totalResultsReturned>1</totalResultsReturned>
        <Result>
            <Nombres>Efren Rogelio</Nombres>
            <Apellidos>Narvaez Peña</Apellidos>
            <Cedula>1105539397</Cedula>
            <celular>0992176345</celular>
            <Nacionalidad>Ecuatoriana</Nacionalidad>
            <Direccion>Peru y Uruguat 14-138</Direccion>
            <FechadeNacimiento>29/09/1992</FechadeNacimiento>
            <FechaInicioUTPL>21/09/2010</FechaInicioUTPL>
            <Titulacon>Sistemas Informáticos y de Computación</Titulacon>
            <Estado>Actvo</Estado>
            <e-mail>
                <institucional>ernarvaez1@utpl.edu.ec</institucional>
                <personal>efren092@gmail.com</personal>
            </e-mail>
        </Result>
    </ResultSet>
</body>
```

Figura 44: Datos Básicos de Contacto de un Estudiante en Formato XML
Fuente: El Autor
Elaboración: El Autor

Las llamadas al API REST que se muestra en las figuras 38-45 demuestran el correcto funcionamiento del prototipo desarrollado sobre la arquitectura planteada como parte de la solución a los problemas de interoperabilidad de la UTPL.

Luego de mostrar como el RESTful API y sus tiempos de respuesta en los diferentes formatos de serialización, se presenta la siguiente gráfica comparativa con los tiempos de respuestas de otras soluciones que actualmente se están empleando en la UTPL. La información del tiempo de respuesta del API-UGTI es tomado del catálogo de servicios que posee el cual se encuentra alojado en *SharePoint* de la UTPL.

la figura 45 muestra los diferentes tiempos de respuesta entre la solución propuesta y las alternativas existentes actualmente.

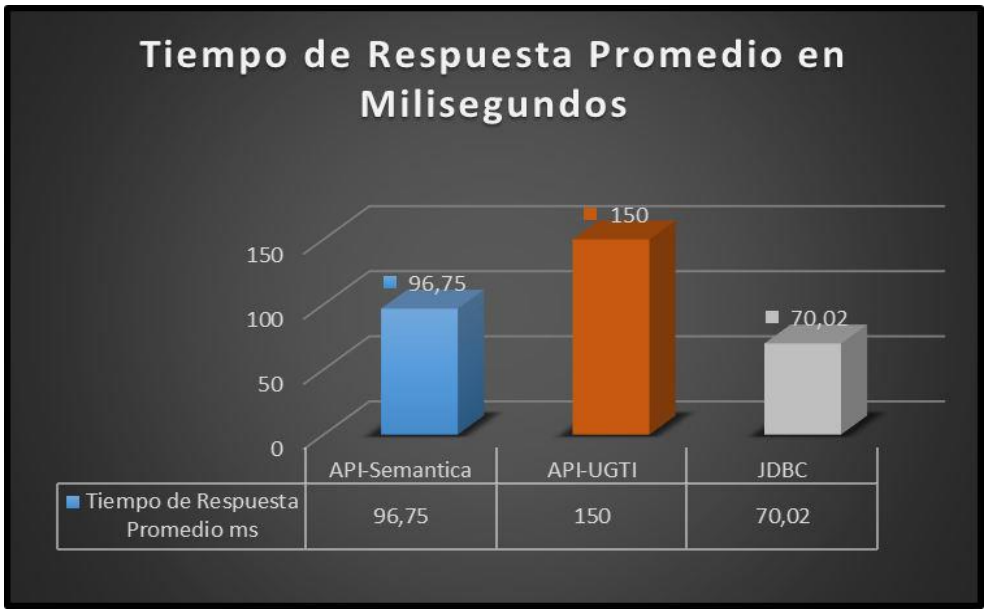


Figura 45: Comparación de tiempos de respuesta entre soluciones
Fuente: El Autor
Elaboración: El Autor

Cabe aclarar que el mejor tiempo lo tiene el método JDBC con una diferencia de 26.73 milisegundos con respecto a la solución planteada en esta investigación. Esta diferencia se debe a los procesos de trasformación y serialización que realiza el API para presentar los datos en los cuatro diferentes formatos anteriormente mencionados y probados.

Las pruebas revelan un correcto funcionamiento de cada una de las partes que conforman la solución que se plantea en el presente trabajo de investigación, demostrado así que los problemas de interoperabilidad de la UTPL pueden ser atendidos implementando soluciones de éste tipo. Centralizando, custodiando y presentando un repositorio central de datos de las principales entidades que existen en la estructura de negocio de la UTPL.

6.4. ¿Cómo la solución mejora los problemas de interoperabilidad identificados?

El presente trabajo de fin de titulación define como una alternativa viable a los problemas de interoperabilidad existentes en la UTPL la implementación de la solución propuesta. La creación de un repositorio centralizado de infracción descrita semánticamente que se convierta en el principal insumo para los desarrollos de software en la UTPL significa un gran paso hacia un gobierno de datos eficaz.

La descripción semántica de la información por medio de la utilización de vocabularios RDF permite eliminar ambigüedades propias del lenguaje natural en los datos. Y por otra parte la centralización de la información tiene como objetivo principal mantener los datos organizados, limpios y actualizados.

La sección 2.1.1 del capítulo 2 de esta investigación identifica las 3 causas principales que dan raíz a los problemas de interoperabilidad en la UTPL.

A continuación, se detalla de forma puntual cómo se eliminan los problemas de interoperabilidad identificados al aplicar la solución propuesta.

6.4.1. Carencia de modelos y diccionarios de datos actualizados de los sistemas.

Centralizar la información de las cuatro entidades seleccionadas en esta investigación implica la existencia de un modelo y diccionario de datos único. De esta forma, solo se mantiene actualizado un solo modelo y diccionario de datos.

El anexo 3 muestra el modelo físico de datos en el cual se almacenó la información extraída de los sistemas: NSGA, EVA, SIAC, SIEC y RRHH.

6.4.2. Heterogeneidad de las tecnologías de almacenamiento.

En base a lo expuesto en la sección 2.1.1.2. de esta investigación, la solución propuesta crea un repositorio de datos centralizado de las entidades: Docente, Estudiante, Autoridad y Componente Académico.

La información se almacena en forma de tripletas (Sujeto, Predicado, Objeto) sobre una plataforma para Linked data. Para la descripción en formato de tripletas se utiliza tecnologías de web semántica como: vocabularios RDF y ontologías que permiten una representación formal de un concepto además de la representación semántica y sintáctica del mismo.

El acceso a esta información en RDF es posible utilizando el lenguaje de consulta SPARQL.

Optar por tecnologías reconocidas dentro de la industria tecnológica como estándares y buenas prácticas garantiza un alto nivel de interoperabilidad entre los sistemas de información. Apache Marmotta, la plataforma sobre la cual se encuentran almacenados los datos establece dos canales de acceso a la información. El primero es un SPARQL endpoint con interfaz web desde el portal de administración y el segundo es un API RESTful de acceso al endpoint.
De esta forma la solución define una sola tecnología para cada dominio.

- Almacenamiento: Apache Marmotta kiwi triplestore
- Lenguajes de modelado de datos: RDFS y OWL
- Descripción de recursos: RDF
- Lenguaje de consulta. SPARQL
- Protocolo de acceso a datos: HTTP
- Arquitectura de Servicios: RESTful

6.4.3. Redundancia e Inconsistencia.

La sección 2.1.1.3 define los problemas de interoperabilidad que se suscitan producto de la redundancia e inconsistencia en los datos. Definir una fuente de datos centralizada logra eliminar en gran medida los problemas de redundancia e inconsistencia que puedan presentarse en los datos. Un problema crítico que no se soluciona al centralizar la información es la ambigüedad en los datos.

La ambigüedad se presenta en gran medida como producto de la mala interpretación del significado real de la información, esto como consecuencia de la utilización del lenguaje natural para la descripción de metadatos.

Contar con datos descritos de forma semántica elimina estas ambigüedades propias del lenguaje natural embebidas en los metadatos que describen la información. Al usar un vocabulario RDF se vincula un dato a una descripción semántica basada en conceptos. Los conceptos son representados por URI las cuales contiene toda la información detallada de lo que representa y relaciones hacia otros recursos

Todos los recursos descritos en RDF tienen asignado un URI que permite identificarlos de forma global y única dentro de un grafo RDF.

Una de las formas de acceder a estos recursos es mediante la ejecución de una consulta SPARQL desde la interfaz web del SPARQL endpoint de Apache Marmotta. La figura 46

muestra un ejemplo de una consulta SPARQL sobre un recurso en específico.

Para verificar, cómo al describir semánticamente un recurso permite eliminar las ambigüedades en sus datos se presenta el caso de evaluación de la sección 6.3.3.1

6.4.3.1. Caso de evaluación.

Recurso seleccionado: 1104352164

URI del Recurso: http://gestiondatos.utpl.edu.ec/marmotta/resource/1104352164

Al ingresar al URI de este recurso el cual es único, se puede observar la descripción de su información básica en formato de tripletas.

La figura 46 muestra las propiedades del recurso del caso de evaluación.

