UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Desarrollo de una API para el consumo y envío de datos RDF desde una aplicación Android a un repositorio en la nube.

TRABAJO DE TITULACIÓN.

AUTOR: Jaramillo Torres, Patricio Andrés

DIRECTOR: Ramírez Coronel, Ramiro Leonardo, Mgs.

LOJA-ECUADOR

2017

2017

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Ingeniero.

Ramírez Coronel Ramiro Leonardo

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: Desarrollo de una API para el consumo y envío de datos RDF desde una aplicación Android a un repositorio en la nube, realizado por Jaramillo Torres Patricio Andrés, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, agosto de 2017

f)……..

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

“Yo, Jaramillo Torres Patricio Andrés declaro ser autor (a) del presente trabajo de titulación: Desarrollo de una API para el consumo y envío de datos RDF desde una aplicación Android a un repositorio en la nube, de la Titulación Sistemas Informáticos y Computación, siendo Ramírez Coronel Ramiro Leonardo director (a) 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 investigativo, 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 o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad”.

f……

Autor: Patricio Andrés Jaramillo Torres

Cédula: 1104745706

DEDICATORIA

Dedico este trabajo primeramente a Dios por brindarme cada una de las oportunidades de salir adelante, y ser la guía en cada momento en mi vida.

A mis padres y hermanas quienes han sido un enorme apoyo, por ser mi compañía y mi soporte para no decaer y siempre seguir adelante, superando cada adversidad de la vida de la mejor manera.

A mis demás familiares y amigos los cuales han brindado su tiempo y conocimiento, en todo el crecimiento profesional que he tenido y más aún como persona.

A todas aquellas personas que fueron muy trascendentales en mi vida, pero ya no se encuentran conmigo, jamás olvidare todo el apoyo que siempre me brindaron.

AGRADECIMIENTO

Agradezco a Dios por brindarme esta oportunidad de alcanzar una más de mis metas, al igual como me brindo una familia y amigos, los cuales siempre están cuando más los necesito.

A mis padres y hermanas por brindarme el apoyo durante todo el proceso de formación profesional y como persona.

A mis familiares y amigos por estar presentes día a día, y que de cierta manera fueron apoyo para lograr un objetivo más.

A mi director de tesis por brindarme su tiempo y conocimiento al ser la guía para el desarrollo de este trabajo.

A la universidad y docentes por ser parte de la formación profesional, la cual aplicare como profesional.

ÍNDICE DE CONTENIDOS

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

RESUMEN

Este trabajo describe el desarrollo de un API en PHP para poder enviar y consumir datos RDF desde un dispositivo móvil Android, hacia un repositorio semántico en la nube. Tomando en cuenta que en la actualidad hay un gran número de personas, que hacen uso diariamente de dispositivos móviles para diversas actividades y entre una de ellas es el consumo de información. Por lo que se requiere poder vincular datos semánticos, en un lenguaje que pueda ser procesado por una aplicación en los dispositivos móviles Android.

Es así como se pretende mejorar el consumo y envió de datos, considerando que los usuarios que hagan uso del aplicativo, no necesitan tener conocimiento de datos semánticos. Para lo cual se requiere el construir un API que permita obtener o enviar información hacia una ontología que este en la nube, la cual es ingresada en un lenguaje natural en la aplicación, pero dicha información será transformada a una consulta dinámicas SPARQL, y obtener así como resultado listados de información y almacenamiento sin ningún inconveniente, como errores de consultas o una mala asignación de los atributos que sean necesarios.

Palabras Claves: RDF, Datos, Android, API, Ontología, PHP, SPARQL.

ABSTRACT

The present research describes the development of an API in PHP to send and consume RDF data from an Android mobile device, to a semantic repository in the cloud. Taking into account that there are currently a large number of people that make use daily of mobile devices for various activities and between one of them is the consumption of information. Therefore, it is necessary to be able to link data semantic, in a language that can be processed by an application in the mobile devices Android.

This is how to improve the consumption and sending of data, considering that users who make use of the application do not need to have knowledge of semantic data. For which it is required to build an API that allows to obtain or send information to an ontology that is in the cloud, which is entered in a natural language in the application, but such information will be transformed to a dynamic query SPARQL, and thus obtain As a result listings of information and storage without any inconvenience, such as query errors or misallocation of the necessary attributes.

Keywords: RDF, Data, Android, API, Ontology, PHP, SPARQL

INTRODUCCIÓN

Existen diversas aplicaciones Android que con el paso del tiempo han ido evolucionando, pasando de ser informativas a dinámicas, lo cual busca la intervención del ser humano, y es este último quien busca donde consumir o almacenar información, la misma que puede ser recolectada en bases de datos. Pero en la actualidad se pretende mejorar la interoperabilidad entre los sistemas informáticos existentes y los que aún están por crearse, esta mejora pretende hacer uso de agentes inteligente, los cuales son programas que buscan operar sin la intervención humana para extracción de la información. Para esto y considerando que diversas personas hacen uso de dispositivos móviles Android, se pretende que formen parte de la Web Semántica, por lo cual se proyecta la creación de una API para mejorar el consumo y envío de datos, desde un dispositivo móvil a un repositorio de datos semánticos. De manera que permita la actualización de la información para que pueda ser aprovechada por quien así lo requiera.

No obstante, debido a la gran cantidad de dispositivos móviles presentes en la actualidad, aún existen limitaciones para gestionar estos datos semánticos. De manera que al tener internet de forma más asequible, se pretende dar la oportunidad de que cualquier usuario pueda almacenar información y consumir datos RDF, para lo cual se requiere el uso de un API, teniendo en cuenta esto el primer paso es investigar proyectos semejantes para el desarrollo de un API en PHP. A continuación, realizar la conexión del dispositivo móvil al API, para almacenar los datos en Virtuoso, para que luego estos datos se muestren en la aplicación móvil.

Una vez implementada la aplicación y comprobando su correcto almacenamiento local, se buscara una ontología que permita almacenar los datos, ya con esto empezar con el desarrollo de un API para almacenar y consultar información en un la ontología establecida, integrando así el API con la aplicación Android para sincronizar los datos entre el repositorio y la aplicación. Concluyendo con la realización de pruebas necesarias para verificar que el almacenamiento y consultas se estén cumpliendo con normalidad y sin presentar ningún error. Para el desarrollo de la aplicación Android y el API en PHP, se aplicara una adaptación de la metodología ágil XP (Xtreme Programming), que usa un enfoque incremental para el desarrollo y pruebas funcionales.

Los objetivos para cumplir con el desarrollo de una API para el consumo y envío de datos RDF desde una aplicación Android a un repositorio en la nube, son los siguientes:

- Investigar tecnologías para el desarrollo de APIs, consumo de datos RDF y almacenamiento en la nube.

- Administrar información desde dispositivo móvil con sistema operativo Android.
- Crear o buscar una ontología que se adecue a las necesidades turísticas.
- Desarrollar un API para enviar datos semánticos desde un dispositivo móvil.
- Mejorar el almacenamiento semántico desde tecnologías móviles.

El presente trabajo de titulación se estructura de la siguiente manera, el capítulo 1 describe conceptos relacionados sobre el desarrollo de aplicación móviles Android, desarrollo de un API en PHP y creación de una ontología. El capítulo 2 especifica el diseño de la solución, describiendo la arquitectura de la aplicación móvil así como del consumo del API, para realizar las acciones requeridas de mantener sincronizados los datos de la aplicación con el repositorio semántico en la nube. El capítulo 3 comprende el desarrollo de la aplicación móvil Android, el desarrollo de un API para el consumo y envío de datos RDF desde una aplicación Android a un repositorio en la nube, y de la creación de la ontología donde se almacenaran los datos. El capítulo 4 consta de la implementación y los resultados obtenidos sobre el almacenamiento y consultas de datos sobre el repositorio semántico.

CAPÍTULO 1: MARCO TEÓRICO

En el presente capitulo se realizara una descripción concreta de ciertos aspectos relevantes como es la Web Semántica, los servicios web, aplicaciones móviles y su relación se concluye con trabajos relacionados, para lo cual se implementó la metodología SLR, que permite evaluar e interpretar diversas investigaciones en base a ciertas interrogantes, para obtener información necesaria de acuerdo a un área de interés en específico, en base a esto se pudo identificar algunos documentos que permiten conocer cómo crear un API para consumir y enviar datos, desde un dispositivo móvil a un repositorio de datos semánticos.

1.1. Web Semántica

Desde la aparición de la WWW (World Wide Web) y la evolución que ha tenido hasta la actualidad se ha convertido a una herramienta vital y de uso cotidiano, que ha transformado la manera de comunicar a la sociedad, de relacionar a las personas, y ha permitido realizar actividades de ocio o para el trabajo. En sus inicios la manera de consultar información en la Web era unidireccional y mostraba paginas estáticas llamando así a esta como Web 1.0, pero con el tiempo ha mostrado una evolución en sus tecnologías (HTML, CSS, PHP, JavaScript, Ajax, entre otras.), es así como en lo que ahora conocemos como Web 2.0 cuyo objetivo es la intervención de usuarios para producir y consumir contenido.

En la actualidad se ha implementado la Web 3.0 o conocida como la Web Semántica, que presenta como finalidad primordial que la información presente en la Web pueda procesarse automáticamente, para ello se hace uso de agentes inteligentes lo cual consiste en programas que procesar información y razonar de manera lógica sin la intervención humana.

La web semántica ayuda a resolver problemas de sobrecarga de información y heterogeneidad de las fuentes de información. De esta manera, la infraestructura de la web semántica al estar basada en metadatos (información de los datos) aporta una extensión a las capacidades de la misma, pudiendo los datos ser interpretados tanto por agentes humanos como por agentes computarizados.(S. González, 2015, p. 4)

1.1.1. Arquitectura de la Web Semántica.

Berners-Lee (2000) propuso una arquitectura para la web semántica Figura 1, la cual ha tenido mucha aceptación por diversos autores, en el cual se representan diversas capas que se explican a continuación.

Capa de recursos

Este nivel comprende las URIs y Unicode, siendo un nivel que permite identificar los recursos Web, dando la relevancia de ser quienes definen el conjunto de recursos distribuidos en la red.

- **URI**

  En español se conoce como identificador de recursos uniforme, lo cual corresponde a una cadena de caracteres que permiten identificar de objetos en la red.

  La URI, a diferencia de la URL (Uniform Resource Locator) no están orientadas a localizar recursos, sino a su identificación, por lo tanto no es necesaria su existencia física y puede ser utilizado para identificar conceptos abstractos utilizados dentro de un modelo RDF.(Lamas, 2006, p. 20)

- **UNICODE**

  Es un estándar de codificación de texto que define un nombre e identificador numérico único para cada carácter o símbolo. Lo cual permite asignar a cada posible carácter un número o nombre único de manera que exprese información sin importar el idioma.

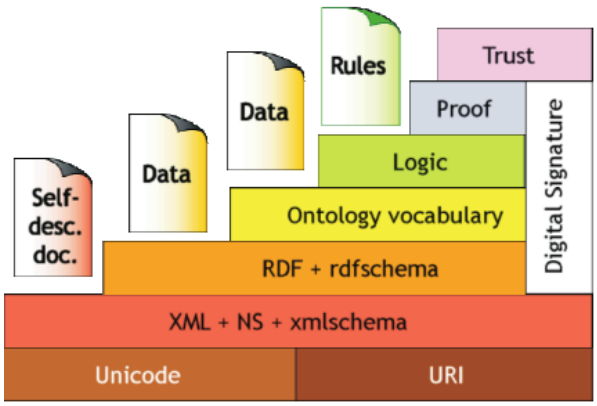

Figura 1. Arquitectura web semántica propuesta por Berners-Lee.
Fuente: Llamas, M. I. (2006). Lenguajes de consulta para documentos RDF. Recuperado de https://goo.gl/TpYddA
Elaboración: Llamas, M. I. (2006). Lenguajes de consulta para documentos RDF. Recuperado de https://goo.gl/TpYddA

Capa Sintáctica

En el presente nivel se soluciona el problema de definir diversos lenguajes de etiquetado para añadir contenido semántico en las páginas web.

- **XML (eXtensible Model Language)**

  Es un metalenguaje lo que corresponde a decir que es un lenguaje para escribir lenguajes. La particularidad que presenta XML es dar la posibilidad tanto de

formalizar y validar documentos, permitir presentar información de manera estructurada y tener la opción de ampliación. El principal uso es permitir la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

"Los entusiastas de XML opinan que algún día todos los navegadores procesarán XML en vez de HTML, a través de una migración progresiva mediante XHTML11, SVG12 (Scalable Vector Graphic), Xlink13, etc." (Lamas, 2006, p. 17).

Capa Semántica

- **RDF**

  En español se conoce como Marco de Descripción de Recursos, está dentro de las especificaciones de World Wide Web Consortium (W3C) está diseñado para normalizar la definición y el uso de las descripciones de los meta datos de los recursos que se encuentran en la Web. No obstante, también es empleado para representar simples datos. El modelo de datos RDF tiene similitud con el modelado conceptual clásico (entidad relación o diagrama de clases) pues se basa en la idea de hacer declaraciones sobre los recursos especialmente en recursos web en forma de expresiones sujeto-predicado-objeto. RDF es un lenguaje de etiquetado fundamentado en XML.

  "RDF ofrece estructuras formales (no ambiguas) gracias al uso de URIs, permitiendo la codificación, el intercambio y el procesamiento automatizado de meta-datos normalizados" (Lamas, 2006, p. 17).

Capa ontológica

- **Ontologías**

  Las ontologías de acuerdo con Berners-Lee, son la parte fundamental ya que pretende que el conocimiento no debe ser comprensible para un solo ordenador, sino que este conocimiento debe permitirse operar entre varios ordenadores y así mismo ser reutilizable. Las ontologías son la clave para representar el conocimiento.