Triples

http://gestiondatos.utpl.edu.ec/marmotta/resource/1104352164

property	has value	context	info
lodutpl:laboralStatus	"Retirado"^^xsd:string	context:persona	
lodutpl:maritalStatus	"SOLTERO (A)"^^xsd:string	context:persona	
dct:identifier	"1104352164"^^xsd:string	context:persona	
schema:nationality	http://es-la.dbpedia.org/resource/Ecuatoriana	context:persona	
rdf:type	foaf:Person	context:persona	
owl:sameAs	http://gestiondatos.utpl.edu.ec/marmotta/resource/SIEC/Catalog:89871	context:persona	
owl:sameAs	http://gestiondatos.utpl.edu.ec/marmotta/resource/NSGA/Catalog:89871	context:persona	
org:org:hasMembership	http://data.utpl.edu.ec/utpl/lod/resource/Category:Docencia_Universitaria	context:persona	
org:org:memberOf	http://dbpedia.org/resource/Universidad_T%C3%A9cnica_Particular_de_Loja	context:persona	
foaf:familyName	"AGUIRRE CELI"^^xsd:string	context:persona	
foaf:mbox	"dlaguirre1@utpl.edu.ec"^^xsd:string	context:persona	
foaf:name	"DANIELA LUCÍA"^^xsd:string	context:persona	

Figura 46: Descripción del recurso seleccionado para el caso de prueba
Fuente: Recuperado de: https://goo.gl/dSIvPy
Elaboración: El Autor

Otro medio que permite ingresar a la información de este recurso es mediante la ejecución de una consulta SPAQRL desde la interfaz web del SPARQL endpoint.

La figura 47 muestra un ejemplo de una consulta SPARQL sobre este recurso seleccionado.

```
SELECT ?property ?hasValue WHERE {
  { <http://gestiondatos.utpl.edu.ec/marmotta/resource/1104352164> ?property
?hasValue }
}
ORDER BY ?property ?hasValue
```

Run

Figura 47: Consulta SPARQL del recurso
Fuente: Recuperado de: https://goo.gl/dSIvPy
Elaboración: El Autor

En figura 48 muestra la visualización del resultado producto de la consulta que la figura 47 muestra .

propiedad	Valor
http://data.utpl.edu.ec/utpl/lod/ontology/laboralStatus	Retirado ^^xsd:string
http://data.utpl.edu.ec/utpl/lod/ontology/maritalStatus	SOLTERO (A) ^^xsd:string
dct:identifier	1104352164 ^^xsd:string
http://schema.org/nationality	http://es-la.dbpedia.org/resource/Ecuatoriana
rdf:type	foaf:Person
http://www.w3.org/2002/07/owl#sameAs	http://gestiondatos.utpl.edu.ec/marmotta/resource/NSGA/Catalog:89871
http://www.w3.org/2002/07/owl#sameAs	http://gestiondatos.utpl.edu.ec/marmotta/resource/SIEC/Catalog:89871
http://www.w3.org/ns/org#org:hasMembership	http://data.utpl.edu.ec/utpl/lod/resource/Category:Docencia_Universitaria
http://www.w3.org/ns/org#org:memberOf	http://dbpedia.org/resource/Universidad_T%C3%A9cnica_Particular_de_Loja
foaf:familyName	AGUIRRE CELI ^^xsd:string

< pref Rows 1 to 10 of overall 12 next >

Figura 48: Resultados de la consulta anterior
Fuente: Recuperado de:
https://goo.gl/dSIvPy
Elaboración: El Autor

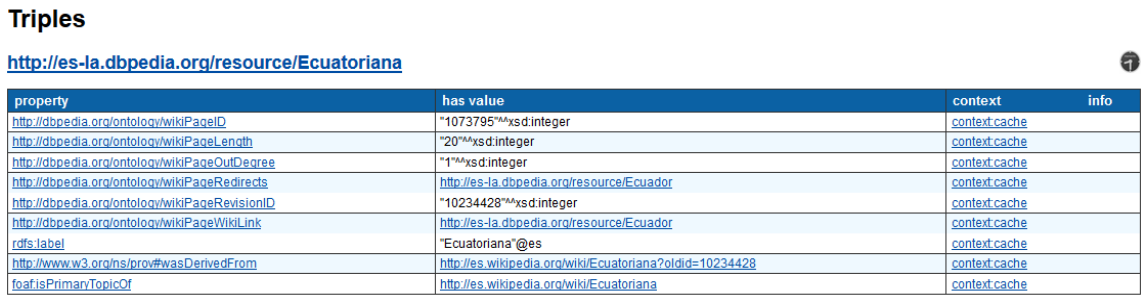
La figura 48 muestra cómo cada propiedad es un concepto descrito como una URI.

Una de las propiedades que describe al recurso anterior es nacionalidad, en el caso particular del recurso seleccionado el valor de esta propiedad es un URI que describe el concepto “Ecuatoriana”, que representa una nacionalidad de las personas nacidas en Ecuador.

Normalmente la nacionalidad de una persona dentro de los sistemas se encuentra represen-

tada en forma de un literal. En el caso puntual de este recurso, la representación de esta propiedad en los sistemas SIEC y NSGA es: “ECUATORIANA”. Esa representación conlleva a una serie de interpretaciones que dependen del conocimiento del modelo de datos de cada sistema. Relacionar la propiedad nacionalidad con el URI que describe el concepto de “Ecuatoriana” como la nacionalidad de las personas nacidas en Ecuador elimina la posibilidad de otras interpretaciones de este dato.

La figura 49 muestra la descripción de “Ecuatoriana” como nacionalidad en la dbpedia.



Triples

http://es-la.dbpedia.org/resource/Ecuatoriana

property	has value	context	info
http://dbpedia.org/ontology/wikiPageID	"1073795"^^xsd:integer	context:cache	
http://dbpedia.org/ontology/wikiPageLength	"20"^^xsd:integer	context:cache	
http://dbpedia.org/ontology/wikiPageOutDegree	"1"^^xsd:integer	context:cache	
http://dbpedia.org/ontology/wikiPageRedirects	http://es-la.dbpedia.org/resource/Ecuador	context:cache	
http://dbpedia.org/ontology/wikiPageRevisionID	"10234428"^^xsd:integer	context:cache	
http://dbpedia.org/ontology/wikiPageWikiLink	http://es-la.dbpedia.org/resource/Ecuador	context:cache	
rdfs:label	"Ecuatoriana"@es	context:cache	
http://www.w3.org/ns/prov#wasDerivedFrom	http://es.wikipedia.org/wiki/Ecuatoriana?oldid=10234428	context:cache	
foaf:isPrimaryTopicOf	http://es.wikipedia.org/wiki/Ecuatoriana	context:cache	

Figura 49: Descripción del recurso nacionalidad ecuatoriana
Fuente: Recuperado de: https://goo.gl/h80bz1
Elaboración: El Autor

Esta persona presenta registros en los sistemas SIEC y NSGA. Cada uno de estos sistemas establece una forma particular de identificar a esta persona dentro de su base de datos. Intentar relacionar esta persona con sus equivalencias en los otros sistemas implica utilizar tablas de equivalencia. Uno de los inconvenientes de usar una tabla de equivalencia como medio de relación es la necesidad de conocer el modelo de datos del o los sistemas con los que se desea establecer la relación.

La web semántica presenta una solución puntual a este problema, definiendo así una propiedad que permite describir las diferentes formas de llamarse de un recuso en varios sistemas o fuentes de información. La propiedad owl:SameAs relaciona un recurso con sus otras instancias en diferentes sistemas estableciendo así un medio directo de relación sin la necesidad de conocer los modelos de datos de los otros sistemas.

La figura 50 muestra las otras instancias del recurso seleccionado para el caso de evaluación en los sistemas NSGA y SIEC.

Triples

http://gestiondatos.utpl.edu.ec/marmotta/resource/1104352164

property	has value	context
lodutpl:laboralStatus	"Retirado"^^xsd:string	context:persona
lodutpl:maritalStatus	"SOLTERO (A)"^^xsd:string	context:persona
dct:identifier	"1104352164"^^xsd:string	context:persona
schema:nationality	http://es-la.dbpedia.org/resource/Ecuatoriana	context:persona
rdf:type	foaf:Person	context:persona
owl:sameAs	http://gestiondatos.utpl.edu.ec/marmotta/resource/SIEC/Catalog:89871	context:persona
owl:sameAs	http://gestiondatos.utpl.edu.ec/marmotta/resource/NSGA/Catalog:89871	context:persona
org:org:hasMembership	http://data.utpl.edu.ec/utpl/lod/resource/Category:Docencia_Universitaria	context:persona
org:org:memberOf	http://dbpedia.org/resource/Universidad_T%C3%A9cnica_Particular_de_Loja	context:persona
foaf:familyName	"AGUIRRE CELI"^^xsd:string	context:persona
foaf:mbox	"dlaguirre1@utpl.edu.ec"^^xsd:string	context:persona
foaf:name	"DANIELA LUCÍA"^^xsd:string	context:persona

Figura 50: Propiedad SemeAs del recurso seleccionado para la evaluación
Fuente: Recuperado de: https://goo.gl/dSIvPy
Elaboración: El Autor

La figura 51 muestra cómo al acceder a una de estas URIs se obtiene una descripción del recurso. La descripción del recurso incluye la fuente o sistema que identifica de esta forma en particular a esta persona y las relaciones hacia los otros sistemas.

Triples

http://gestiondatos.utpl.edu.ec/marmotta/resource/NSGA/Catalog:89871

property	has value	context	info
lodutpl:Source	http://gestiondatos.utpl.edu.ec/marmotta/resource/Sistema/NSGA	context:persona	
dct:description	"Forma de identificar este recurso en el Sistema de Información"^^xsd:string	context:persona	
rdfs:label	"89871"^^xsd:string	context:persona	
owl:sameAs	http://gestiondatos.utpl.edu.ec/marmotta/resource/SIEC/Catalog:89871	context:persona	
owl:sameAs	local:1104352164	context:persona	

Figura 51: Propiedad SemeAs en el NSGA
Fuente: Recuperado de:
https://goo.gl/dSIvPy
Elaboración: El Autor

De esta forma cualquier sistema que desee acceder a la información de este recurso lo puede hacer definiendo su identificador al final del URI del catálogo por sistema y automáticamente establecer una relación entre su instancia y la de la solución.

De igual manera el sistema en el cual existe la instancia de esta persona se encuentra descrito como un recurso.

La figura 52 muestra una breve descripción del sistema NSGA.

Triples

http://gestiondatos.utpl.edu.ec/marmotta/resource/Sistema/NSGA

property	has value	context	info
dct:description	"El nuevo sistema para la gestión académico de la UTPL en vigencia desde el 2010."^^xsd:string	context:sistema	
rdf:type	http://purl.org/dc/dcmitype/Software	context:sistema	
rdfs:label	"NSGA"^^xsd:string	context:sistema	
foaf:name	"Nuevo Sistema de Gestión Académica"^^xsd:string	context:sistema	

Figura 52: Descripción del recurso NSGA
Fuente: Recuperado de: https://goo.gl/dSIvPy
Elaboración: El Autor

Las figuras 53 y 54 fueron generadas con las ayuda del RDF validator que la W3C provee en su sitio web. Para esto se ingreso la los datos en formato RDF de un recurso para obtener la representación gráfica de las tripletas.

Como parte final de la demostración de este caso de evaluación la figura 53 muestra una vista del grafo RDF perteneciente al recurso seleccionado este grafo esta reducido es decir no se profundiza en sus demás nodos o recursos.

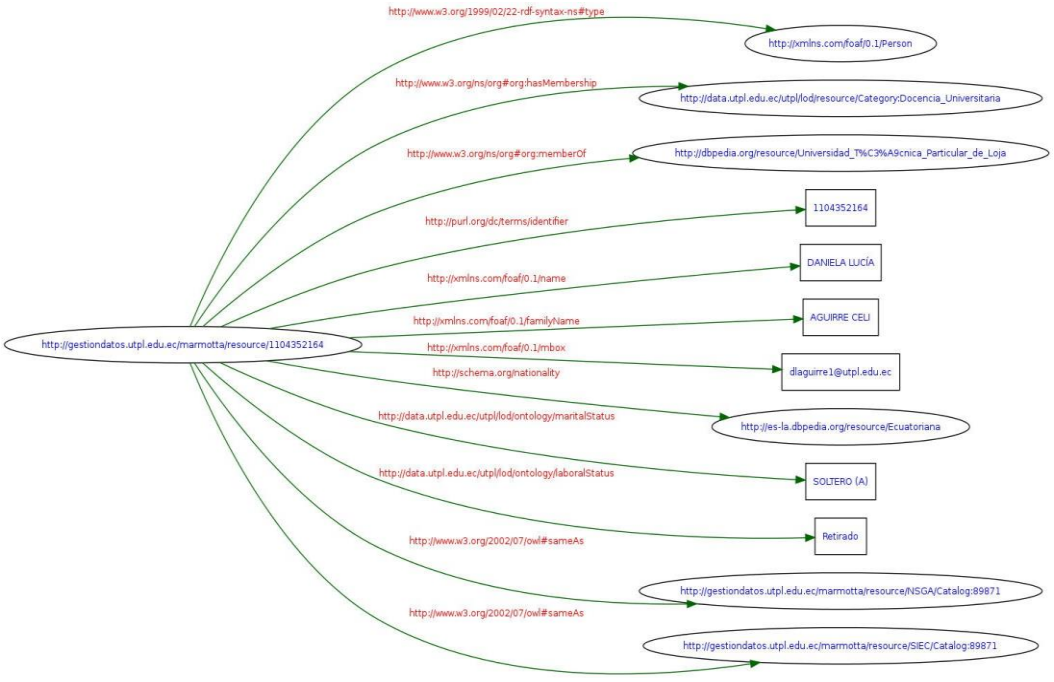


Figura 53: Grafo RDF del recurso seleccionada en el caso de evaluación
Fuente: El Autor
Elaboración: El Autor

La figura 54 muestra una proyección más profunda del grafo RDF del recurso seleccionad para este caso de evaluación y se observa como se conecta con otros recursos:

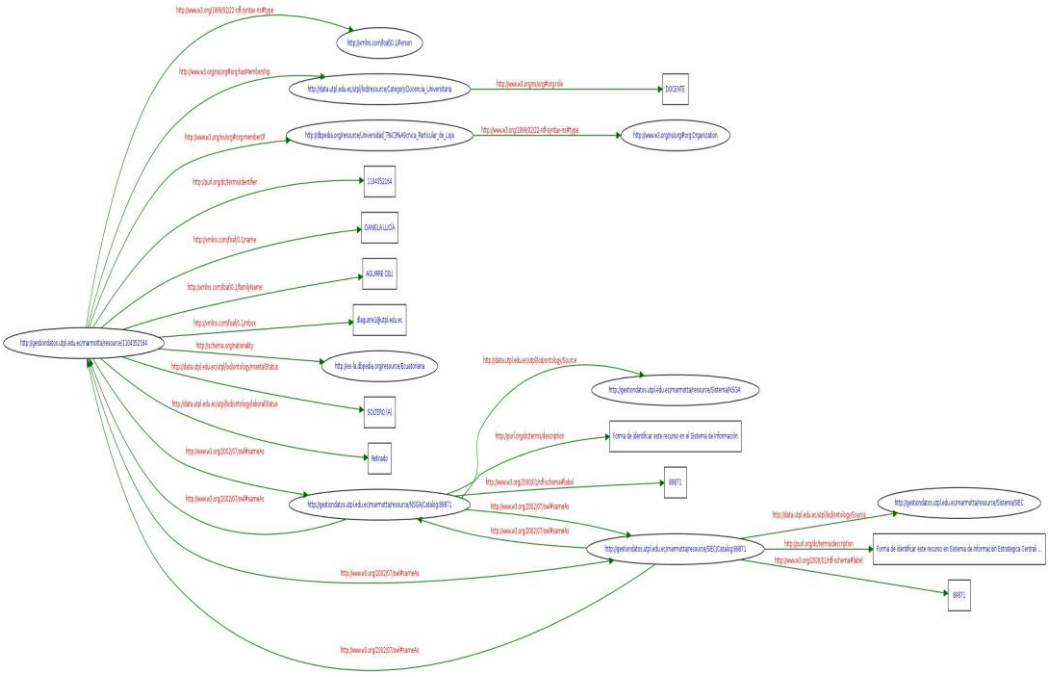
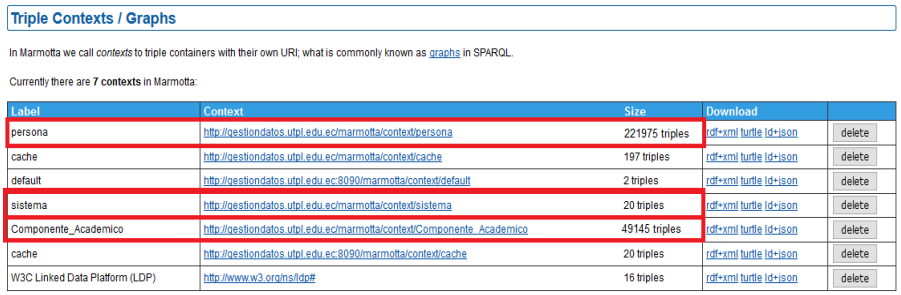


Figura 54: Grafo RDF expandido del recurso seleccionada en el caso de evaluación
Fuente: El Autor
Elaboración: El Autor

Almacenar la información en una estructura como un grafo RDF permite descartar los registros con valores nulos. Comúnmente en los sistemas tradicionales al no contar con la información para poblar un atributo en la tabla, obliga a crear un registro en nulo. En una estructura de grafo cuando no existe el valor para una propiedad en la ontología el nodo de esa propiedad no se crea para ese recurso y de esa forma se crea una estructura más dinámica.

Finalmente se evidencia que la aplicación de tecnologías de Web Semántica para la descripción de las entidades de datos permite alcanzar el objetivo principal de este trabajo de investigación.

La figura 55 muestra detalladamente el número de entidades y los contextos en los cuales éstas se encuentran almacenadas.



Triple Contexts / Graphs

In Marmotta we call *contexts* to triple containers with their own URI; what is commonly known as graphs in SPARQL.

Currently there are **7 contexts** in Marmotta:

Label	Context	Size	Download	
persona	http://gestiondatos.utpl.edu.ec/marmotta/context/persona	221975 triples	rdf+xml turtle ld+json	delete
cache	http://gestiondatos.utpl.edu.ec/marmotta/context/cache	197 triples	rdf+xml turtle ld+json	delete
default	http://gestiondatos.utpl.edu.ec:8090/marmotta/context/default	2 triples	rdf+xml turtle ld+json	delete
sistema	http://gestiondatos.utpl.edu.ec/marmotta/context/sistema	20 triples	rdf+xml turtle ld+json	delete
Componente_Academico	http://gestiondatos.utpl.edu.ec/marmotta/context/Componente_Academico	49145 triples	rdf+xml turtle ld+json	delete
cache	http://gestiondatos.utpl.edu.ec:8090/marmotta/context/cache	20 triples	rdf+xml turtle ld+json	delete
W3C Linked Data Platform (LDP)	http://www.w3.org/ns/ldp#	16 triples	rdf+xml turtle ld+json	delete

Figura 55: Lista de contextos en donde se alojan los datos
Fuente: Recuperado de: https://goo.gl/dSIvPy
Elaboración: El Autor

Apache Marmotta maneja el concepto de contextos para referirse a los Grafos en los cuales se almacenan las tripletas y cómo se puede observar existen tres contextos cuya informacion es detallada en la tabla 34.