  "El lenguaje de ontologías web (OWL) es la forma más apropiada de compartir conocimiento en la Web. Técnicamente, OWL se descompone en tres sublenguajes incrementales para ser utilizado según las necesidades: OWL lite, OWL DL y OWL full" (Lamas, 2006, p. 17).

Capa lógica

Este nivel está compuesto por diversos axiomas y reglas de inferencia que consiste en una función que tomas las premisas, analiza la sintaxis y retorna una o varias conclusiones, a partir de estas conclusiones los agentes ya sean humanos o computacionales pueden relacionar y procesar información según se requiera. Las reglas mencionadas permiten generar nuevas sentencias en base de datos y estructuras definidas en las capas XML Y RDF, y usando además datos y estructuras de la capa ontológica. En este nivel de la lógica podemos concluir que se pretende el dar esa flexibilidad a la arquitectura para que de esta manera se realicen las consultas necesarias y se pueda inferir conocimiento en base a las ontologías del nivel anterior.

Últimas capas

En las últimas capas consideramos el nivel de seguridad con lo cual nos permite asignar grados tanto de confianza, como de seguridad a aquellos recursos existentes y que se encuentran distribuidos en el Internet, esto ya mediante las firmas digitales, lo que son redes de confianza o cualquier técnica de autenticación en la red. Las firmas digitales corresponde a bloques encriptados de datos que tanto ordenadores y agentes puede usar para verificar información que ha sido adicionada, la cual viene de una fuente que ha sido identificada y es de confianza. En la actualidad no existe una recomendación sugerida por W3C sobre el resto de capas de la arquitectura propuesta.

1.1.2. Ontología.

Las ontologías nos permiten capturar conocimiento sobre ciertos dominios que son de interés, de igual manera proveen de un correcto entendimiento del dominio al cual está representando. Lo que quiere decir es que las ontologías permiten describir los conceptos del dominio y aquellas relaciones que existen entre los mismos.

Dentro de las ontologías existen algunos componentes importantes, como podemos ver en la figura 2 algunos de sus componentes más relevantes.

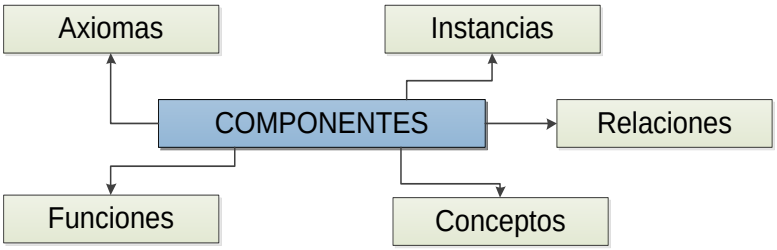

Figura 2. Componentes de la ontología.
Fuente: El autor
Elaboración: El autor

- **Conceptos**

  Comprende aquellas ideas básicas que se pretenden formalizar. Dentro de esto se encuentra las clases de objetos, métodos, planes, estrategias, entre otros.
- **Relaciones**

  Es la interacción y enlace presente entre los conceptos del dominio, estas relaciones son la parte que forma la taxonomía, como ejemplo esta: sub_clase_de, pertenece_a, etc.
- **Funciones**

  Corresponde a un concreto tipo de relaciones donde se identifica un elemento en base al cálculo de una función, como ejemplo tenemos: categorizar-clase, asignar-fecha, etc.
- **Instancias**

  Son usados para representar aquellos objetos ya determinados por un concepto.
- **Axiomas**

  Son aquellos teoremas que son declarados sobre las relaciones, los cuales deben cumplir los elementos de una ontología, por ejemplo: Si a y b pertenecen a la clase c, entonces a no es subclase de b.

Hoy en día solo las personas pueden extraer información de la web 2.0, esto se debe a que no existe aún una semántica, que pueda ser interpretada tanto por agentes inteligentes, por aplicaciones tanto web como móviles de manera que pueda reconocer la información. Es debido a esto que se busca implementar ontologías en la web semántica, para que las maquinas puedas tener la capacidad de recopilar información y lo que se busca alcanzar el razonar dicha información para la resolución de problemas.

1.1.3. Lenguajes de ontología.

Entre los más destacados se encuentra OWL y RDFS (RDF Schema) este último es una extensión de OWL.

- **OWL**

OWL es un lenguaje que ha sido recomendado por W3C, cuya función principal es que los agentes inteligentes, aplicaciones web, móviles, u otros, puedan interpretar el contenido que existe en la web. OWL emplea igual que RDF tripletas pero tiene un mecanismo mejorado, el cual interpreta de mejor manera los contenidos de la web y posee una semántica formal, con lo que se puede concluir que OWL, posee un poder más expresivo que RDF y XML.

“Se utiliza OWL cuando se requiere que la información sea procesada por el software y no para representarla”(S. González, 2015, p. 11).

- **RDFS**

RDF Schema (RDFS) proporciona un vocabulario de modelado de datos para los datos RDF, es decir, es una extensión semántica de RDF. Proporciona mecanismos para la descripción de grupos de recursos y las relaciones entre esos recursos, usando propiedades y valores.(S. González, 2015, p. 10)

RDFS como se menciono es una extensión de OWL, que a partir de este adiciona conceptos y relaciones. De igual manera brinda un vocabulario de modelado de datos para datos RDF. Es decir que proporciona mecanismos para describir grupos de recursos y relaciones que puedan existir entre esos recursos, para ello usa propiedades y valores. Así es como provee de mecanismos para especificar tantas clases, propiedades y las relaciones forman parte de un vocabulario.

1.1.4. Lenguaje de consulta: SPARQL.

SPARQL es un lenguaje que permite realizar la consulta de patrones que pueden ser obligatorios u opcionales de un grafo de conceptos, junto con sus conjunciones y disyunciones, además tienes soporte para ampliar o restringir el ámbito de las consultas mencionando los grafos sobre los que está operando, como ejemplo esta la figura 3 que muestra la estructura de la consulta. La respuesta a estas consultas puedes ser tanto de datos como de estructura de grafos semánticos.

Algunos de los elementos básicos del lenguaje son:

- **PREFIX**: declaración del espacio conceptual.
- **SELECT**: realiza una proyección de los resultados
- **DESCRIBE**: devuelve un grafo que describe los recursos encontrados.
- **ASK**: indica si la combinación sujeto-verbo-predicado existe en la ontología RDF.
- **FROM**: indica los elementos sobre los que se ejecutará la consulta, en caso de no definirse tomará todos los grafos definidos en el endpoint.
- **ORDER BY**: es opcional y permite ordenar los resultados de la consulta.
- **LIMIT Y OFFSET**: permite seleccionar un subconjunto de los resultados.
- **GROUP BY**: realiza agregación en los resultados.

```
PREFIX MD: <http://www.etsi.org/isg/moi/data.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?traceroute ?rtt {
    ?traceroute a MD:Traceroute ;
        MD:hasMeasurementData ?a .
    ?a a MD:RoundTripDelayMeasurement ;
        MD:RoundTripDelayMeasurementValue ?rtt .
    FILTER (?rtt < "20"^^xsd:long )
}
LIMIT 100
```

Figura 3. Ejemplo de consulta SPARQL.
Fuente**:** Pino, M. P. (2014). Extensión Semántica de OML.
Elaboración**:** Pino, M. P. (2014). Extensión Semántica de OML.

Hoy en día se suele emplear SPARQL 1.1 el cual es un estándar de permite definir la sintaxis y semántica del lenguaje de consulta en los endpoints semánticos. El modelo de SPARQL tiene mucho parecido con SQK, pero con la diferencia que SQL opera con datos que se encuentran en tablas y SPARQL con datos que se encuentran en grafos. Las resultados de las consultas que se establecen con SPARQL suelen ser en URIs, literales o un grafo RDF.

1.1.5. SPARQL Endpoint.

“SPARQL Endpoint es un servicio para que los humanos y las máquinas puedan realizar consultas y se pueda presentar el resultado de manera legible para ambos”(S. González, 2015, p. 12).

Es lo que se conoce en español como el punto final SPARQL, se define como la interfaz donde se puede realizar consultas, hacia una base de conocimiento mediante el lenguaje SPARQL, esta herramienta permite ejecutar consultas SPARQL sobre un grafo de entrada compuesto por tripletas RDF. Los resultados de estas consultas son formatos que una maquina puede procesar. El SPARQL Endpoint está compuesto generalmente por un mensaje de entrada y dos mensajes de salida. El mensaje de entrada pertenece a la consulta SPARQL que se espera ejecutar, existe un grafo que se muestra de manera opcional sobre el cual es ejecutada la consulta.

Los resultados de la consulta se muestran tanto en el primer mensaje que corresponden a resultados de la consulta, esto en caso de que no existan errores, y el segundo mensaje que es donde se muestran los mensajes de error en caso de que existe alguna falla en la consulta, las fallas generalmente se muestra por errores ya sea de sintaxis, de semántica, manejo de excepciones en tiempo de ejecución, entre otros problemas.

1.1.6. Endpoint Virtuoso.

Virtuoso OpenSource, un sistema de gestión de bases de datos relacionales (conocido en ingles como RDBMS) que puede representar los datos como tablas relacionales y/o grafos de propiedades. Entre otras cosas, también nos permite guardar todo el contenido de nuestro repositorio en formato RDF, actuando a modo de Triplestore. De esta manera podemos guardar la ontología en forma de grafo en este RDBMS, con el cual interactuaremos mediante SPARQL para almacenar y lanzar consultas.(González, 2016, p. 30)

Virtuoso dentro de las funcionalidades que ofrece como servidor, permite trabajar con datos enlazados conocido como Linked data. El endpoint de Virtuoso es uno de los más conocidos y avanzados que existe para lo que corresponde al control de acceso, ya que este brinda un sistema desde el cual se puede administrar permisos sobre los grafos. Virtuoso brinda un modelo de manejo del acceso similar a los que administran las bases de datos actuales. Para poder acceder al endpoint de virtuoso se realiza mediante la dirección http://localhost:8890/sparql.

1.2. Servicios Web

Existe un crecimiento exponencial en el número de servicios web, para lo cual existe una necesidad de emplear algunos mecanismos que nos permitan gestionar y monitorizar, esto ya que es complicado el determinar qué servicios están implementados y el estado en que se encuentra. Esta gestión y monitorización es muy compleja por lo que requiere algunas herramientas como ayuda.

Los servicios Web o como más se los conoce en inglés web service, poseen diversas definiciones pero la que se acopla a la definición que permite involucrar todo lo que implica es que los servicio web corresponden a un conjunto de aplicaciones o de tecnologías que tienen la capacidad de cooperar usando un conjunto de protocolos común en la web. Las aplicaciones o tecnologías que se mencionan tienen como objetivo brindar servicios, para lo cual realizan el intercambio de datos. De esta manera es como se muestran los proveedores los cuales brindan sus servicios como procedimientos remotos y por otro se encuentran los usuarios que son quienes solicitan alguno de los servicios.

1.2.1. SOAP.

“SOAP es un protocolo ligero cuyo fin es el intercambio de información de una manera descentralizada, está basado en tecnología XML, este framework de mensajería fue diseñado para ser independiente de la plataforma”(Mosquera, 2014, p. 22).

Simple Object Access Protocol más conocido por sus siglas como SOAP, el cual se deriva de un protocolo que fue creado por Dave Winer en 1998, al cual se lo conocía como XML-RCP. Microsoft, IBM y otros fueron quienes crearon a SOAP, el cual hoy en día tiene el auspicio de la W3C. SOAP corresponde a un protocolo estándar el cual especifica dos objetos en diferentes procesos, los cuales se comunican mediante el intercambio de datos XML y es de los protocolos más utilizados en los que corresponde a servicios web.

De acuerdo a varios diseñadores que han hecho uso de SOAP, han determinado de su uso es demasiado complicado, por lo que la mayor parte está haciendo uso de servicios web pero basados en REST, el cual permite administrar grandes cantidades de datos, empresas como eBAY y Google son un ejemplo de uso de REST.

1.2.2. REST.

“REST es un modelo de arquitectura de software que consta de un conjunto coordinado de restricciones arquitectónicas aplicadas a los componentes, conectores y datos, dentro de un sistema hipermedia distribuido”(Puerta, 2015, p. 18).

La arquitectura que presenta se basa en ciertos estándares que existen en la Web, lo que implica que debe hacer uso de identificadores de recursos uniformes (URI), lo que permite identificar los recursos, además el protocolo de transferencia de hipertexto (HTTP), lo cual permite acceder y modificar las representaciones de los recursos, y el marco de descripción de recursos (RDF) el cual es el modelo de datos unificado que permite la descripción de los recursos.

REST tiende a no considerar aquellos detalles de implementación de componentes y establecer una sintaxis de protocolo, en cambio busca priorizar las funciones de los componentes, aquellas restricciones sobre la interacción hacia otros componentes y la interpretación de los datos.

El uso de servicios web que se basan en REST, ha sido creciente ya que este es considerado como una arquitectura simple y ligera, en comparación de todas las arquitecturas existentes. Para que REST obtenga el éxito que tiene en la actualidad, fue necesario que se aproveche de los estándares Web y su manejo sea simple al momento de desarrollar los servicios web. Es por ello que REST va en crecimiento a diferencia de SOAP.

1.2.3. Tecnologías.

Entre las tecnologías necesarias para el desarrollo del API, se definieron los siguientes lenguajes:

- **PHP**

Hypertext Preprocessor es conocido en sus siglas como PHP, el cual corresponde a un lenguaje que es de código abierto y que tiene bastante aceptación dentro del desarrollo Web. PHP puede ser usado dentro de HTML, para esto va entre dos etiquetas que identifican en donde empieza “<?php” y termina “?>”. PHP a diferencia de lenguajes como JavaScript, ejecuta el código en el lado del servidor, para luego si generar HTML y finalmente dar respuesta al cliente.