Tabla 34: Lista de Contextos y su relación con las entidades.

Contexto	Número de Tripletas	Entidades que contiene el contexto	Número de entidades
componente academico	49145	Componente académico	5.773
persona	221975	Docente y Autoridad	6.411
		Estudiante	53.991
Sistema	20	Sistema	5

Fuente: El Autor
Elaboración: El Autor

Debido al gran volumen de datos y a los altos niveles de tráfico de red que representa el centralizar la información, la solución se encuentra desplegada sobre 3 servidores con la finalidad de garantizar la alta disponibilidad de la información. Adicionalmente a eso se han realizado configuraciones para el balanceo de carga y de esta forma soportar los grandes niveles de tráfico que puedan presentarse en un futuro o en fechas picos como lo son periodo de matrículas y fin de semestre.

6.5. Comentarios Finales

Al culminar este capítulo se puede apreciar de que forma el caso de prueba seleccionado muestra como la solución resuelve las tres principales causas al problema de interoperabilidad identificado. De igual forma se observa como el rendimiento en tiempo de respuesta de la solución es favorable y superior a las alternativas tradicionales.

DISCUSIÓN FINAL

Como respuesta a la problemática planteada en el capítulo 2, se define una solución enmarcada en el diseño e implementación de una arquitectura distribuida en tres niveles. La arquitectura propuesta en su nivel medio (Nivel de Aplicación) implementa patrones de diseño como: MVC e inyección de dependencias. El desarrollo se guía en el marco de trabajo de Spring MVC y se utiliza Java como el lenguaje de programación.

La presente investigación además del diseño de una arquitectura que pueda hacer frente a los problemas de interoperabilidad propone la implementación de la solución sobre un prototipo. La implementación está enmarcada en el desarrollo de un RESTful API que permite acceder a la información básica de: Docentes, Estudiantes, Autoridades de la UTPL y de Componentes Académicos descrita semánticamente.

Esta aproximación de un modelo de interoperabilidad de datos que propone el presente trabajo de investigación se encuentra orientado a los desarrolladores de software que pertenecen al departamento de tecnología de la UTPL. Orientar esta solución a un número limitado de desarrolladores radica en la sensibilidad de la información personal que se maneja.
El caso de aplicación sobre el cual esta investigación implementa la solución planteada identifica a las entidades: Docente, Estudiante, Autoridades y Componente académico como las más redundantes dentro de los sistemas: NSGA, UTE, Distributivo, EVA, SIAC, FileMaker, Buxis y NSFA.

Al analizar estas cuatro entidades de datos en cada uno de los sistemas mencionados anteriormente se identifican problemas como:

- Múltiples Instancias de la Entidad Docente, Estudiante y Autoridad en los Sistemas: Nómina, NSGA, Distributivo, EVA, SIAC y UTE.
- Múltiples Instancias de la Entidad Componente Académico en los Sistemas: Distributivo, EVA y NSGA.
- Información incompleta dentro del NSGA en donde están migrados los programas académicos anteriores al 2008, notas y estados de graduación.
- Distintos nombres de las columnas en las tablas de cada sistema para referirse al mismo dato. Ejemplo: para identificar el grado académico de una persona se lo expresa de tres maneras diferentes (Nivel Académico, Grado Académico y Nivel de Formación Académica).
- Limitada y compleja capacidad de enriquecimiento del modelo relacional de datos.
- Asignación de distintos identificadores únicos de cada entidad en los sistemas.

- Utilización de tablas de equivalencia para relacionar una misma entidad entre el sistema A y el sistema B.
- Heterogeneidad de tecnologías de almacenamiento y transferencia de datos, es el caso puntual del EVA y el NSGA, en donde el primer sistema posee una base de datos MySQL y el segundo una base de datos Oracle.
- Dispersión de las fuentes de información en donde el EVA se encuentra tercerizado a la empresa level3 y la conexión a la base de datos se hace mediante un servicio Cloud. El sistema académico mantiene su propia base de datos. Recursos humanos se encuentra migrando su información a BUXIS un sistema propietario y del cual el proveedor no facilita el modelo relacional de los datos para explotarlos.
- Desarrollos aislados que promueven la creación o formación de silos de información.

Una vez identificadas estas particularidades que dan lugar a los problemas de interoperabilidad existentes dentro de la UTPL la solución expuesta en esta investigación plantea lo siguiente:

- La centralización de la información de cada una de las entidades.
- Modelar de forma semántica los datos de cada entidad empleando un vocabulario RDF.
- Estandarizar la tecnología de almacenamiento de datos y el lenguaje de consulta.
- Establecer dos canales únicos de acceso a los datos centralizados.

Centralizar la información de estas entidades en una sola plataforma facilita el acceso a los datos y de esta forma se elimina las instancias redundantes de esta entidad en los demás sistemas.

Los desarrolladores a quienes esta solución va orientada pueden acceder a los datos centralizados y descritos semánticamente de dos formas distintas. La primera es efectuando una serie de consultas sobre el SPARQL endpoint de Apache Marmotta y la segunda invocando uno de los servicios del RESTful API. El request que se realice a un servicio debe indicar por cabecera el formato de serialización en el que se desea que los datos sean presentados.

Describir un recurso semánticamente y asignar a este un identificador global y único que sea inmutable en el tiempo es posible utilizando lenguajes de modelado semántico de datos como: RDFS y OWL.

Utilizar tecnologías de web semántica permite definir vínculos entre un recurso y sus instancias en los otros sistemas evitando así utilizar tablas de equivalencia para relacionar el mismo recurso entre los diferentes sistemas.

Una de las muchas ventajas que representa describir semánticamente la información radica en que las propiedades que describen un recurso son definidas como conceptos y descritas como recursos. Cada instancia de esas propiedades puede ser otro recurso o un valor literal que es descrito como un concepto, en donde se define dominios y rangos de esa instancia.

Es claro que la adopción de tecnologías de web semántica para representar los datos institucionales de la UTPL se convierte en un punto clave para establecer una solución definitiva a los problemas de interoperabilidad identificados previamente.

Los resultados obtenidos producto de la fase de pruebas permiten verificar y sustentar la viabilidad de una solución de este tipo frente a los problemas de interoperabilidad que se tratan en el capítulo 2 del presente trabajo de investigación.

CONCLUSIONES

Al terminar el presente trabajo de investigación se ha podido concluir lo siguiente:

- Realizar la identificación y una clara descripción de las situaciones que originan los problemas de interoperabilidad entre las distintas fuentes de datos que posee la Universidad Técnica Particular de Loja, se convierte en una tarea fundamental puesto que permita tener una idea más clara y detallada de los problemas que se deben solucionar. Al contar con toda esta información claramente detallada se puede iniciar actividades de investigación que permitan encontrar y definir tecnologías útiles que puedan dar solución a estos problemas de forma definitiva.

- El realizar la definición de una arquitectura robusta y escalable sobre la cual se pueda desplegar e implementar un prototipo de la solución a los problemas puntuales de interoperabilidad de las fuentes de datos instituciones de la UTPL, garantiza la alta disponibilidad de la información y las herramientas de software.

- La centralización de la información de las entidades comunes en un solo repositorio de datos y bajo una sola tecnología de almacenamiento permite definir un almacén de datos actualizado. Al optar por la centralización se elimina la posibilidad de inconsistencias en los datos, producto de tener una instancia aislada de cada entidad en los sistemas. De esta forma también se permite establecer parámetros y procedimientos de control sobre un único almacén de datos que permita garantizar la integridad de la información que los sistemas utilizan y presentan a los usuarios finales.

- La integración de información proveniente de fuentes de datos heterogéneas y distribuidas aproxima a la UTPL a un nivel de interoperabilidad entre datos y aplicaciones. Permita contar con información actualizada y libre de inconsistencias. Lograr esto representa una de las tareas más arduas puesto que se debe comprender el modelo de datos de donde se extrae la información y combinar varias tecnologías para lograrlo.

- Identificar de forma clara los principales problemas de interoperabilidad de los datos permitieron definir a las tecnologías de web semántica como las herramientas para poder solucionar estos problemas. El modelar la información existente siguiendo el ciclo de publicación de datos enlazados permite identificar y definir uno o varios vocabularios RDF. De igual forma se selecciona los conceptos a los cuales irán ligados cada uno de los datos.

- Los vocabularios que se seleccionaron permiten establecer un concepto sobre el cual se crea un dato y de esta forma se puede eliminar las ambigüedades propias del lenguaje natural y así descartar la posibilidad de una mala interpretación de los datos por parte de quienes los utilicen.

RECOMENDACIONES

Durante el proceso de elaboración del presente trabajo de investigación se ha recopilado las siguientes recomendaciones:

- Al identificar las principales causas que dan cabida a los problemas de interoperabilidad se puede evidenciar la carencia de políticas de control y gestión de datos que permitan determinar un marco de referencia para el manejo adecuado de la información, por lo tanto, se deben establecer políticas que regulen la creación, actualización, eliminación y acceso a los datos.