En el lado del cliente solo se muestran resultados de ejecutar algunos scripts, sin conocer lo que existe por debajo. PHP tiene un atributo que es muy interesante como es la simplicidad, por lo que muchos desarrolladores principiantes pueden manejarlo con facilidad, pero así también PHP posee algunas características avanzadas como por ejemplo es la subida de archivos, constantes mágicas, métodos mágicos, autoloading, reflection y objetos como arrays; que son de mucho uso para programadores más avanzados.

- **JSON**

JSON es un formato liviano que nos permite el intercambio de datos. Para las personas es fácil escribirlo y leerlo, y para las maquinas es sencillo el que lo interpreten y puedan generarlos. Este lenguaje se lo conoce como un subconjunto de JavaScript, Standard ECMA-262 3rd Edition – Diciembre 1999. JSON presenta una de las propiedades que lo caracteriza principalmente lo cual es que al ser un lenguaje independiente, pero aplicable en diversos lenguajes permite realizar el intercambio de datos.

Existen dos estructuras que conforman el lenguaje JSON, lo primero corresponde a una colección de pares de nombre/valor, esto en algunos lenguajes es conocido como objeto estructura, registro, tabla hash, lista de claves, registro o arreglo asociativo, y segundo se compone de una lista ordenada de valores, esto generalmente en gran parte de los lenguajes se conoce como arreglos, ventores, listas o secuencias.

- **JAVASCRIPT**

“El lenguaje JavaScript funciona del lado del cliente y los navegadores se encargan de interpretar el código. Se utiliza principalmente para crear páginas web dinámicas”(S. González, 2015, p. 35).

JavaScript se considera como un lenguaje ligero e interpretado directamente por los navegadores, el cual está orientado a objetos que tienen funciones de primera clase, se lo conoce más como el lenguaje de script para páginas web, aunque también tiene uso sin

navegador. Este lenguaje es un script multi–paradigma el cual se basa en prototipos, es dinámico, soporta programación funcional, se encuentra orientada a objetos y es imperativa. La capacidad dinámica que posee JavaScript permite construir objetos en tiempos de ejecución, listas variables de parámetros, variables que puedes contener funciones, crear scripts dinámicos, la introspección de objetos, y la recuperación del código fuente,

1.2.4. Interfaz de programación de aplicaciones.

La Interfaz de programación de aplicaciones permite establecer una conexión entre componentes dentro del software. Dentro de esta existen llamadas hacia bibliotecas que permiten ofrecer diversos servicios.

APIs Web

La interfaz de programación de aplicaciones, en inglés conocida como API, corresponde al conjunto de rutinas que permite acceder a funciones de un software determinado. Estas APIs suelen publicarse por desarrolladores de software, los cuales brindan acceso para ejecutar alguna tarea. Lo que nos permiten las APIs es la reutilización de código que ha sido desarrollado y probado, con lo cual se conoce que ese código funciona correctamente y puede favorecer a ejecutar operaciones que se requieran.

“Finalidad de tener un único punto de acceso a los datos en tiempo real y que cualquier aplicación puede consumirla realizando las correspondientes llamadas a los servicios implementados a través de las correspondientes URLs”(S. González, 2015, p. 17).

Aquellos clientes que no han sido diseñados con la Api específicamente no funcionan, ya que el contenido no podrá ser accedido por motores de búsqueda o cualquier otro agente de web genérico, cada mashup permite acceder a datos que han sido limitados por el desarrollador.

1.3. Aplicaciones móviles

Es una aplicación informática desarrollada para ejecutarse en dispositivos inteligentes, tabletas o cualquier dispositivo móvil. La aplicación permite interactuar con el usuario para realizar alguna tarea que se especificó previamente, como puede ser aplicaciones de ocio, educativas, que requieran acceso a servicios, entre otras. La finalidad de las aplicaciones es el agilizar y facilitar actividades que requiera hacer uso cualquier persona.

Bajo las aplicaciones se requiere un sistema operativo para poder instalar y ejecutar cada una de las aplicaciones. Actualmente existen diversos sistemas operativos sobre los cuales se puede desarrollar aplicaciones móviles.

1.3.1. Sistemas operativos móviles.

Entre los sistemas operativos móviles más usados tenemos:

- **WINDOWS 10:** Es un sistema operativo lanzado por Microsoft para sus dispositivos, su año de lanzamiento fue en 2015, esto luego de dejar de lado su sistema operativo anterior Windows móvil el cual fue lanzado el 2000 no tuvo la acogida que se esperaba. En un inicio Windows 10 no tenía demasiados usuarios por la escasez de aplicaciones, pero conforme pasan los meses esto ha ido mejorando, además es un sistema integral ya que la misma versión se usa tanto en portátiles y PC como en Smartphone y Tablet.
- **IOS:** Corresponde a un sistema operativo móvil, cuyo propietario es la empresa Apple Inc. Este sistema operativo funciona solamente sobre los dispositivos fabricados por la misma empresa, estos teléfonos son conocidos como los iPhone. Su última versión presentada en este año es la 10; este sistema operativo es el segundo sistema operativo más usado, después de Android.
- **ANDROID:** "Android pertenece a Google, pero es un sistema abierto cualquier fabricante puede desarrollar en él sus productos"(Aguirre & Sinche, 2013).
  Es el sistema operativo móvil más usado, se encuentra en la versión 7 denominada Nougat y es el sistema operativo más usado, debido a la variedad de dispositivos existentes que lo implementan.

1.3.2. Android OS.

Android es un sistema operativo desarrollado por la Open Headset Alliance, la cual está conformada por empresas del área tecnológica como compañías móviles entre las que se encuentra a China Mobile, HTC, Google, Qualcomm entre otras. Es un sistema que está basado en el núcleo Linux, el cual ha sido desarrollado para teléfonos, relojes, televisores, automóviles. En un principio se conoce que Google apoyo económicamente este sistema, hasta que en 2005 fue el mismo quien lo compro. La ideología de Google sobre su sistema operativo Android es brindar un sistema operativo libre basado en la mejor experiencia que pueda tener un usuario.

1.3.3. Arquitectura de aplicaciones Android.

"La arquitectura de Android, está dividida en 4 grandes capas: Aplicación, Framework de aplicaciones, librerías de SO y Kernel Linux"(Brähler, 2010).

La figura 4 da una descripción apropiada y acertada sobre la arquitectura Android, la cual ha sido considera por diversos autores.

- **Capa de aplicación:** Son programas escritos en lenguaje Java que se ejecutan en una máquina virtual. Este nivel está conformado por diversas aplicaciones que se han instalado en un dispositivo Android.
- **Entorno de aplicaciones:** Es la capa en donde se encuentran las APIs de Google, a las cuales los desarrolladores pueden acceder bajo las siguientes líneas de diseño:
    - Vistas escalables, incluidos elementos internos.
    - Proveedor de contenidos, que accede o comparte datos entre aplicaciones.
    - Gestor de recursos, como vistas o configuraciones.
    - Gestor de notificaciones, personalizadas por la aplicación.
    - Gestor de actividades, que determina el ciclo de vida de la aplicación.
    - Gestor de paquetes, ventanas.

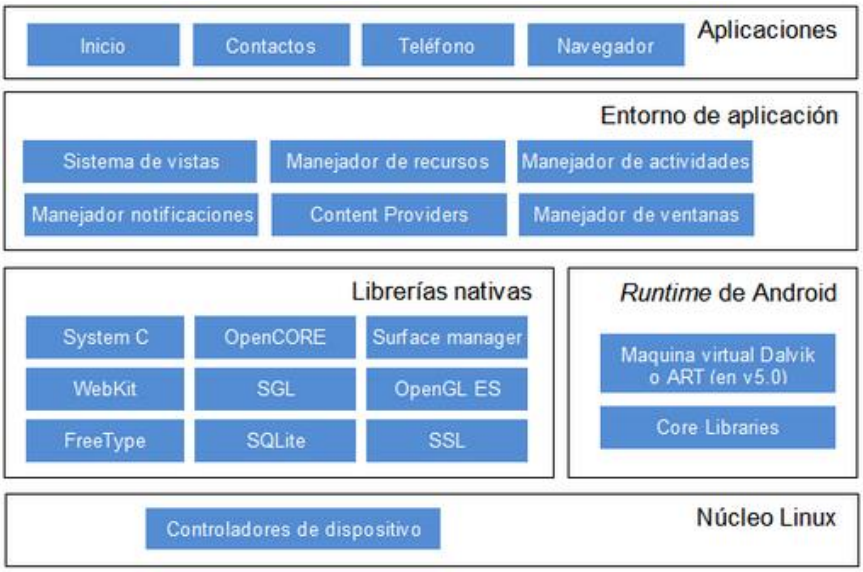

Figura 4. Arquitectura Android
Fuente: Recuperado de https://goo.gl/EHXcdV
Elaboración: Recuperado de https://goo.gl/EHXcdV

- **Librerías de sistema operativo:** existen algunos componentes soportados gracias al uso de C/C++ como son:
    - Media Framework, basado en la grabación y reproducción de múltiples formatos de audio, video e imagen.
    - SGL, motor de gráficos 2D.
    - SSL, protocolos TCP/IP con soporte de encriptación de comunicaciones de datos.
    - OpenGL, soporte de gráficos 3D.
    - SQLite, base de datos relacional.
    - Webkit, motor de navegación web.

- **Kernel Linux:** Aquellos servicios que posee Android se basan en el Kernel Linux, algunos de estos servicios que se encuentran en esta capa esta la seguridad, procesamiento de datos, gestión de memoria y protocolos. También es la interacción entre hardware y software brindado controladores para pantalla, teclado, almacenamiento, cámaras, sonido, Bluetooth, WiFi y energía.

1.3.4. Aplicaciones Android.

Google tiene una plataforma web con herramientas para desarrolladores. Actualmente, para comenzar a crear aplicaciones, primero se debe contar con la herramienta de desarrollo provista por la empresa, llamada Android Studio, un entorno de desarrollo basado en IntelliJ el cual ya posee el SDK (Software Development Kit) junto con la última librería de Android (otras para descarga) y todo lo necesario para empezar a codificar. También posee integración con otros entornos de desarrollo como Eclipse, a través de plugins.

1.3.5. Web service para Android.

Existen diversas APIs y estas son lo que se conoce comúnmente como servicios web. REST es una de las mejores alternativas al momento de la creación de servicios web, los cuáles se ejecuta sobre el protocolo HTTP, el cual actúa como mecanismo de transporte entre el servidor y el cliente. En la actualidad este intercambio de información se suele dar en formato JSON o XML.

Para realizar la transferencia mediante REST, lo que se pretende es que cada url exprese de manera limpia al recurso o la colección sobre la que se va a operar y establecer las condiciones para acceder a aquello que se requiere, de manera que esto sea intuitivo para consumir.

La comunicación de los datos como se menciono es en formato JSON o XML, a pesar de que se suele permitir otros formatos como es HTML, CSV, PDF, entre otros. En el caso de JSON se suele definir una cabecera de esta manera “Content-Type: application/json” y para XML la cabecera seria “Content-Type: text/xml”.

Existen algunas operaciones que se pueden llevar a cabo con los métodos de HTML, las acciones que generalmente más se ocupan se representan como CRUD lo que quiere decir create, read, update y delete. Para estas operaciones existen algunos métodos como se describen en la tabla 1 que permiten realizar algunas acciones.

Tabla 1. Métodos para ejecutar acciones CRUD

Método	Acción
GET	Obtiene un recurso o una colección
POST	Genera un recurso nuevo
PUT	Actualiza un recurso específico
DELETE	Elimina un recurso específico
PATCH	Actualiza parcialmente un recurso especifico

Fuente: El autor.
Elaboración: El autor

Para la respuesta del servidor el cual nos permite conocer si es satisfactorio o erróneo según el tipo de problema que exista. Comúnmente a la respuesta que suele presentar el servidor, se los organiza por familias como seria en el caso de que la consulta sea exitosa, retornara lo que se conoce como la familia 2xx, en caso de que se requiera alguna acción adicional estará en 3xx, para algunos errores que se puedan dar por el cliente esta 4xx y para los errores que puede dar el servidor 5xx.

Las cabeceras en los servicios REST presenta una diversa lista de propósitos, pero para el API lo más importante es la Autorización, lo cual permite dar autorización para el uso del API.

1.4. Trabajos relacionados

1.4.1. Metodología SLR.

El SLR ha sido propuesto en Software Engineering Research como un método para reportar conclusiones fiables acerca de un área de investigación recogiendo evidencias de calidad.

El método de revisión sistemática de literatura se basa en la metodología propuesta por Barbara Kitchenham y otros, esta metodología tiene sus raíces en revisiones bibliográficas realizadas para Ciencias Humanas y Medicina, pero en los últimos años se han propuesto adaptaciones para otras disciplinas como la ingeniería.(Rudas, Gómez, & Toro, 2013)

El proceso global de búsqueda consiste en tres fases figura 5 que permite conocer las actividades realizadas en cada etapa:

- Planificación de la búsqueda: para desarrollar un protocolo de revisión.
- Realización de la búsqueda: para ejecutar el protocolo anterior
- Presentación del informe de revisión: para proporcionar los resultados obtenidos.

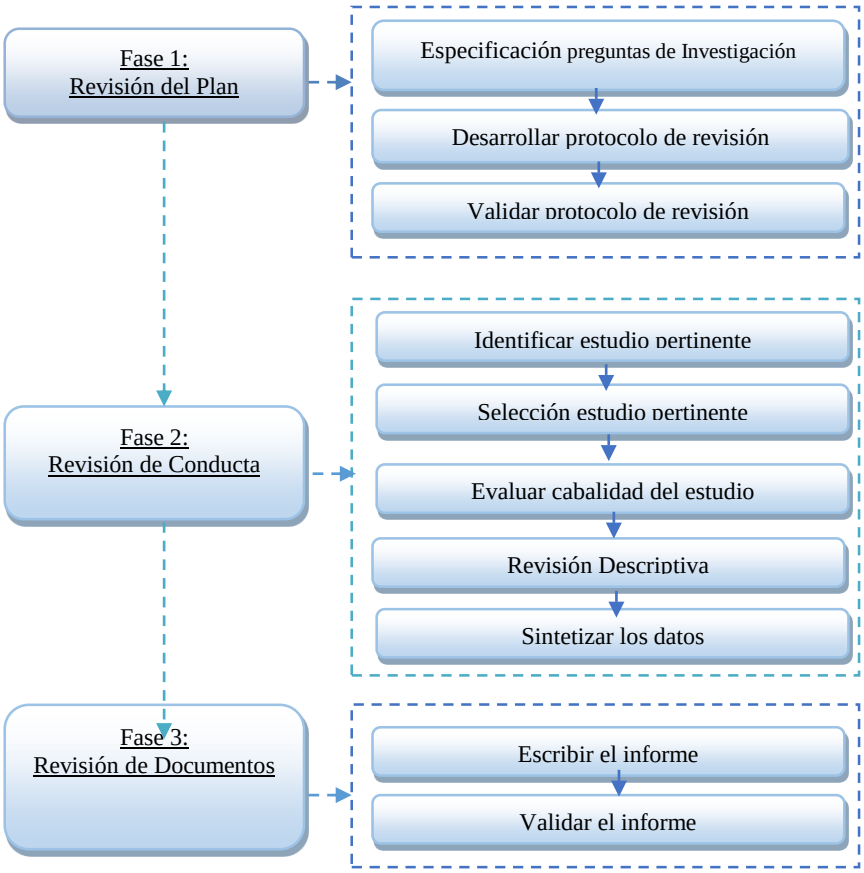

Figura 5. Fases de SLR
Fuente: El autor.
Elaboración: El autor

Una vez que los datos de los estudios integrales son extraídos correctamente, deben ser esquematizados con la finalidad de proporcionar información que pueda ser relevante. Si la información está disponible a partir de estudios en curso, se debe incluir el suministro de información de calidad apropiada sobre el estudio de criterios relacionados, con la calidad se tomaron en cuenta al evaluar los estudios identificados en el mapeo para evaluar la documentación, que se basa en el proceso de SLR para lo cual la tabla 2 permite conocer esos criterios de calidad.

Tabla 2. Criterios de calidad

Nro.	Criterios de calidad
1	¿Está el papel basado en la investigación (o es simplemente un informe "lecciones aprendidas", basada en la opinión de expertos)?
2	¿Existe una declaración clara de los objetivos de la investigación?
3	¿Existe una descripción adecuada del contexto en el que se llevó a cabo la investigación?
4	¿Fue el diseño de investigación apropiado para abordar los objetivos de la investigación?
5	¿Fue la estrategia de contratación adecuada a los objetivos de la investigación?
6	¿Hubo un grupo de control con el cual comparar los tratamientos?
7	¿Fue de los datos recogidos de una manera que abordó el tema de investigación?
8	¿Fue el análisis de datos suficientemente rigurosos?
9	¿A la relación entre el investigador y los participantes han considerado a un nivel adecuado?

Fuente: Recuperado de https://goo.gl/krF8yy.
Elaboración: Recuperado de https://goo.gl/krF8yy

1.4.2. Análisis de trabajos relacionados.

Para poder identificar los trabajos relacionados, primero se debe determinar las fuentes electrónicas, las cuales tenga validez y aceptación para lo cual tenemos la tabla 3, que son fuentes que tienen relevancia para el desarrollo de un API.

Tabla 3. Fuentes Electrónicas

Bases Electrónicas		
Id	**BD**	**Enlace**
1	IEEE Xplore	www.ieeexplore.ieee.org
2	ACM Digital Library	www.dl.acm.org
3	ISI Web of Knowledge	www.webofknowledge.com
4	Science Direct	www.sciencedirect.com
5	Springe	www.springer.com

Fuente: El autor
Elaboración: El autor

Existen varios trabajos que se relacionan con el tema propuesto “Desarrollo de una API para el consumo y envío de datos RDF desde una aplicación Android a un repositorio en la nube.” los cuales se muestran en la tabla 4 los cuales han sido determinados en base a ciertos criterios de búsqueda.

Tabla 4. Trabajos relacionados

Recurso	Año	Autor	Título
1	2011	Laia Descamps-Vila , Joan Casas , Jordi Conesa , A.Pérez-Navarro	Cómo introducir semántica en las aplicaciones SIG móviles : expectativas , teoría y realidad
2	2015	González Rodríguez Salomé	Trabajo de Fin de Grado Publicación de datos socio sanitarios : Una API basada en Open Linked Data
3	2015	Puerta González José Manuel	Desarrollo de una API para la descripción y gestión de Servicios Web REST
4	2014	Martínez González, M. Mercedes Alvite Díez, Ma. Luisa	Propuesta metodológica de evaluación de gestores de tesauros compatibles con la web semántica.
5	2012	Saorín Tomás	Cómo Linked open data impactará en las bibliotecas a través de la innovación abierta
6	2011	Rafael Mauricio Vizuete Aguilera	Información basado en Ontologías de Dominio Agrícola para las Ciencias Agropecuarias de la Universidad Central ¨Marta Abreu¨ de Las Villas
7	2015	Suaréz García Julia María	Desarrollo de una API basada en Linked Data para el sector del Turismo de Tenerife
8	2015	Flores, Lucía Batista	MEMORIA ACADÉMICA UCA LINKED DATA
9	2014	Álvarez Jesús Río	Estudio y propuesta para enriquecimiento de información utilizando fuentes Open Data: Una experiencia con videos educativos multidominio
10	2014	González Izar Sinde	ESTUDIO DE LAS POSIBILIDADES DE LOS DATOS ABIERTOS ENLAZADOS (LINKED OPEN DATA) PARA LA REALIZACIÓN DE MASHUPS DE ÁMBITO GEOGRÁFICO.
11	2016	Sulé Andreu, Centelles Miquel, Franganillo Jorge, Gascón Jesús	Aplicación del modelo de datos RDF en las colecciones digitales de bibliotecas, archivos y museos de España
12	2015	Detchart Marco	Mejora de un portal de Patrimonio Inmaterial mediante etiquetado semántico
13	2015	Juan José Puello Fuentes, Javier Enrique Peniche Padilla	PROPUESTA DE UN MODELO ARQUITECTÓNICO PARA UN SISTEMA RECOMENDADOR TURÍSTICO
14	2016	Aquino Teófilo Chambilla	LENGUAJE DE ESPECIFICACION PARA LA DELEGACION DE TAREAS EN SERVIDORES WEB MEDIANTE AGENTES
15	2015	Nydia Gabriela Fuentes Jasso, Alejandro González Solís, Sergio Iván Villaseñor Sánchez	Aplicación Web de apoyo de emergencias para pacientes con enfermedades crónicas utilizando geolocalización, Web semántica y dispositivos móviles
16	2015	Amezcua Aguilar Christian Adan, Israel Gonzáles López Ernesto	Sistema de Administración de conocimiento por medio de dispositivos móviles Agora
17	2016	Criollo Vintimilla Diego Andrés	ESTUDIO, ANÁLISIS Y DESARROLLO DE UNA APLICACIÓN WEB HÍBRIDA
18	2016	Irving Caro Fierros, Víctor Hugo Menéndez Domínguez y María Enriqueta Castellanos Bolaños	Bases de Datos Semánticas
19	2016	Schwartz Sebastian, Riesco Daniel, Abdelahad Corina	Generación automática de servicios backend y sus bases de datos utilizando MDE

Recurso	Año	Autor	Título
20	2016	García Fernández Juan Manuel	NUBV (Necesito Unas Buenas Vacaciones) Sistema Big Data para la recomendación de hoteles
21	2016	Castillo Pérez José Nelson , Hernández Díaz María Fernanda , Mosquera Rincón Nubia	Web semántica y su aporte a la estrategia de datos abiertos del Estado Colombiano
22	2015	González Bustán Ana María, Jiménez González Daniel Froilán	Principios y Tecnologías Linked Data para la publicación de Datos Académicos de las diferentes Carreras de la Universidad Nacional de Loja
23	2014	Del Razo González Victor Eliseo	Turimóvil Semántico (versión extendida): App de búsqueda turística semántica para el Centro Histórico de la Ciudad de México utilizando realidad aumentada.
24	2014	Pablo Mosquera Díaz	DESARROLLO DE NUEVA FUNCIONALIDAD E INTERFAZ WEB Y MÓ VIL PARA LA API MAPPINGAPI2
25	2016	García A Becerril	Enfoque semántico para el descubrimiento de recursos sensible al contexto sobre contenidos académicos estructurados con OAI-PMH
26	2014	Marisol De Jesús Paredes Reyes	Integración semántica de datos geoespaciales utilizando una ontología de aplicación
27	2016	Luis Alberto de Paz Benhamu	Desarrollo de herramienta de generación de Linked Data Educativo
28	2016	Lina Marcela García Vásquez	Editor Gráfico De Ontologías Para Los Lenguajes Semánticos Rdf, Rdf(S) Y Owl, Como Extensión Del Framework Ontoconcept.
29	2016	González Enrique	Enriquecimiento automático de un Data Lake con metadatos
30	2016	Antonio Yoan, Rodríguez López, Delgado Yusniel Hidalgo, Martínez Nemury Silega	Método para la integración de ontologías en un sistema para la evaluación de créditos
31	2016	Soledad Sonzini María Pascual Leone Horacio	Una Arquitectura basada en la Web Semántica para la Gestión de Versiones de Familias de Producto Gestión de Variabilidad en dos dimensiones : Temporal y Espacial
32	2015	Guamán Morocho Gabriela Paola, Martínez Pacheco John Carlos	Sistema Multiagente basado en un modelo Ontológico para la búsqueda de Objetos de Aprendizaje.
33	2016	Eduardo Ruiz de Pascual Núñez	Sistema de búsqueda de respuestas sobre Dbpedia
34	2016	Elena Oliván Salvador	Linked Map Service: Aplicación de estándares abiertos para la explotación transparente de datos geográficos y Linked Data
35	2014	María Dolores López Martínez	Sistema de recomendación de servicios médicos basado en ontologías y servicios de localización

Fuente: El autor
Elaboración: El autor

La presente investigación que se realizó sobre otros artículos o investigaciones realizadas tiene como fin el hacer uso de la metodología SLR, para la obtención de los mejores resultados, que nos permitan mostrar la relación con el desarrollo de un API para el

consumo y envió de datos RDF a la nube, el cual posee ciertas características como arquitectura u otras ideas que favorecen al desarrollo que vale tener en cuenta para obtener mejores resultados.

Aunque se identificaron diversos artículos de este proceso de búsqueda, se identificaron 35 estudios que muestran resultados relevantes. Además, se conoció algunos se están desarrollando. Otros estudios potencialmente relevantes que fueron excluidos por ser en años anteriores a los resultados esperados o que no presentaban un detalle más específico para lo requerido.

Las búsquedas fueron realizadas en consideración de ciertos criterios mencionados anteriormente, que nos permitieron conocer e identificar algunas posibles soluciones para el desarrollo del API, estos criterios son los siguientes:

- API basada en Open Linked Data.
- api rest web semántica.
- API rest web service y ontología.
- API rest web service y semántica.
- Desarrollo Api gestionar REST.
- Desarrollo api RDF.
- desarrollo de un api + RDF.
- Desarrollo para RDF.
- OWL

Los artículos considerados relevantes, permitieron crear la tabla 5 para realizar una comparación de los artículos presentados de la tabla 4, entre algunas características esenciales para la presente investigación, para determinar herramientas y analizar propuestas para realizar el API.

Tabla 5. Cuadro comparativo de trabajos relacionados

Recurso de trabajos relacionados	Elemento								
	Ontología	SPARQL	Framework semántico Jena	API REST	PHP	JSON	JavaScript	Android	Web service Android
[1]	x	x	x	x				x	x
[2]	x	x	x	x	x	x	x		
[3]	x			x	x	x	x		
[4]	x			x	x	x			
[5]	x			x					
[6]	x	x	x	x	x	x			
[7]	x	x	x	x		x			
[8]	x	x	x	x	x	x	x		
[9]	x	x	x	x		x	x		
[10]	x	x	x	x	x	x	x		
[11]	x	x		x	x	x	x		
[12]	x	x		x	x	x	x		
[13]	x			x					
[14]	x	x		x		x		X	
[15]	x	x	x	x	x	x	x	X	x
[16]	x	x	x	x		x	x	X	
[17]				x	x	x	x	x	
[18]	x	x	x	x	x		x		
[19]				x		x	x		
[20]				x		x	x		
[21]	x	x		x					
[22]	x	x	x			x			
[23]	x	x		x	x	x		x	x
[24]	x	x		x	x	x	x	x	
[25]	x	x	x	x					
[26]	x	x	x	x	x	x	x		
[27]	x			x					
[28]	x	x		x			x		
[29]	x	x		x		x	x		
[30]	x		x	x					
[31]	x	x	x	x		x	x		
[32]	x	x	x	x			x		
[33]	x	x		x					
[34]	x	x	x	x		x	x		
[35]	x	x	x	x		x	x	x	x

Fuente: El autor
Elaboración: El autor

1.4.3. Descripción de trabajos relacionados.

De acuerdo a los trabajos relacionados de la tabla 4 que han sido considerados, a continuación la relación que presenta con nuestra documentación en los aspectos más importantes para ser considerados de acuerdo con los elementos considerados en la tabla 5.