- Desplegar una solución de software que soporte una gran cantidad de solicitudes de datos debe realizarse sobre una infraestructura dedicada de hardware de altas prestaciones con el objetivo de garantizar la alta disponibilidad de los datos hacia las aplicaciones.
- Iniciativas como la centralización o integración de la información provenientes de fuentes de datos heterogéneas y dispersas precisan la utilización de Modelos y diccionarios de datos actualizados. Por esta razón se debe enfatizar en la creación de estos modelos y diccionarios de datos de todos los sistemas que posee la UTPL y de igual forma definir la frecuencia con la que estos deben ser actualizados.
- Intentar extraer información de fuentes de datos demasiado privativas como es el caso de los sistemas BUXIS y FileMaker se convierte una verdadera hazaña puesto que el proveedor del primer sistema no proporciona un modelo y diccionario de datos y en el caso del segundo sistema se debe utilizar varias técnicas desarrolladas por la comunidad para logar un puente de conexión y poder extraer la información. Debido a esto se recomienda antes de adquirir algún producto de software solicitar un modelo y diccionario de datos y procurar seleccionar sistemas que utilicen tecnologías estándar.
- El modelado semántico de datos en tripletas semánticas requiere el uso de una serie de vocabularios RDF para la asociación de recursos. Por lo cual se debe procurar la reutilización de vocabularios ya existentes y disponibles en la web. En los casos puntuales donde sea necesario la creación de un vocabulario propio se debe seguir los estándares internacionales establecidos por la W3C para la creación y publicación de vocabularios. Estas medidas se enfatizan con la finalidad de garantizar la interoperabilidad de recursos y aplicaciones.
- El diseño de la ontología puede convertirse en un trabajo laborioso por lo que se recomienda la utilización de herramientas diseñadas para estas tareas en específico como lo es el caso de Protégé que un software de código abierto que permite la creación y edición de ontologías. El emplear este tipo de software agiliza los procesos de creación ontologías y de desarrollo puesto que también se puede generar todo el código necesario para la transformación de la data a RDF.

BIBLIOGRAFÍA

ARENS, Y., CHEE, C. Y., HSU, C.-N., & KNOBLOCK, C. A. (2003). RETRIEVING AND INTEGRATING DATA FROM MULTIPLE INFORMATION SOURCES. *International Journal of Cooperative Information Systems*, *02*(02), 127–158.

Aris M. Ouksel, A. P. S. (2010). Semantic Interoperability in Global Information Systems: A Brief Introduction to the Research Area and the Special Section. *SIGMOD Record*, *28*, 5–12.

Bass, L., Clements, P., & Kazman, R. (2013). Software architecture in practice. 2nd addison-wesley. *Reading, MA*.

Booch, G. (1994). *Object-oriented Analysis and Design with Applications (2Nd Ed.)*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

Bukhres, O. & Elmagarmid, A. (1996). *Object-oriented multidatabase systems: a solution for advanced applications*. Prentice Hall.

Carey, M. J., Haas, L. M., Schwarz, P. M., Arya, M., Cody, W. F., Fagin, R., Flickner, M., Luniewski, A. W., Niblack, W., Petkovic, D., Thomas, J., Williams, J. H., & Wimmers, E. L. (1995). Towards heterogeneous multimedia information systems: the garlic approach. In *Research Issues in Data Engineering, 1995: Distributed Object Management, Proceedings. RIDE-DOM '95. Fifth International Workshop on*, (pp. 124–131).

Franklin, M., Halevy, A., & Maier, D. (2005). From databases to dataspaces: A new abstraction for information management. *SIGMOD Rec.*, *34*(4), 27–33.

Hilliard, R. (1991). Ieee standard computer dictionary: A compilation of ieee standard computer glossaries. *IEEE Std 610*, 1–217.

Hilliard, R. (2000). 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. *IEEE Software*.

Hogan, A., Harth, A., Umbrich, J., Kinsella, S., Polleres, A., & Decker, S. (2014). Searching and browsing linked data with swse: The semantic web search engine. *Web semantics: science, services and agents on the world wide web*, *9*(4), 365–401.

Hurson, A. & Bright, M. (1991). Multidatabase Systems: An Advanced Concept in Handling Distributed Data. *Advances in Computers*, *32*, 149–200.

Jacquart, R. (2015). *Building the Information Society: IFIP 18th World Computer Congress Topical Sessions 22–27 August 2004 Toulouse, France*. IFIP Advances in Information and Communication Technology. Springer US.

Janssen, M., Estevez, E., & Janowski, T. (2014). Interoperability in big, open, and linked data–organizational maturity, capabilities, and data portfolios. *Computer*, *47*(10), 44–49.

Khalid, N., Pasha, M., Rehman, S. U., Ahmad, H. F., & Suguri, H. (2014). Ontology services between agents and owl based web services. In *Semantics, Knowledge and Grid, Third International Conference on*, (pp. 176–181).

Klein, M. H., Kazman, R., Bass, L., Carriere, J., Barbacci, M., & Lipson, H. (1999). Attribute-based architecture styles. In *Software Architecture* (pp. 225–243). Springer.

Kruchten, P. (1995). Architectural Blueprints The ”4+1 ”View Model of Software Architecture. *IEEE Software*, *12*(6), 42–50.

Mena, E., Illarramendi, A., Kashyap, V., & Sheth, A. P. (2013). OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies. *Distributed and Parallel Databases*, *8*(2), 223–271.

Navathe, S., Gala, S., & Sheth, A. (2000). On automatic reasoning for schema integration. *ACM Comput. Surv.*, *43*(3), 183–236.

Piedra, N., Tovar, E., Colomo-Palacios, R., Lopez-Vargas, J., & Alexandra Chicaiza, J. (2014). Consuming and producing linked open data: the case of opencourseware. *Program*, *48*(1), 16–40.

Rubin, K. S. & Goldberg, A. (1992). Object behavior analysis. *Communications of the ACM*, *35*(9), 48–62.

Sheth, A. P. & Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, *22*(3), 183–236.

Villazón-Terrazas, B., Vilches-Blázquez, L. M., Corcho, O., & Gómez-Pérez, A. (2014). Methodological guidelines for publishing government linked data. In *Linking government data* (pp. 27–49). Springer.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, (pp. 108–117). Citeseer.

Ziegler, P. & Dittrich, K. R. (2010). Data Integration — Problems, Approaches, and Perspectives. Recuperado el 25 de junio del 2015, de https://pdfs.semanticscholar.org/ac7c/ed257ae5598d4a6048ea9c182773d317126c.pdf.

GLOSARIO DE TERMINOS

En esta sección encontramos un glosario de términos con algunas de las palabras utilizadas en este documento, con la finalidad de facilitar al lector un entendimiento más completo de algunos conceptos.

A

API: Application Programming Interface, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ASC X12: Comité de Normas acreditados X12 (también conocido como ASC X12) es una organización de estándares . Patrociando por el American National Standards Institute (ANSI) en 1979, que se desarrolla y mantiene el X12 intercambio electrónico de datos (EDI) y Contexto Component Architecture Inspirado estándares (CICA), junto con los esquemas XML que impulsan los procesos de negocio a nivel mundial.

B

BUXIS: Empresa especializada en el desarrollo de software para la gestión del Capital Humano dentro de las organizaciones.

BI: Se denomina inteligencia empresarial, inteligencia de negocios o BI al conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos existentes en una organización o empresa.

C

CDI: Contextos e inyección de dependencias - también conocido como Web Beans - es un estándar Java.,

CRAWLER: Es como se llaman en inglés a las arañas web de los buscadores que rastrean las páginas web en busca de información para un correcto posicionamiento de las webs en sus resultados.

CRUD: Es el acrónimo de Çrear, Leer, Actualizar y Borrar”(del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

D

DBMS: Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

DISTRIBUTIVO: Nombre que recibe la hoja electrónica de Excel que almacena la información de todo lo que ocurrió en la UTPL en el ámbito académico.

E

EVA: Entorno Virtual de Aprendisaje el mismo que provine de moddle.

F

FILEMAKER: Es una aplicación multiplataforma de base de datos relacional de FileMaker Inc.

H

HARDWARE: Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

HTTP: Abreviatura de la forma inglesa Hypertext Transfer Protocol, 'protocolo de transferencia de hipertextos', que se utiliza en algunas direcciones de internet.

I

IEEE: Es una organización sin ánimo de lucro, la mayor asociación del mundo para el desarrollo tecnológico. Su nombre completo es el Instituto de Ingenieros Eléctricos y Electrónicos, aunque normalmente se le conoce con las letras I-E-E-E, pronunciadas como Ï-E-cubo".

J

JDBC: Es el API para la ejecución de sentencias SQL. (Como punto de interés JDBC es una marca registrada y no un acrónimo, no obstante, a menudo es conocido como "Java Database Connectivity"). Consiste en un conjunto de clases e interfases escritas en el lenguaje de programación Java.

JSON: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

JSONLD: JavaScript Object Notation for Linked Data, es un método de codificación de da-

tos enlazados usando JSON.

L

LAN: Son las siglas de Local Area Network, Red de área local. Una LAN es una red que conecta los ordenadores en un área relativamente pequeña y predeterminada (como una habitación, un edificio, o un conjunto de edificios).

LD: Son las siglas para referirse a Linked Data.

LDP: Linked Data Platform es una especificación de Linked data definiendo una serie de patrones de integración para la construcción de servicios web RESTful HTTP que sean capaces de leer y escribir datos en RDF.

LINUX: Es un sistema operativo de software libre (no es propiedad de ninguna persona o empresa), por ende, no es necesario comprar una licencia para instalarlo y utilizarlo en un equipo informático.

M

MIDDLEWERE: Es un software de computadora que conecta componentes de software o aplicaciones para que puedan intercambiar datos entre éstas. Es utilizado a menudo para soportar aplicaciones distribuidas.

MTOM: (SOAP Message Transmission Optimization Mechanism) es un estándar de Web Services que permite transferir datos binarios de una forma más eficiente a los SOAP Attachments.

MVC: El modelo–vista–controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

My SQL: Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo.

N

NSFA: Son las siglas del Nuevo sistema Financiero Académico que utiliza la Universidad Técnica Particular de Loja.

NSGA: Son las siglas del Nuevo sistema de Gestión Académica que utiliza la Universidad Técnica Particular de Loja.

O

ODBC: Open DataBase Connectivity, es un estándar de acceso a las bases de datos desarrollado por SQL Access Group (SAG) en 1992. El objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos.

ORACLE: Es básicamente una herramienta cliente/servidor para la gestión de base de datos, es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que solo se vea en empresas muy grandes y multinacionales, por norma general.

OWL: Es el acrónimo del inglés Ontology Web Language, un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW. OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML.

P

P2P: Una red peer-to-peer, red de pares, red entre iguales o red entre pares (P2P, por sus siglas en inglés) es una red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí.

PL/SQL: (Procedural Language/Structured Query Language) es un lenguaje de programación incrustado en Oracle.

POSGRES: Es un Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia PostgreSQL, similar a la BSD o la MIT.

PROTOCOLO: Es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos entre dos puntos finales.

R

RDF: The Resource Description Framework , es un lenguaje para referenciar la información de los recursos de la World Wide Web.

RDFS: RDF Schema o Esquema RDF es una extensión semántica de RDF. Un lenguaje primitivo de ontologías que proporciona los elementos básicos para la descripción de vocabularios.

REST: REpresentational State Transfer", que traducido vendría a ser "transferencia de representación de estado". Se refería originalmente a un conjunto de principios de arquitectura. En la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre

los datos.

RESTful: Denominación que reciben los servicios web que siguen los principios REST.

RRHH: Los recursos humanos de una empresa.

S

SCRIPT: Archivo de órdenes, archivo de procesamiento por lotes o, cada vez más aceptado en círculos profesionales y académicos, guion es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

SESAME: Es un framework open source para consultar y analizar RDF.

SIAC: Son las siglas del Sistema de Información Científica Académica que utiliza la Universidad Técnica Particular de Loja.

SIEC: Son las siglas del Sistema de Información Estratégica Centralizada que utiliza la Universidad Técnica Particular de Loja.

SOA: La Arquitectura Orientada a Servicios (en inglés Service Oriented Architecture), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

SOAP: Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC.

SOFTWARE: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

SPARQL: Es un acrónimo recursivo del inglés SPARQL Protocol and RDF Query Language. Se trata de un lenguaje estandarizado para la consulta de grafos RDF, normalizado por el RDF Data Access Working Group (DAWG) del World Wide Web Consortium (W3C).

SQL: Por sus siglas en inglés Structured Query Language; en español lenguaje de consulta estructurada, es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos.

SQL SERVER: Es un sistema de gestión de bases de datos relacionales (RDBMS) de Microsoft que está diseñado para el entorno empresarial.

T

TRIPLESTORE: Es una base de datos construida a medida para el almacenamiento y recuperación de triples a través de consultas semánticas. Un triple es una entidad de datos

compuesta de sujeto- predicado -objeto.

TXT: La extensión TXT representa ”textfile”(archivo de texto), que sustituyó a su antiguo nombre ”flatfile”(archivo sin formato). Este archivo informático estructura series de líneas de texto.

U

UGDA: Son las siglas de la unidad de gestión de datos académicos centralizados que posee la Universidad Técnica Particular de Loja.

URI: Son las siglas en inglés de Uniform Resource Identifier (en español identificador uniforme de recursos), que sirve para identificar recursos en Internet, precisamente lo que el nombre indica.

URL: Es una secuencia de caracteres que se utiliza para nombrar y localizar recursos, documentos e imágenes en Internet. URL significa Üniform Resource Locator”, o bien, ”Localizador Uniforme de Recursos”.

UTE: Son las siglas del sistema de Unidad de Titulación Especial que posee la Universidad Técnica Particular de Loja.

UTPL: Son las siglas de la Universidad Técnica Particular de Loja.

W

W3C: Son las siglas de World Wide Web Consortium, un consorcio fundado en 1994 para dirigir a la Web hacia su pleno potencial mediante el desarrollo de protocolos comunes que promuevan su evolución y aseguren su interoperabilidad.

WAN: Es la sigla de Wide Area Network (“Red de Área Amplia”). El concepto se utiliza para nombrar a la red de computadoras que se extiende en una gran franja de territorio, ya sea a través de una ciudad, un país o, incluso, a nivel mundial.

WEB SCRAPPY: Es una técnica utilizada mediante programas de software para extraer información de sitios web. Usualmente, estos programas simulan la navegación de un humano en la World Wide Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación.

WS: Son las siglas de Web Services o en español Servicio Web.

WS-ADDRESSING: WS-Addressing proporciona mecanismos de transporte para direccionar servicios web y mensajes. En concreto, esta especificación define XML.

WS-CDL: Lenguaje para la descripción de Coreografías de Servicios Web (Web Services Choreography Description Language (WS-CDL) es un lenguaje basado en XML que describe la colaboración entre pares peer-to-peer, mediante la definición - desde un punto de vista

global - de los comportamientos comunes y observables de cada participante de un proceso de negocio.

WSDL: Las siglas de Web Services Description Language, es un formato del Extensible Mar-kup Language (XML) que se utiliza para describir servicios web (WS).

X

XML: Es un subconjunto de SGML(Estándar Generalised Mark-up Language),simplificado y adaptado a Internet.

XML-RPC: Es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

Y

YAML: Es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl, así como el formato para correos electrónicos especificado por el RFC 2822.

ANEXOS

ANEXO 1. Instalación y Configuración de Apache Marmotta en un Ambiente de Producción.

Figura 56: Logotipo Apache marmotta
Fuente: Apache Marmotta Home Page
Elaboración: Apache Marmotta Team

Apache Marmotta una recomendación de la WC3 desde el 2013, es una plataforma abierta para Linked Data la misma que está construida sobre una arquitectura SOA utilizando CDI/-Weld service framework la cual la convierte en la mejor alternativa al momento de hablar de interoperabilidad semántica. En la figura 55 se puede apreciar su logotipo con su mascota oficial.

Apache Marmota LDP se encuentra desplegada en un servidor de aplicaciones Tomcat 8.0.32 y utiliza el sub dominio público de la Unidad De Gestión de Datos Académicos (UDGA) de la UTPL y puede ser accedida desde la siguiente URL: http://gestiondatos.utpl.edu.ec/marmotta/

La figura 56 muestra la pagina principal de la instancia desplegada en los servidores de la UTPL.