[1] Cómo introducir semántica en las aplicaciones SIG móviles: expectativas, teoría y realidad

La finalidad es almacenar y administrar información geográfica, la cual pueda ser mostrada y ejecutar otras acciones, la relación que presenta de acuerdo con la documentación seria que presenta una ontología y hace uso del Framework Jena, requiere de un API para gestionar la información y está enfocada en dispositivo móvil Android, el cual tiene integrado un web service para optimizar las actividades que necesita realizar.

[2] Trabajo de Fin de Grado Publicación de datos socio sanitarios : Una API basada en Open Linked Data

Se busca mostrar un API REST el cual permita realizar consultas sobre datos de múltiples fuentes sobre centros socio-sanitarios, para que pueda ser consumida por cualquier dispositivo o aplicación. La relación que presenta de acuerdo con la documentación, presenta una ontología y hace uso del Framework Jena, requiere de un API para gestionar la información y hace uso de lenguajes como PHP, JSON, JavaScript que son lenguajes que son aplicados para la solución pero está enfocado para cualquier dispositivo móvil no solo para dispositivos Android.

[3] Desarrollo de una API para la descripción y gestión de Servicios Web REST

Desarrollar una API para la gestionar las descripciones de servicios y otra API para gestión de servicios. Además, también se ha implementado una interfaz web para utilizar la API de monitorización para que sea más fácil de usar. En comparación hace uso de una ontología, desarrolla dos API's REST y los lenguajes empleados con los que fue desarrollada la solución son similares.

[4] Propuesta metodológica de evaluación de gestores de tesauros compatibles con la web semántica.

Se realizara un análisis sistemático de diversas herramientas para la gestión de tesauros, haciendo uso de diversos criterios que tienen en cuenta el propósito, las funcionalidades que debe tener la aplicación, el mantenimiento de la integridad o coherencia que debe presentar el vocabulario, así como otras cuestiones cruciales referidas a la interoperabilidad, todo esto teniendo muy en cuenta la compatibilidad con los estándares de la web semántica. La relación con la documentación se enfoca en el uso de una ontología, el desarrollo de un API REST, y lenguajes como PHP y JSON para la representación de los datos de la web semántica.

[5] Cómo Linked open data impactará en las bibliotecas a través de la innovación abierta

Determinar como Linked data puede llegar a considerarse como una apuesta tecnológica segura, y determinar a qué sectores de la información afecta. Esto se debe a que en las bibliotecas existe la participación de la publicación de datos haciendo uso de tecnologías RDF, que permitan el potenciar servicios innovadores desarrollados por terceros, reutilizando sus datos. La relación es hacer uso de una ontología y desarrollar un API para interactuar con los datos obtenidos por parte de las bibliotecas

[6] Información basado en Ontologías de Dominio Agrícola para las Ciencias Agropecuarias de la Universidad Central ¨Marta Abreu¨ de Las Villas

Crear un servicio que permita realizar búsqueda y recuperación de información basado en Ontologías, para el sector Agrícola dirigido a las Ciencias Agropecuarias de la Universidad Central “Marta Abreu” de las Villas. Con la finalidad de mejorar y ser más accesible a la información científica en el sector agrícola. Esta investigación tiene mucha relación con ya que hace uso de una ontología y de SPARQL Endpoint, de igual manera hace uso del framework Jena, y para la obtención de la información desarrolla un API con el uso de PHP Y JSON para luego si presentar la información de manera más entendible para quien requiera de la misma.

[7] Desarrollo de una API basada en Linked Data para el sector del Turismo de Tenerife

Desarrollar un API aplicando Linked Data para el sector turístico de Tenerife, e investigar tecnologías de la Web semántica 3.0. Para luego si permitir a cualquier usuario pueda acceder para obtener información turística del sector haciendo uso de fuentes tanto internas como externas. Comparando la documentación hace uso de una ontología, el framework Jena y como se mencionaba el desarrollo del API, con los cuales presenta datos en formato JSON que luego son categorizados para conocer la información turística.

[8] Memoria académica UCA LINKED DATA

Se pretende el ofrecer el método de realizar la publicación de datos entrelazados y el concepto de datos abiertos. De manera que teniendo una ontología generada con datos de la Universidad de Cádiz. Lo cual se genera una aplicación que permita no solo llegar a visualizar de manera gráfica los datos, sino también realizar la comparación de ficheros externos que sean de otras universidades. En comparación se tiene como se mencionó una ontología, además se hace uso del

Endpoint o realiza el desarrollo de un API que permite el consumir dicha información, los lenguajes son muy similares a los considerados para plantear una solución.

[9] Estudio y propuesta para enriquecimiento de información utilizando fuentes Open Data: Una experiencia con videos educativos multidominio

Se realizara un análisis de algunos trabajos que se consideren relevantes proporcionados por la UNED dentro del área educativo y tengan una relación con la anotación semántica de videos y tecnologías que tengan relación con Linked Open Data, con ese estudio es prevé crear entornos que posean información tanto para una propuesta metodológica como práctica. Lo que se pretende es la categorización de videos, de los cuales se pueda realizar extracción de información por cada video y brindar contenido que tenga relación a partir de fuentes externas Open Data. Para concluir con el análisis de dichos resultados y plantear líneas de investigación para un futuro. Existe mucha relación en cuanto a la implementación de una ontología y la creación de un API, considerando de igual manera el uso del Framework Jena y que va a trabajar tanto con JSON y JavaScript para solventar la problemática.

[10] Estudio de las posibilidades de los datos abiertos enlazados (Linked open data) para la realización de mashups de ámbito geográfico.

Mostrar como emplear datos enlazados del ámbito geográfico, los cuales han sido proporcionados de diversas fuentes, ya sea de servidores externos o de Endpoints SPARQL. La manera de aprovechar la información obtenida es la creación de una aplicación web la cual posee algunos módulos y opciones que permiten interactuar con algunas consultas que son realizadas a servidores, y así mismo esta aplicación presenta el entorno más agradable y fácil de usar. En comparación se puede mencionar que el parecido de acuerdo a los elementos de comparación es muy similar, a diferencia que la presente investigación está dirigida a una aplicación web.

[11] Aplicación del modelo de datos RDF en las colecciones digitales de bibliotecas, archivos y museos de España

Analiza la aplicación de un modelo RDF dirigido a las colecciones digitales españolas de materiales primordiales. Datos Abiertos y Datos Enlazados se introducen en el modelo. Al realizar una búsqueda por repositorios digitales, se obtienes registros RDF. En base a esto los resultados se describen en modelos EDM de Europeana y OntoWeb. En conclusión la aplicación de RDF es desigual ya que está condicionada hacia aplicaciones que convierten los registros en tripletas

RDF. En comparación hace uso de una ontología y del SPARQL Endpoint, de igual manera realizan la implementación de un API REST para trabajar sobre los datos y los lenguajes de programación para todo el desarrollo de la solución son similares.

[12] Mejora de un portal de Patrimonio Inmaterial mediante etiquetado semántico

Analizar herramientas semánticas para crear un portal de datos sobre patrimonio inmaterial. Se pretende extraer información relevante, una vez que los datos se hayan etiquetado como es el caso de la información semántica. Este etiquetado lo que permite es facilitar la búsqueda de contenido, y manejar grandes cantidades de datos. La comparación con la documentación considera una ontología y Endpoint SPARQL, para etiquetado y consumo el desarrollo de un API con lenguajes de programación similares.

[13] Propuesta de un modelo arquitectónico para un sistema recomendador turístico

Desarrollar un software que permita dar recomendaciones en base a datos del perfil de usuario y en base a un conjunto de preferencias previamente almacenadas. De manera que permita obtener mayor precisión hacia las necesidades de información se aplicó una técnica de filtrado para eliminar información considerada como no relevante. El software recomendador tiene relación comparando la documentación, al hacer uso de una tecnología y al implementar un API que permita optimizar el sistema recomendador, para poder tomar una buena decisión.

[14] Lenguaje de especificación para la delegación de tareas en servidores web mediante agentes

Presenta varios pasos para poder definir el lenguaje para poder integrar tareas a los servidores con el uso de agentes. Pero como conclusión de los pasos a seguir lo que se pretende es poder realizar la validación del lenguaje que ha sido especificado en Linked Data, el cuál al ser una plataforma homogénea y la semántica al ser parte de esta permite a los agentes a que procesen el contenido, para luego que puedan razonar sobre el mismo y finalmente realizar deducciones lógicas. Todo este proceso ha sido realizado en los Endpoint SPARQL con consultas propias, las cuales han sido expresadas en NautiLOD. La comparación se encuentra en el uso de una ontología y como se mencionó del Endpoint SPARQL, además se implementa un API que retorna los datos, siendo JSON uno de los lenguajes que muestra dichos resultados y está investigación si considera el uso de dispositivos Android.

[15] Aplicación Web de apoyo de emergencias para pacientes con enfermedades crónicas utilizando geolocalización, Web semántica y dispositivos móviles

Desarrollar una aplicación móvil como web, que permita obtener datos que se encuentran en un modelo semántico, el cual se basa tanto en relaciones de familiares y/o amigos de los pacientes. Adicional a ello se integró una ontología que dé respuesta a la búsqueda de hospitales especializados. Se integró un módulo para envío de alertas hacia los contactos del paciente como a los hospitales. Y finalmente la integración de Google Mapas para realizar el trazo de las rutas y ubicaciones. En comparación contiene todos los elementos comparativos, con lo que el presente documento será muy importante para obtener una mejor solución.

[16] Sistema de Administración de conocimiento por medio de dispositivos móviles Agora

El sistema de administración de conocimiento que se desarrollara permitirá a diversos usuarios el poder acceder a una base institucional dentro de un ambiente colaborativo. De igual manera además de que permita el realizar consultas, también tiene la opción de agregar contenido relevante. La necesidad de este sistema web es el facilitar la distribución de material educativo. El contenido estará clasificado en base a catálogo de perfiles, lo que brinda la posibilidad de que el usuario solo pueda visualizar aquello que le corresponde según el perfil que tenga. Comparando la documentación tiene la mayoría de los elementos comparativos a diferencia que no se usara el lenguaje de programación PHP.

[17] Estudio, análisis y desarrollo de una aplicación web híbrida

Realizar la integración de API's hacia una aplicaciones, la cual está relacionada con futbol, con lo que se pretende demostrar la integración y la interoperabilidad de varias interfaces de programación de aplicaciones. Se considera que estas API's permiten empaquetar diversas funciones y tareas que se realizan de manera independiente, las cuales pueden llegar a ser integradas hacia otras aplicaciones sin la necesidad de volver a crear. La comparación de esta documentación y su aporte está dirigido como se mencionó a los API's, además hace uso de los lenguajes de programación y al considerarse una aplicación hibrida se realiza el desarrollo para dispositivos Android

[18] Bases de Datos Semánticas

Analizar el concepto de base de datos semántica. Además mostrar un ejemplo donde se muestran diversas operaciones básicas que abarcan la gestión de la información que se encuentra almacenada en este tipo de base de datos. En

comparación se considera la mayoría de los elementos a diferencia de la implementación en dispositivos Android.

[19] Generación automática de servicios backend y sus bases de datos utilizando MDE

Mostrar la posibilidad de generar automáticamente la representación formal del servicio backend REST y además la estructura de la base de datos tomando como partida varios documentos JSON. En comparación con este artículo hay relación con la creación de un API REST y la implementación de JSON para conocer la estructura de la base de datos.

[20] NUBV (Necesito Unas Buenas Vacaciones) Sistema Big Data para la recomendación de hoteles

Desarrollar un sistema de recomendación con la finalidad de generar oportunidades comerciales, el cual hace uso de diversas herramientas Big Data. Lo cual se enfoca en búsqueda de datos de alojamientos que permita el captar nuevos clientes en base de un servicio personalizado. En este trabajo de grado la relación está en el desarrollo del API REST y el uso de JSON para extraer los datos.

[21] Web semántica y su aporte a la estrategia de datos abiertos del Estado Colombiano

Mostrar las ventajas que tiene la construcción de la web semántica, de manera que se apoye la estrategia estatal que busca la vinculación de datos que se tiene acceso libremente. Para esto se pretende establecer centros de investigación tanto para el análisis, como el diseño de redes semánticas que hagan uso de ontologías para así lograr obtener datos abiertos. El artículo hace define tanto ontologías, como el uso de un API para poder gestionar los datos abiertos que se comparan con la documentación para el desarrollo de un API.

[22] Principios y Tecnologías Linked Data para la publicación de Datos Académicos de las diferentes Carreras de la Universidad Nacional de Loja

Estructurar semánticamente la información que esté relacionada dentro de la Universidad Nacional en el tema de la oferta académica. La relación y el aporte de esta investigación se encuentran en la descripción detallada la creación de la ontología, del uso de SPARQL Endpoint para realizar pruebas de consulta y del uso del framework Jena para obtener resultados en formato JSON.

[23] Turimóvil Semántico (versión extendida): App de búsqueda turística semántica para el Centro Histórico de la Ciudad de México utilizando realidad aumentada.

Desarrollar una aplicación móvil para dispositivos Android en la cual se haga uso de la semántica, de manera que se logre un mejor desempeño al ser una extensión de una aplicación turística que anteriormente ya fue desarrollada, que tiene como objetivo el poder ayudar a los turistas en sus visitas al centro histórico de la ciudad de México. La aplicación pretende el navegar en un espacio abierto, el poder filtrar información y que la mismas sea fácil de interpretar como hacer uso de imágenes y/o texto, mostrar rutas haciendo uso de realidad aumentada o con el servicio de Google Maps y poner a consideración de los usuarios los itinerarios de los sitios que se puede visitar. Dentro de los elementos que permiten realizar la comparación están integrados todos en esta investigación a diferencia de que no hace uso del Framework Jena.