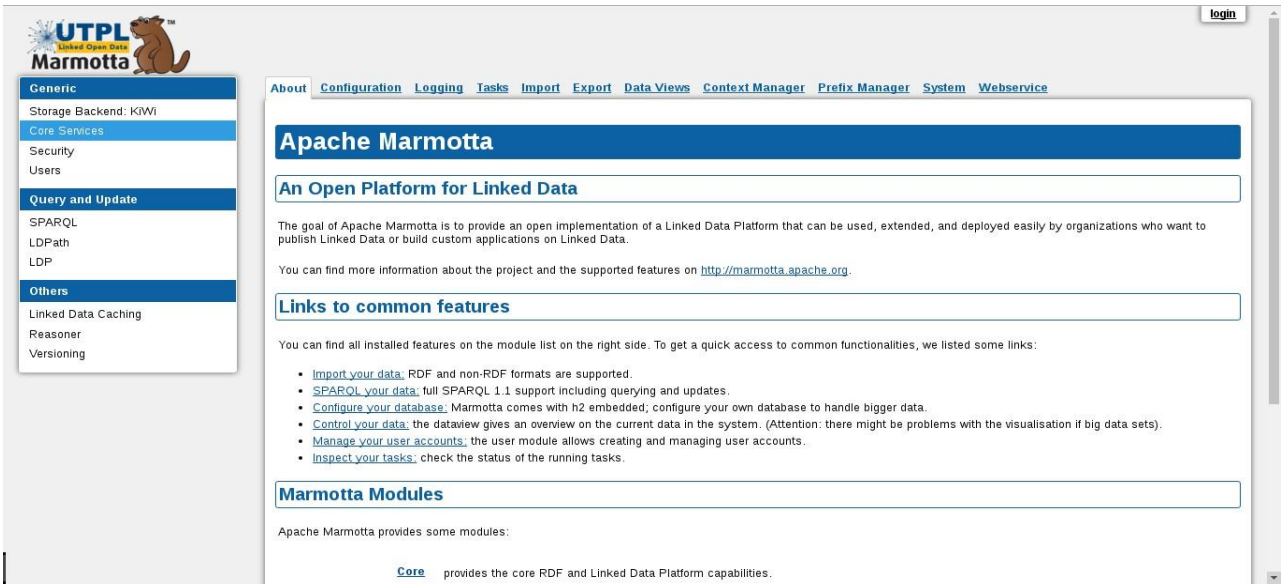


Figura 57: Pagina principal de Apache Marmotta LDP 3.3.0
Fuente: UTPL-LDP Marmotta
Elaboración: El Autor

Las configuraciones que se deben realizar en Apache Marmotta para su correcto funcionamiento son realmente necesarias, desde configuraciones de seguridad,configuraciones del Storage Backend hasta configuraciones de la apariencia de la plataforma para un correcto funcionamiento de ambientes de producción y manejo de grandes cantidades de datos

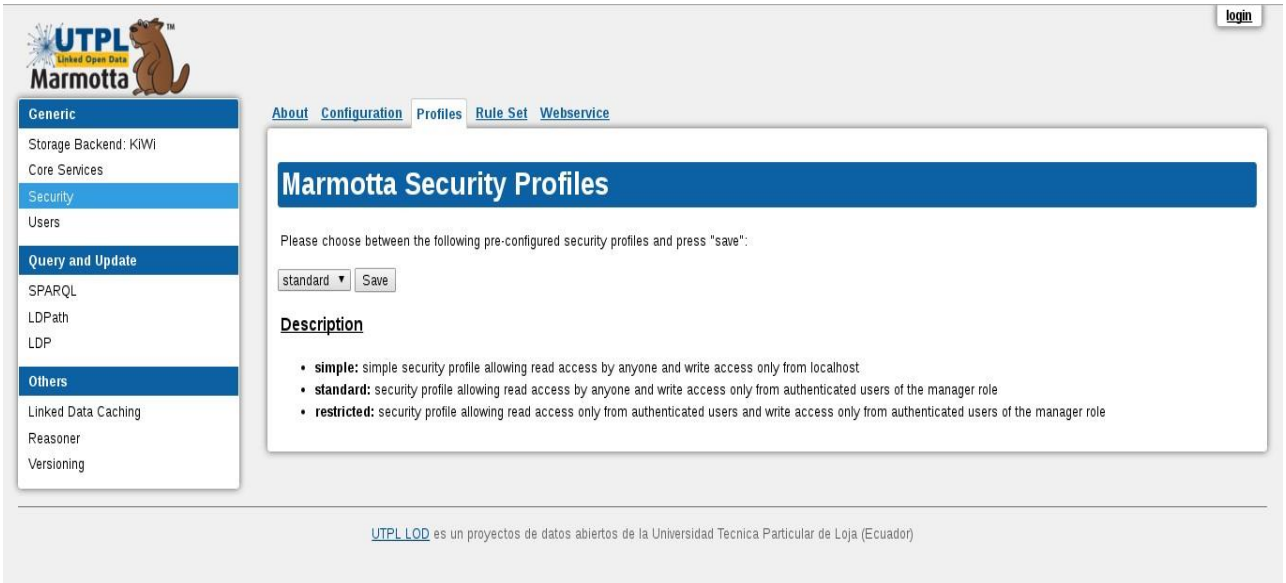


Figura 58: Modulo de Seguridad de Marmotta
Fuente: UTPL-LDP Marmotta
Elaboración: El Autor

El módulo de seguridad de Apache Marmotta registra tres tipos de perfiles (simple, standard, restricted) que puede manejar la plataforma, tal como se muestras en la figura 57.

Simple: permite el acceso de lectura y escritura por cualquier persona el acceso sólo desde localhost

Standard permite el acceso de lectura por cualquier persona y escribir únicamente el acceso a los usuarios autenticados de la función de administrador

Restricted Permita el acceso de lectura sólo de usuarios autenticados y escribir únicamente el acceso a los usuarios autenticados de la función de administrador

El cambio de estos perfiles se los puede hacer directamente desde la interfaz de administración de Marmotta o desde la edición del archivo “system-config.properties”, ubicado en la carpeta base de Marmotta

La figura 58 muestra la porción de código que debe ser agregada en el archivo system-config.properties.

```
user.anonymous.webid = http://gestiondatos.utpl.edu.ec:8090/marmotta/user/anonymous
security.profile = standard
user.datos.pwhash = :sha1::132c32e44955bdd65161a2d3b8d4dcc9b870990e
```

Figura 59: extracto del archivo de configuración temporal de Marmotta
Fuente: UTPL-LDP Marmotta
Elaboración: El Autor

La configuración del modulo de Linked Data Caching es fundamental para que los datos que albergar Marmotta se puedan relacionar con recursos existentes en Liked Data Cloud y de esta forma llegar a la 5 estrella de datos abiertos; en nuestra istancia de Apache Marmota se han configurado dos endpoint adicionales a los que Marmotta tiene por defecto (DBpedia y DBpediaLatam), como se muestra en la figura 59.

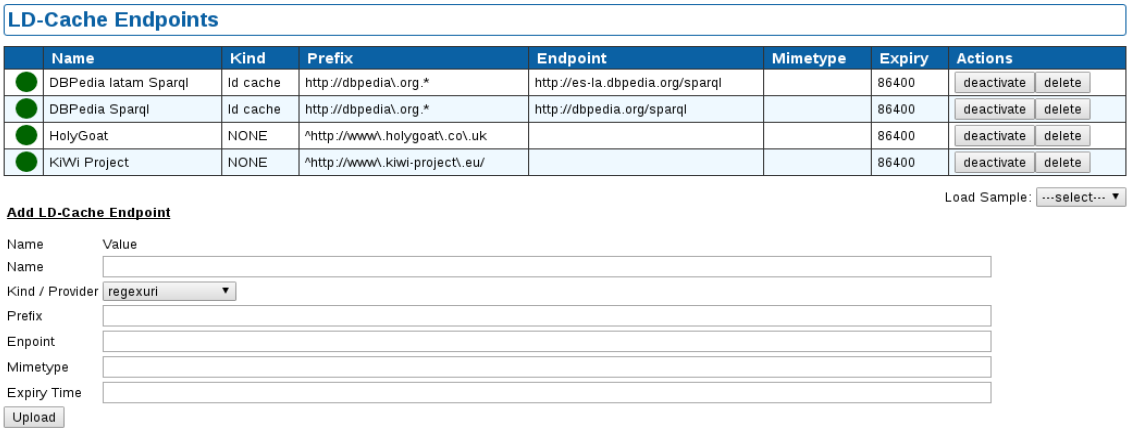


Figura 60: SPARQL endpoint configurados en Marmota
Fuente: UTPL-LDP Marmotta
Elaboración: El Autor

ANEXO 2. Descripción RDFS de la Ontología de la Solución.

A continuación se proporciona todo el RDFS de la ontología diseñada para la implementación de la solución propuesta en el presente trabajo de fin de titulación.