[24] Desarrollo de nueva funcionalidad e interfaz web y móvil para la API mappingapi2

Integrar una nueva funcionalidad para el API mappingapi2 que permita generar consultas para poder acceder a la información que está disponible en diversos sitios web y de manera heterogénea. La API base fue desarrollada por el grupo de informática biomédica, con esta nueva funcionalidad se pretende darle uso en el proyecto p-medicine, por esta razón se creó un web service RESTful, el cual permita generar consultas sin importar la plataforma, de manera que permita ser más accesible. Tiene mucha similitud al desarrollar un API para gestionar datos sobre una ontología y en los lenguajes que se desarrolló para trabajar en dispositivos móviles Android, pero no se implementó el Framework Jena.

[25] Enfoque semántico para el descubrimiento de recursos sensible al contexto sobre contenidos académicos estructurados con OAI-PMHS

El artículo describe un enfoque en el cual considera a los recursos estructurados con el Protocolo para Cosecha de Metadatos de la Iniciativa de Archivos Abiertos conocido como OAI-PMH por sus siglas en inglés, además una representación ontológica y el contexto que tiene el usuario como insumos de un framework para el desarrollo de aplicaciones que permitan la recuperación de información. En comparación, el artículo hace uso de una ontología y de SPARQL y de un API para trabajar sobre los datos.

[26] Integración semántica de datos geoespaciales utilizando una ontología de aplicación

Desarrollar un sistema web que haga uso de una ontología, para poder organizar e integrar información turística en base de fuentes de datos heterogéneas. Esta investigación tiene parecido con lo que se requiere hacer con el uso de ontologías y un API, pero en este caso no está centrado en un dispositivo móvil Android sino a un sitio web.

[27] Desarrollo de herramienta de generación de Linked Data Educativo

En base a un modelo que se desarrolló antes para mostrar información de estudios superiores con una red de ontologías, lo cual brindaba una definición común de algunos conceptos relevantes. Se pretende desarrollar una herramienta, la cual pueda generar datos educativos, la información extraída será de diferentes fuentes y permitirá la herramienta transformar los datos educativos en datos enlazados.

[28] Editor gráfico de ontologías para los lenguajes semánticos Rdf, Rdf(S) y Owl, como extensión del Framework Ontoconcept

Desarrollar un editor ontológico, para lo cual se considerara el estudio de algunas opciones presentes para crear o modificar ontologías, teniendo como resultado un análisis de procesos para la construcción de una ontología, Ontoconcept emplea características de arquitecturas que se basan en servicios, para luego continuar con el desarrollo del resto de componentes del framework. Como resultado se pretende reemplazar el editor actual, con un editor gráfico que presente una interfaz más amigable. Comparando con la investigación, esta hace uso de ontologías y del desarrollo de un API para la creación de un nuevo editor.

[29] Enriquecimiento automático de un Data Lake con metadatos

Crear una plataforma la cual tome como base la información proporcionada por un cliente, debido a que la iniciativa es que un cliente necesitaba una plataforma que pueda gestionar los datos que tenía y sacar el mejor partido de los mismos. Para lo cual se creó un repositorio para metadatos semántico, el cual contiene una ontología que contiene así mismo toda la información considerada como relevante para el cliente. Comparando este trabajo de titulación este también requiere una ontología y desarrolla un API para gestionar los datos del cliente.

[30] Método para la integración de ontologías en un sistema para la evaluación de créditos

Proponer un método para integrar ontologías en sistemas de manera que permita realizar una evaluación de los créditos con base de datos relacionales. El método que se pretende crear se basa en ciertas actividades, para operar sobre los datos.

Para realizar la evaluación del método se hizo uso de un sistema para la evaluación de créditos en ontologías, el cual está dirigido al Banco Nacional de Cuba. Comparando con el artículo muestra como desarrollo la ontología y como con el uso de Jena y el API pudieron conseguir resultados para evaluar los créditos.

[31] Una Arquitectura basada en la Web Semántica para la Gestión de Versiones de Familias de Producto

Mostrar el desarrollo de una arquitectura y como es implementada sobre una aplicación que será el prototipo. El modelo planteado comprende una arquitectura de 3 módulos que tienden a comunicarse entre sí y cumpliendo con funciones predeterminadas. La herramienta propuesta pretende gestionar la variabilidad del tiempo y espacio de la familia de productos sea cual sea su industria. El articulo al compararlo se identifica que hace uso de muchos de los elementos pero el desarrollo de las aplicaciones y herramientas no están pensadas para dispositivos Android.

[32] Sistema Multiagente basado en un modelo Ontológico para la búsqueda de Objetos de Aprendizaje.

Analizar, diseñar e implementar un sistema Multiagente que permita realizar búsquedas de objetos de aprendizaje, para la construcción del software se establece la web semántica operando en conjunto con agentes de software, los cuales se encargan de hacer uso de servicios web que estarán disponibles en la nube. El artículo de trabajo de titulación tiene relación al generar una ontología y desarrollar un API para gestionar los objetos de aprendizaje.

[33] Sistema de búsqueda de respuestas sobre Dbpedia

Crear un prototipo que emplee herramientas para Procesado de Lenguaje Natural, para que comprenda y pueda procesar preguntas abiertas y así estas puedan acceder de manera más flexible hacia bases de conocimientos en lo que se conoce como la web semántica, considerando particularmente a Dbpedia, para de ahí si poder obtener la respuesta a dichas preguntas. Este trabajo de grado en relación crea una ontología y hace uso de SPARQL Endpoint y para dar respuestas hace uso de un API.

[34] Linked Map Service: Aplicación de estándares abiertos para la explotación transparente de datos geográficos y Linked Data

Desarrollar un servicio el cual sea capaz de poder comunicarse con otros servicios de mapas externos, los cuales generen mapas referenciados espacialmente, esto a partir de información geográfica, todo esta información tiene la opción de lectura y

escritura en los almacenes de datos RDF. El trabajo tiene mucha similitud a diferencia que no está enfocado para dispositivos móviles.

[35] Sistema de recomendación de servicios médicos basado en ontologías y servicios de localización

Desarrollar una aplicación móvil, la cual teniendo como base un perfil y haciendo el uso de los servicios de ubicación permita al usuario tener recomendaciones de aquellos centros médicos que se encuentren más cerca a su posición. La implementación de la ontología es para describir los servicios que están integrados a un hospital y las especialidades que tiene cada uno. Este trabajo de titulación si se lo relaciona es el que más parecido tiene en cuanto a los elementos que se definieron, es por eso que este trabajo será un punto clave para la definición de la solución.

1.5. Aportes

El primer capítulo corresponde a la teoría necesaria para determinar tecnologías y herramientas que se pueden emplear para dar solución al desarrollo del API. Además permite conocer mediante los trabajos relacionados, cuales documentos presentan mayor relevancia para ser parte de la solución, es así como de acuerdo a la tabla 5 que es el cuadro comparativo de los trabajos relacionados, nos permite tomar en consideración el recurso [15] “Aplicación Web de apoyo de emergencias para pacientes con enfermedades crónicas utilizando geolocalización, Web semántica y dispositivos móviles”, como una buena fuente para trabajar ya que tiene cierta similitud en temas teóricos y posibles soluciones. Así como este resultado, existen otros documentos que presentan cierta relación al tema de la presente investigación, lo que permite tomar en consideración ciertas ideas.

CAPÍTULO 2: DISEÑO DE LA SOLUCIÓN

En este capítulo se describe la solución de acuerdo a las necesidades que se plantearon en base a los objetivos y las herramientas necesarias. Para ello se conoce que hay mucha información en ontologías que debe ser aprovechada por aplicativos Android, tanto para consumir o almacenar información. La información es exponencial y más aún si es liberada para diferentes dispositivos, para lo cual diversos usuarios puedan formar parte de estos datos enlazados y brindar su aporte.

Los datos que se almacenan en las ontologías puede ser variado según el área para la cual se use cada ontología, es por ello que se requiere desarrollar un API que pueda realizar la integración entre un dispositivo móvil o la web, hacia una ontología para así obtener un JSON que luego puede ser visto en base a la una petición que se realizó al API. Además de consumir se pueda guardar nueva información mediante POST al API desde el dispositivo o la web. Todo esto se realiza en el endpoint de virtuoso donde se realizaran las acciones configuradas en el API, el cuál aplica consultas dinámicas según la configuración se realice, estableciendo así cuales son los datos se almacenan o se buscan.

Para cumplir con todo esto el API es la parte primordial que permite realizar la conexión y consultas hacia virtuoso donde estarán los grafos necesarios para obtener todos los datos que se requiera.

2.1. Solución Propuesta

Para definir una solución es necesario conocer el objetivo que se pretende cumplir, para luego poder definir herramientas y diagramas que sirvan de base para definir una solución, hacia todas las necesidades que se requiere cumplir. La ontología a la cual se realizara las pruebas es una ontología turística.

2.1.1. Objetivo General.

Desarrollar una API que permita consumir y enviar datos RDF, desde una aplicación Android o web hacia un repositorio semántico en la nube.

2.1.2. Objetivos Específicos.

- Crear o buscar una ontología turística, para realizar pruebas de almacenamiento y consumo de datos.
- Desarrollar un API para enviar y consultar datos semánticos, el cual se pueda consumir desde un dispositivo móvil o la web.

- Mejorar el almacenamiento semántico desde tecnologías móviles o la web, generando una aplicación móvil y web, que se pueda integrar al API para consumir una ontología definida.

2.1.3. Descripción de la Solución.

Para el consumo y envió de datos desde el dispositivo móvil, se enviara una petición al API en la cual se tiene las consultas dinámicas, que se configuraran en un panel de administración tanto para consultas SELECT O INSERT, de lo cual se obtendrá el resultado en formato JSON, el cual será fácilmente leído en el dispositivo y listado en el caso de consulta, y para ingresar se generara un formulario para poder ingresar nuevos datos. En la figura 6 se expresa el esquema que se mantiene para dar solución en petición y respuesta entre el dispositivo y el API.

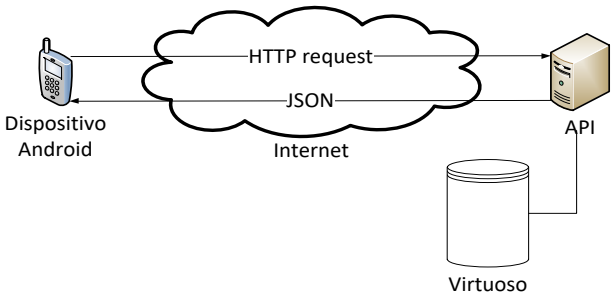

Figura 6. Esquema de la solución
Fuente: El autor
Elaboración: El autor

2.2. Virtuoso universal server

Virtuoso Universal Server versión 7.20 es la cual se ha elegido para realizar la conexión con el endpoint, de igual manera la integración se puede realizar con otra versión disponible mientras se pueda acceder al endpoint. Esta versión es de código abierto conocida como Open Link Virtuoso. Una de las diferencias entre la versión comercial y la versión de código abierto, se encuentra en que la edición de código abierto no posee el motor de base virtual y replicación de datos.

La base de datos virtual permite realizar búsqueda de diversas bases de datos, haciendo uso de una sola consulta. Además permite la gestión de datos, el acceso y la integración.

2.2.1. Virtuoso endpoint.

El endpoint de Virtuoso permite realizar las consultas necesarias como son SELECT e INSERT, al endpoint se accede mediante la dirección http://localhost:8890/sparql, la figura 7 representa la interfaz en la cual se realiza las peticiones.

Figura 7. Interfaz de endpoint virtuoso
Fuente: El autor
Elaboración: El autor

Las consultas se realizaran en SPARQL, y la respuesta del endpoint puede devolverse en diferentes formatos como se puede ver en la figura 8. Se planteó que en caso de respuesta el formato será JSON para poder leer en la aplicación.

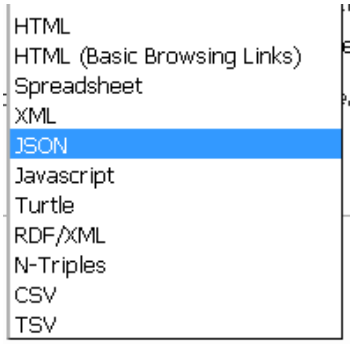

Figura 8. Formatos de respuesta de endpoint virtuoso
Fuente: El autor
Elaboración: El autor

2.3. Aplicación Móvil Android

Ya que se requiere que la aplicación se integre a un API, se necesita que tenga permiso a internet para poder enviar las peticiones desde el dispositivo móvil. Y en la aplicación se deberá especificar la ubicación del API, además se deberá generar formulario para envió de datos y otra vista para visualizar los elementos que se soliciten al API.

2.3.1. Arquitectura de la aplicación.

La arquitectura de la aplicación se puede ver en la figura 9, donde se tiene una idea más clara de la solución planteada en la aplicación.

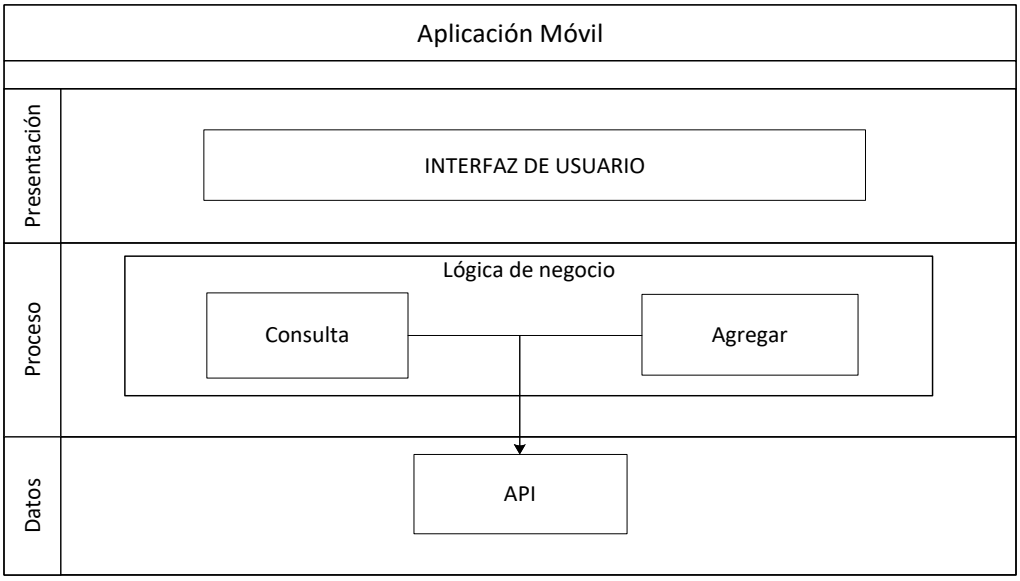

Figura 9. Arquitectura de la aplicación móvil
Fuente: El autor
Elaboración: El autor

Tomando como base la arquitectura, se puede definir los módulos de la aplicación.

2.3.2. Módulo de consulta.

El módulo de consulta permite mostrar todos los datos, que han sido ingresados en la ontología que previamente se ha subido a virtuoso, en una lista para que sea fácil de ver los campos ingresados bajo una consulta predefinida en el API la figura 10 muestra el diagrama de cómo funciona este módulo.

Lo primero es que los datos estén ingresados en virtuoso, para luego si poder hacer una petición al API, el cual retorna los valores en formato JSON, cuyos datos de respuesta pasan a listarse en la aplicación, esta acción se realiza siempre que se tenga una conexión a internet, en caso contrario se emite un mensaje de fallo de conexión para que se verifique que esté conectado ya sea a WIFI o por medio de datos.

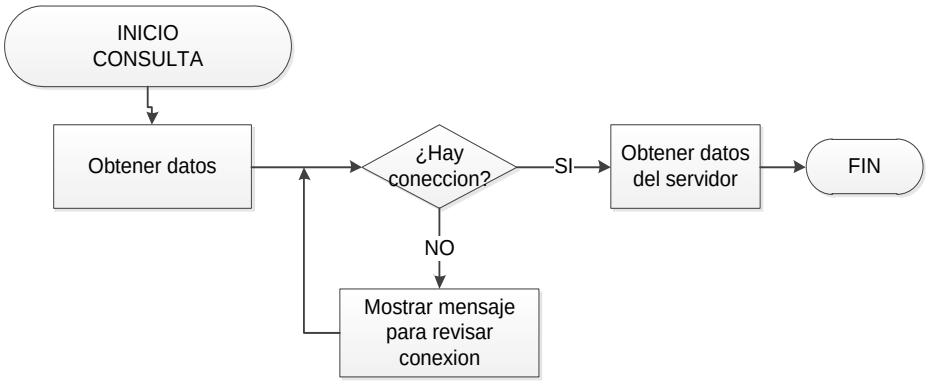

Figura 10. Diagrama de flujo de módulo de consultas
Fuente: El autor
Elaboración: El autor

2.3.3. Módulo de Agregar.

El módulo de agregar permite ingresar datos especificados previamente entre la aplicación y el API, en la figura 11 se muestra el diagrama para guardar, los datos que se guardaran se cargaran en un formulario.

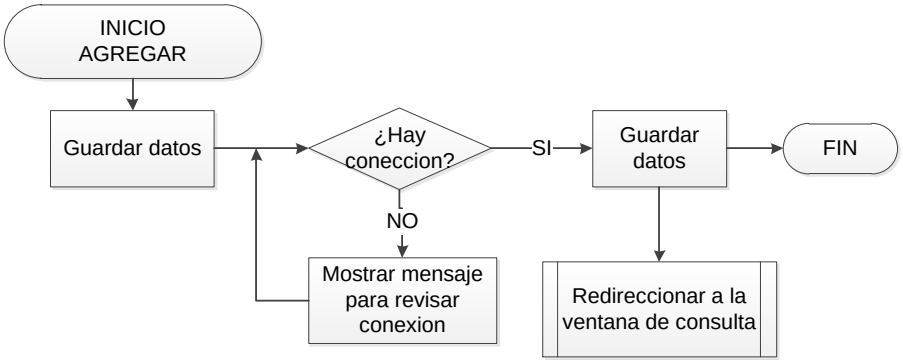

Figura 11. Diagrama de flujo del módulo de agregar
Fuente: El autor
Elaboración: El autor

Para guardar datos se ingresara a un formulario donde los campos sean los definidos en el API, luego se enviara los datos mediante POST al API, que en caso de no tener conexión a internet nos solicitara que verifiquemos dicha conexión, y en caso de tener la conexión se almacenara y se cargara la vista del módulo de consultas, en donde se visualizara el nuevo campo que ha sido ingresado.

2.4. API

El API se encarga de receptar la petición para consultar o guardar los datos que se requiera, a la ontología que está definida y se encuentre en virtuoso.

2.4.1. Arquitectura del API.

La arquitectura del API se puede ver en la figura 12, donde se puede ver como se integra la librería sparqllib con los módulos que aceptan las peticiones, y luego se realiza consultas dinámicas en base a la configuración del API, sobre el repositorio Virtuoso. Mostrando además de las peticiones como es la repuesta en cada caso del consumo del API.

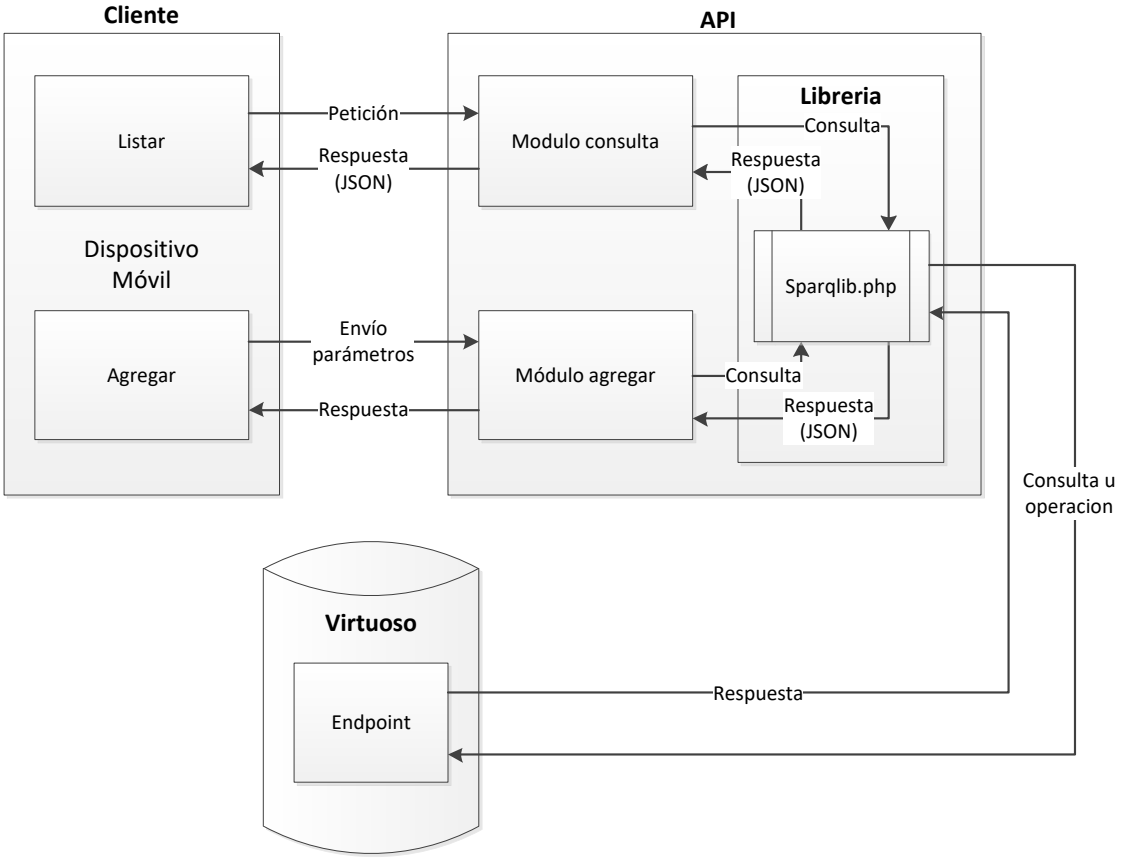

Figura 12. Arquitectura del API
Fuente: El autor
Elaboración: El autor

Tomando como base la arquitectura, se puede definir los módulos que envían las consultas al endpoint de virtuoso.

2.4.2. Librería sparqllib.

Para generar el API y poder realizar las consultas SPARQL usando PHP, se optó por considerar una librería gratuita que se llama sparqllib.php, la cual ya dispone de diversas funciones que se puede reutilizar para la conexión y especificar por cada módulo que tipo de consulta se va a realizar. Esta librería es la que se conecta con el endpoint de virtuoso.

2.4.3. Módulo de consulta.

El módulo de consulta permite retornar todos los datos en formato JSON, que han sido ingresados en la ontología que previamente se ha subido a virtuoso, la figura 13 presenta el diagrama de cómo funciona este módulo.

De acuerdo a los datos ingresados en virtuoso, al llamar a la función que retorne los datos en formato JSON, si no hay datos retornara la respuesta 0 para que muestre una alerta, y en caso de existir datos los retornara en formato JSON y la respuesta 1 para que se listen luego en la aplicación. El módulo de consulta toma como referencia la librería sparqllib.php para la conexión, y en esta función se define la consulta que se va a realizar.

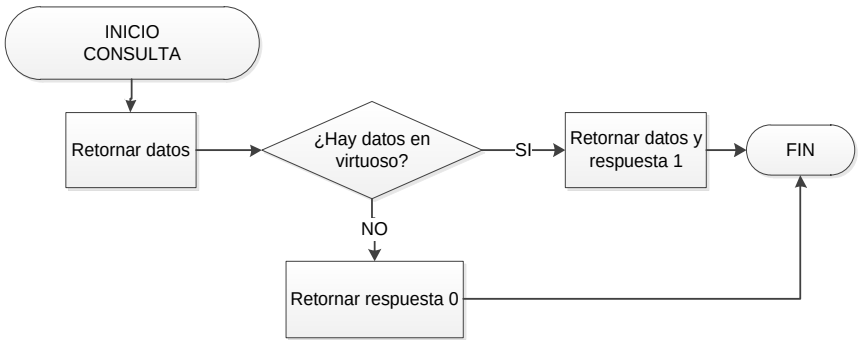

Figura 13. Diagrama de flujo del módulo de consultas API
Fuente: El autor
Elaboración: El autor

2.4.4. Módulo de Agregar.

El módulo de agregar permite Inserta datos especificados previamente entre la aplicación, en la figura 14 corresponde al diagrama para guardar, los datos que se guardaran se cargaran en un formulario.

Figura 14. Diagrama de flujo del módulo de agregar API
Fuente: El autor
Elaboración: El autor

Primero se leen los datos enviados desde la aplicación mediante POST, y luego se ejecuta la consulta para almacenar los datos. En caso de que no se ingrese bien el grafo que se encuentra en virtuoso se debe responder a la aplicación que existe un problema en el API, pero si no existe problema se debe ejecutar normal la consulta y almacenar los datos.

2.5. Ontología

La ontología que se puede usar en el API, puede ser cualquiera ya que el API realiza las consultas en el endpoint, gracias a la librería que permite conectarse a cualquier ontología. La ontología se cargara a virtuoso previo a esto si se deberá validar que los datos estén ingresados correctamente para luego poder realizar las consultas sin tener ningún inconveniente.

2.6. Alcance de la solución

La solución para el desarrollo de API y de la aplicación se muestra en la arquitectura de la figura 15, la cual es un resumen de la figura 12 donde se muestra cómo se plantea una solución para la integración de sus partes.

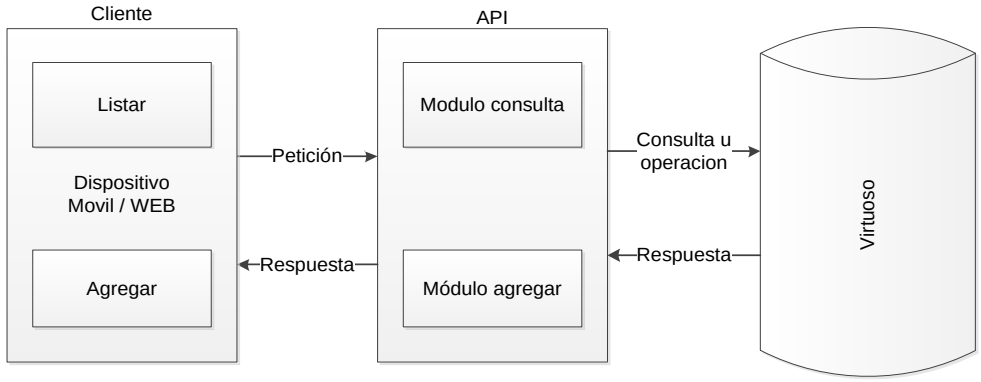

Figura 15. Arquitectura de la aplicación y API integrados.
Fuente: El autor
Elaboración: El autor

El alcance está definido por la aplicación móvil Android y el API. Además en la solución se adicionara una aplicación web desarrollada en PHP y JavaScript, la cual tendrá todas las características del aplicativo móvil con la integración del API.

2.6.1. Aplicación Móvil Android.

Desarrollada para dispositivos con el sistema operativo Android desde la versión 4.0 (Ice Cream Sándwich) y probada hasta la versión 5.1 (Lollipop).

En la aplicación se configura las url para hacer la petición de consulta y otra url para enviar datos mediante POST.

No se contempla realizar pruebas de compatibilidad de la aplicación móvil en la versión 7 (Android N) debido a que dicha versión (a la fecha de realización de esta aplicación) aún no se encuentra oficialmente disponible por los fabricantes en los dispositivos que se dispone para la realización de pruebas de funcionamiento.