Detalle 8.1: RDFS de la Ontología Persona

```
0 <?xml version="1.0"?>
1 <rdf:RDF xmlns="#"
2      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3      xmlns:owl="http://www.w3.org/2002/07/owl#"
4      xmlns:xml="http://www.w3.org/XML/1998/namespace"
5      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
6      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
```

```
7     <owl:Ontology/>
8
9     <!-- http://data.utpl.edu.ec/utpl/lod#isPartOf -->
10
11    <owl:ObjectProperty rdf:about="http://data.utpl.edu.ec/utpl/lod#
          isPartOf">
12        <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
              Estudiante"/>
13        <rdfs:range rdf:resource="http://purl.org/vocab/aiiso/schema#
              Faculty"/>
14    </owl:ObjectProperty>
15
16    <!-- http://www.w3.org/ns/org#memberOf -->
17
18    <owl:ObjectProperty rdf:about="http://www.w3.org/ns/org#memberOf">
19        <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
20        <rdfs:range rdf:resource="http://dbpedia.org/ontology/University
              "/>
21    </owl:ObjectProperty>
22
23    <!-- http://data.utpl.edu.ec/utpl/lod/ontology/LaboralStatus -->
24
25    <owl:DatatypeProperty rdf:about="http://data.utpl.edu.ec/utpl/lod/
          ontology/LaboralStatus">
26        <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
              Autoridad"/>
27        <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
              Docente"/>
28        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
              Literal"/>
29    </owl:DatatypeProperty>
30
31    <!-- http://data.utpl.edu.ec/utpl/lod/ontology/MatiralStatus -->
32
33    <owl:DatatypeProperty rdf:about="http://data.utpl.edu.ec/utpl/lod/
          ontology/MatiralStatus">
34        <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
```

```
35          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
36      </owl:DatatypeProperty>
37
38      <!-- http://data.utpl.edu.ec/utpl/lod/page/ontology#AcademicStatus
            -->
39
40      <owl:DatatypeProperty rdf:about="http://data.utpl.edu.ec/utpl/lod/
            page/ontology#AcademicStatus">
41          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Estudiante"/>
42          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
43      </owl:DatatypeProperty>
44
45      <!-- http://purl.org/dc/terms/identifier -->
46
47      <owl:DatatypeProperty rdf:about="http://purl.org/dc/terms/identifier
            ">
48          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
49          <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
50          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
51      </owl:DatatypeProperty>
52
53      <!-- http://schema.org/birthPlace -->
54
55      <owl:DatatypeProperty rdf:about="http://schema.org/birthPlace">
56          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
57          <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
58          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
59      </owl:DatatypeProperty>
60
61      <!-- http://schema.org/endDate -->
62
```

```
63      <owl:DatatypeProperty rdf:about="http://schema.org/endDate">
64          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
65          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Autoridad"/>
66          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Docente"/>
67          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
                dateTime"/>
68      </owl:DatatypeProperty>
69
70      <!-- http://schema.org/startDate -->
71
72      <owl:DatatypeProperty rdf:about="http://schema.org/startDate">
73          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
74          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Autoridad"/>
75          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Docente"/>
76          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
                dateTime"/>
77      </owl:DatatypeProperty>
78
79      <!-- http://vivoweb.org/ontology/core#vivo:AcademicDegree -->
80
81      <owl:DatatypeProperty rdf:about="http://vivoweb.org/ontology/core#
            vivo:AcademicDegree">
82          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Autoridad"/>
83          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Docente"/>
84          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
85      </owl:DatatypeProperty>
86
87      <!-- http://www.loc.gov/mads/rdf/v1#extension -->
88
```

```
89      <owl:DatatypeProperty rdf:about="http://www.loc.gov/mads/rdf/v1#
            extension">
90          <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Autoridad"/>
91          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
92      </owl:DatatypeProperty>
93
94      <!-- http://www.ontotext.com/proton/protonex#nationalityOf -->
95
96      <owl:DatatypeProperty rdf:about="http://www.ontotext.com/proton/
            protonex#nationalityOf">
97          <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
98          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
99      </owl:DatatypeProperty>
100
101     <!-- http://www.w3.org/2000/01/rdf-schema#label -->
102
103     <owl:DatatypeProperty rdf:about="http://www.w3.org/2000/01/
            rdf-schema#label">
104         <rdfs:domain rdf:resource="http://dbpedia.org/ontology/
                University"/>
105         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
106     </owl:DatatypeProperty>
107
108     <!-- http://www.w3.org/ns/org#Role -->
109
110     <owl:DatatypeProperty rdf:about="http://www.w3.org/ns/org#Role">
111         <rdfs:domain rdf:resource="http://data.utpl.edu.ec/utpl/lod#
                Autoridad"/>
112         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
113     </owl:DatatypeProperty>
114
115     <!-- http://xmlns.com/foaf/0.1/Age -->
116
```

```
117     <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/Age">
118         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
119         <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
120         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
121     </owl:DatatypeProperty>
122
123     <!-- http://xmlns.com/foaf/0.1/FamilyName -->
124
125     <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/
            FamilyName">
126         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
127         <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
128         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
129     </owl:DatatypeProperty>
130
131     <!-- http://xmlns.com/foaf/0.1/birthday -->
132
133     <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/birthday"
            >
134         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
135         <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
136         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
                dateTime"/>
137     </owl:DatatypeProperty>
138
139     <!-- http://xmlns.com/foaf/0.1/gender -->
140
141     <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/gender">
142         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
143         <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
144         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
```

```
145     </owl:DatatypeProperty>
146
147     <!-- http://xmlns.com/foaf/0.1/mbox -->
148
149     <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/mbox">
150         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
151         <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
152         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
153     </owl:DatatypeProperty>
154
155     <!-- http://xmlns.com/foaf/0.1/name -->
156
157     <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/name">
158         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
159         <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
160         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
161     </owl:DatatypeProperty>
162
163     <!-- http://xmlns.com/foaf/0.1/phone -->
164
165     <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/phone">
166         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
                topDataProperty"/>
167         <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
168         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#
                Literal"/>
169     </owl:DatatypeProperty>
170
171     <!-- http://data.utpl.edu.ec/utpl/lod#Autoridad -->
172
173     <owl:Class rdf:about="http://data.utpl.edu.ec/utpl/lod#Autoridad">
174         <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Person"
                />
175     </owl:Class>
```

```
176
177     <!-- http://data.utpl.edu.ec/utpl/lod#Docente -->
178
179     <owl:Class rdf:about="http://data.utpl.edu.ec/utpl/lod#Docente">
180         <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Person"
                />
181     </owl:Class>
182
183     <!-- http://data.utpl.edu.ec/utpl/lod#Estudiante -->
184
185     <owl:Class rdf:about="http://data.utpl.edu.ec/utpl/lod#Estudiante">
186         <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Person"
                />
187     </owl:Class>
188
189     <!-- http://dbpedia.org/ontology/University -->
190
191     <owl:Class rdf:about="http://dbpedia.org/ontology/University"/>
192
193     <!-- http://purl.org/vocab/aiiso/schema#Faculty -->
194
195     <owl:Class rdf:about="http://purl.org/vocab/aiiso/schema#Faculty"/>
196
197     <!-- http://xmlns.com/foaf/0.1/Person -->
198
199     <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Person"/>
200
201     <owl:NamedIndividual rdf:about="http://data.utpl.edu.ec/utpl/lod#
            Docente">
202         <rdfs:label>Docente</rdfs:label>
203     </owl:NamedIndividual>
204
205     <!-- http://data.utpl.edu.ec/utpl/lod#Estudiante -->
206
207     <owl:NamedIndividual rdf:about="http://data.utpl.edu.ec/utpl/lod#
            Estudiante">
208         <rdfs:label>Estudiante</rdfs:label>
209     </owl:NamedIndividual>
```

```
210
211     <!-- http://data.utpl.edu.ec/utpl/lod/ontology/MatiralStatus -->
212
213     <owl:NamedIndividual rdf:about="http://data.utpl.edu.ec/utpl/lod/
            ontology/MatiralStatus">
214         <rdfs:label>lodutpl:MaritalStatus</rdfs:label>
215     </owl:NamedIndividual>
216
217     <!-- http://dbpedia.org/ontology/University -->
218
219     <owl:NamedIndividual rdf:about="http://dbpedia.org/ontology/
            University">
220         <rdfs:label>dbpedia-owl:University</rdfs:label>
221     </owl:NamedIndividual>
222
223     <!-- http://purl.org/dc/terms/identifier -->
224
225     <owl:NamedIndividual rdf:about="http://purl.org/dc/terms/identifier"
            >
226         <rdfs:label>dcterms:identifier</rdfs:label>
227     </owl:NamedIndividual>
228
229     <!-- http://schema.org/birthPlace -->
230
231     <owl:NamedIndividual rdf:about="http://schema.org/birthPlace">
232         <rdfs:label>schema:birthPlace</rdfs:label>
233     </owl:NamedIndividual>
234
235     <!-- http://schema.org/endDate -->
236
237     <owl:NamedIndividual rdf:about="http://schema.org/endDate">
238         <rdfs:label>schema:endDate</rdfs:label>
239     </owl:NamedIndividual>
240
241     <!-- http://schema.org/startDate -->
242
243     <owl:NamedIndividual rdf:about="http://schema.org/startDate">
244         <rdfs:label>schema:startDate</rdfs:label>
```

```
245     </owl:NamedIndividual>
246
247     <!-- http://xmlns.com/foaf/0.1/Age -->
248
249     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/Age">
250         <rdfs:label>foaf:Age</rdfs:label>
251     </owl:NamedIndividual>
252
253     <!-- http://xmlns.com/foaf/0.1/FamilyName -->
254
255     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/FamilyName
        ">
256         <rdfs:label>foaf:FamilyName</rdfs:label>
257     </owl:NamedIndividual>
258
259     <!-- http://xmlns.com/foaf/0.1/Person -->
260
261     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/Person">
262         <rdfs:label>foaf:Person</rdfs:label>
263     </owl:NamedIndividual>
264
265     <!-- http://xmlns.com/foaf/0.1/birthday -->
266
267     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/birthday">
268         <rdfs:label>foaf:birthday</rdfs:label>
269     </owl:NamedIndividual>
270
271     <!-- http://xmlns.com/foaf/0.1/gender -->
272
273     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/gender">
274         <rdfs:label>foaf:gender</rdfs:label>
275     </owl:NamedIndividual>
276
277     <!-- http://xmlns.com/foaf/0.1/mbox -->
278
279     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/mbox">
280         <rdfs:label>foaf:mbox</rdfs:label>
281     </owl:NamedIndividual>
```

```
282
283     <!-- http://xmlns.com/foaf/0.1/name -->
284
285     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/name">
286         <rdfs:label>foaf:name</rdfs:label>
287     </owl:NamedIndividual>
288
289     <!-- http://xmlns.com/foaf/0.1/phone -->
290
291     <owl:NamedIndividual rdf:about="http://xmlns.com/foaf/0.1/phone">
292         <rdfs:label>foaf:phone</rdfs:label>
293     </owl:NamedIndividual>
294 </rdf:RDF>
```

ANEXO 3. Modelo Físico de Datos y Listado de Tablas.

La tabla 35 muestra un listado de las tablas empleadas y una descripción de la información que almacena y sus relaciones con las demás tablas del modelo.

Tabla 35: Lista de Tablas y Descripción

Nombre	Descripción
TDI_CARGO	Almacena los cargos que pueden tener los docentes. Además indica según el cargo la carga administrativa correspondiente.
TDI_CARGO_DOCENTE	Almacena la información histórica del docente con su cargo diariamente
TDI_CARGO_PERFIL	Almacena información del perfil al cual pertenece cada carga
TDI_COMPONENTE_EDUCATIVO	Indica los componentes educativos, asignaturas y materias
TDI_DOCENTE	Almacena la información general del Docente. Esta información no cambia por periodos. Además contiene las equivalencias de los Sistemas Externos con los que se integrará la información
TDI_ESTUDIANTE	Almacena la información general del estudiante, así como sus equivalencias con otros sistemas externos, para poder integrarlos
TDI_GLOSARIO	Permite almacenar los ítems y catálogos del sistema
TDI_SECCION_DEPARTAMENTAL	Almacena las secciones departamentales existentes

Fuente: El Autor
Elaboración: El Autor