Dispositivos deben tener conexión a internet ya sea mediante WIFI o datos, para poder conectar al servidor donde se encuentre el API.

No se contempla la distribución de la aplicación en la tienda de aplicaciones oficial de Android (Google Play).

2.6.2. API.

Desarrollada bajo el lenguaje PHP, y para las respuestas del API con el lenguaje JSON, para envió de respuestas e información solicitada. Implementación de la librería gratuita sparqllib.php desarrollada en PHP por Christopher Gutteridge & University of Southampton. El servidor es apache y extensiones habilitadas para soportar el intérprete de lenguaje PHP.

La administración de las ontologías se realizara en Virtuoso. Y la integración con el API se dará mediante el endpoint que se integre con la librería mencionada para luego poder devolver resultados a la aplicación.

El API es configurable para realizar consultas SPARQL dinámicas, tanto para consultar y agregar datos, de manera que exista un entorno de configuración que permite dinamizar así el uso del API para cualquier ontología. Las consultas se generan a partir de los atributos que se almacenan en una base a datos, la figura 16 muestra los campos disponibles para realizar luego las consultas dinámicas, los cuales pueden ser modificados.

atributos

	Field	Type	Extra
P	ATR_ID	int(11)	Auto Increment
	ATR_NOMBRE	varchar(50)	
	ATR_COMPLETO	varchar(200)	
	ATR_URI	varchar(150)	
	ATR_PREFIJO	varchar(25)	Allow Null
	ATR_CONSULTA	varchar(3)	Allow Null
	ATR_INGRESAR	varchar(3)	Allow Null

clase

	Field	Type	Extra
P	CLA_ID	int(11)	Auto Increment
	CLA_NOMBRE	varchar(150)	

grafo

Field	Type	Extra
GRAFO	varchar(500)	

Figura 16. Base de datos de configuración del API.
Fuente: El autor
Elaboración: El autor

2.6.3. Aplicación Web.

Desarrollada bajo el lenguaje PHP y JavaScript, que tiene las mismas características de consulta y almacenamiento de datos, presentes en el aplicativo móvil. El servidor es apache y extensiones proporcionadas para soportar el intérprete de lenguaje PHP.

Se requiere conexión a internet para poder realizar la integración con el API, al igual que en la aplicación móvil se configura las url de la ubicación del API para hacer la petición de consulta y otra url para enviar los parámetros mediante POST.

2.7. Metodología de Desarrollo

El desarrollo se llevará tomando como base una metodología ágil, por lo que se usarán algunos principios de la metodología XP (Extreme Programming).

2.7.1. Extreme Programming XP.

XP es una nueva metodología de desarrollo: ligera, pero eficiente considerando riesgos, flexible, predecible, enfocada en el desarrollo y orientada a muchos cambios.

La forma de trabajar con XP es empezar por recoger las historias de usuario para poder así estimar los requerimientos, luego se codifica, se generan paquetes de software que son probados y finalmente integrados ya al sistema final. El camino que lleva XP es lograr una planificación incremental y la retroalimentación continua de los ciclos cortos del desarrollo, con lo que se logra un proceso evolutivo. Un atributo importante también es que permite la Comunicación constante, lo que permite detectar errores antes de integrar ya al sistema final.

2.7.2. Fases de XP.

XP posee 4 fases establecidas, las cuales se muestran en la figura 17.

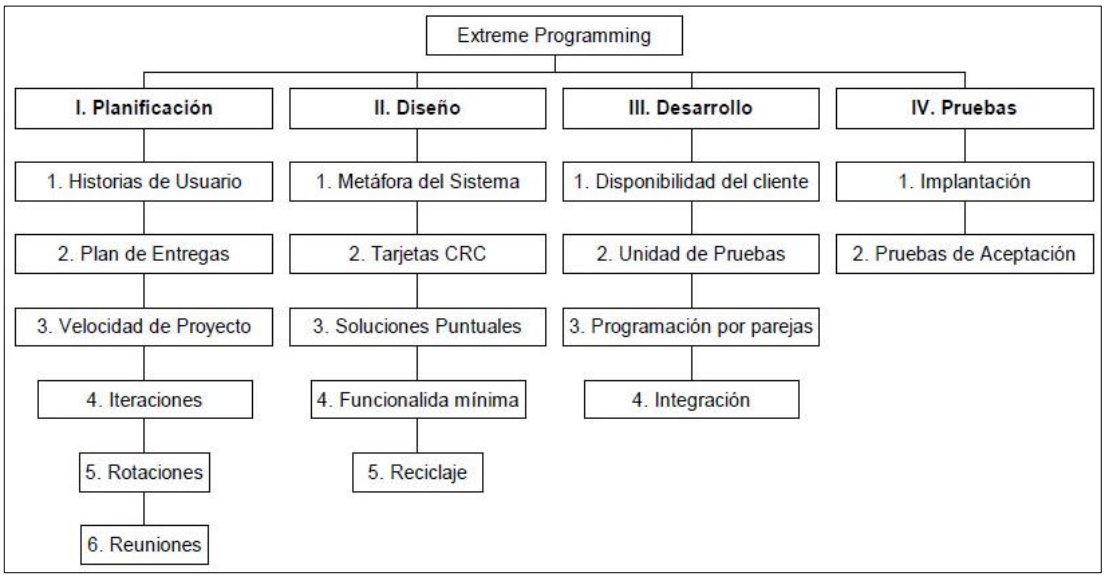

Figura 17. Fases de la metodología XP.
Fuente: Recuperado de http://goo.gl/em7QwN
Elaboración: Recuperado de http://goo.gl/em7QwN

- **Planificación:** Es la fase en la cual se realizan reuniones entre el cliente y los desarrolladores, aquí es donde se definen los requerimientos y son documentadas como “historias de usuario”. El resultado de esta fase es la visión general del sistema y la prioridad que se le da a cada historia de usuario.
- **Diseño:** Se define una solución del sistema, estableciendo la funcionalidad mínima y cómo el sistema debe comportarse, lo que sería un diseño simple.
- **Desarrollo:** Comprende la codificación del sistema de forma iterativa, se genera al final de cada iteración un entregable que sea funcional, el cual muestra el resultado de una o más historias de usuario. La integración constante en esta fase es una parte fundamental.
- **Pruebas:** Se valida la integridad y funcionalidad del sistema, aplicando pruebas unitarias y de aceptación.

2.7.3. Plan de Entregas.

Teniendo en cuenta las fases de XP, podemos definir un plan de entregas en base a iteraciones de la siguiente forma:

Tabla 6. Plan de iteraciones

Módulo	Iteración	Actividad	Tipo
Consulta	1	Conexión para obtener respuesta JSON del API.	App. Móvil / App. Web
	2	Mostrar los resultados.	
Agregar	1	Conexión para envió datos mediante POST.	App. Móvil / App. Web
	2	Espera de respuesta para mostrar mensaje o actualizar vista.	
Consulta	1	Establecer conexión a endpoint virtuoso.	API
	2	Generar la consulta.	
	3	Definir retornar respuesta y/o datos.	
Agregar	1	Establecer conexión a endpoint virtuoso.	API
	2	Definir consulta para guardar daros	
	3	Definir retornar respuesta y/o datos.	

Fuente: El autor
Elaboración: El autor

2.7.4. Fases pruebas e implantación.

Las pruebas e implementación de nuevas ontologías al API se realizaran luego de concluir con el desarrollo para así determinar que se pueda integrar cualquier ontología.

Para realizar las pruebas primero se consideran los siguientes parámetros:

- Ontología: La ontología este correcta.
- Dispositivo móvil: Cuente con conexión a internet.
- API: Se pueda conectarse al aplicativo móvil.

Se realizaran pruebas de validación y aceptación. Las pruebas de validación corresponden tanto al funcionamiento de la aplicación y del API. Las pruebas de aceptación se generaran a partir de los entregables elaborados durante los procesos de iteración y las dos pruebas de validación realizadas tanto para el API como el aplicativo móvil, teniendo como prueba la integración de las mismas.

2.8. Herramientas de desarrollo

Para el desarrollo se ha contemplado el uso del sistema operativo Windows 10 con los siguientes recursos:

Tabla 7. Herramientas de desarrollo

Herramienta	Característica
Android Studio	Entorno de desarrollo enfocado en el desarrollo de aplicaciones Android, brinda funcionalidades para codificación y pruebas.
Android SDK	Kit de desarrollo de Google para la plataforma de Android.
Android Development Tool (ADT) Plugin para Eclipse	Complemento para el IDE Eclipse, que permite hacer uso de las librerías y herramientas del Android SDK.
Virtuoso	Herramienta para administración de ontologías
Protege	Editor de Ontologías
Sublime Text 2	Editor de código.
Navicat Premium	Administrador de bases de datos de múltiples conexiones
XAMPP	Paquete de instalación independiente de plataforma, que permite la gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl.

Fuente: El autor
Elaboración: El autor

2.9. Diseño de la ontología

Antes del desarrollo del API y de la aplicación, es necesario tener la ontología con los datos preparados, para poder realizar las consultas necesarias. Por lo cual para la creación de una ontología turística, se requiere información para poder realizar pruebas de consumo, para ello se empleó una base de datos, la figura 18 es un ejemplo de una de las tablas.

CIU_ID	PAIS_ID	CIU_NOMBRE	CIU_LATITUD	CIU_LONGITUD	CIU_DESCRIPCION
1	1	aixas	42.4833333	1.4666667	(Null)
2	1	aixirivali	42.4666667	1.5	(Null)
5	1	aixovall	42.4666667	1.4833333	(Null)
6	1	andorra	42.5	1.5166667	(Null)
12	1	ansalonga	42.5666667	1.5166667	(Null)
13	1	anyos	42.5333333	1.5333333	(Null)
14	1	arans	42.5833333	1.5166667	(Null)
15	1	arinsal	42.5666667	1.4833333	(Null)
16	1	aubinya	42.45	1.5	(Null)
21	1	canillo	42.5666667	1.6	(Null)
22	1	casas vila	42.5333333	1.5666667	(Null)

Figura 18. Base de datos base para construir ontología.
Fuente: El autor
Elaboración: El autor

2.9.1. Protege.

Para la creación de la ontología se debe especificar la IRI, luego establecer los prefijos, para que una vez ingresados si poder ingresar las clases así como se puede ver en la figura 19,

para ello se debe ingresar el prefijo y el nombre luego cada vez que se crea una nueva clase, de igual manera luego se ingresan los Data Properties, y finalmente los Objects Properties que son los enlaces entre clases.

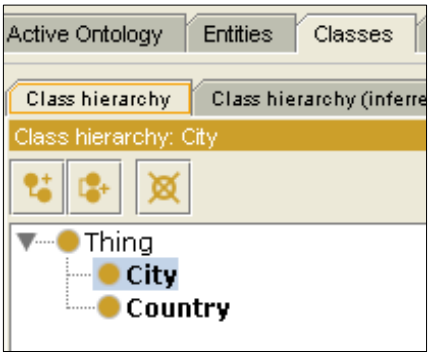

Figura 19. Creación de clases en Protege.
Fuente: El autor
Elaboración: El autor

Para poder integrar todo esto es necesario establecer los Dominios y Rangos de las propiedades. Finalmente se obtiene así un archivo "*.owl" el cual será validado en primer lugar para luego poder cargar datos al mismo.

2.9.2. Validación.

Para poder validar la ontología turística generada, se requiere un validador el cual puede ser "https://www.w3.org/RDF/Validator/rdfval", en donde existe una ventana que nos permite ingresar la ontología genera, para lo cual copiamos el archivo generado en la entrada y ejecutamos la validación.

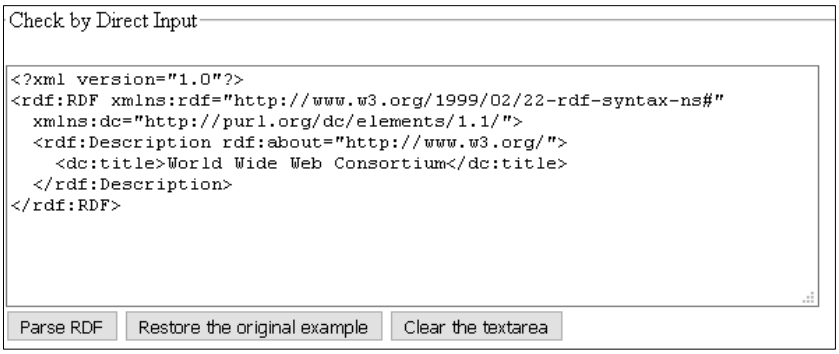

Figura 20. Ejemplo de ingreso de RDF
Fuente: El autor
Elaboración: El autor

Una vez ejecutado el RDF que corresponde al lenguaje que permite especificar metadatos, esperamos la respuesta del sitio, la figura 21 es el resultado de la ontología turística. Con lo cual se estable que la ontología puede subirse a virtuoso o se puede cargar los datos previamente, sabiendo que esta esta correcta.

Figura 21. Resultado de validador RDF
Fuente: El autor
Elaboración: El autor

2.9.3. Carga de datos.

Para cargar datos se requiere la implementación de un aplicativo que permita la carga de datos a la ontología, el cual toma los valores seleccionados de la base de datos, para ser ingresados en la ontología en cada una de las propiedades donde se configure. Una vez cargados los datos se puede realizar otra validación para poder definir que no exista ningún problema para poder así consumir o ingresar información en la ontología.

2.9.4. Subida a virtuoso de ontología.

Para acceder a Virtuoso se debe ingresar el usuario y clave como se puede ver en la figura 22.

Figura 22. Acceso a Virtuoso
Fuente: El autor
Elaboración: El autor

En la pestaña “Linked Data”, se debe ingresar a “Quad Store Upload” en donde se subirá el archivo de la ontología, la interfaz se muestra en la figura 23, donde se debe especificar la IRI donde se va a establecer luego la conexión con el API.

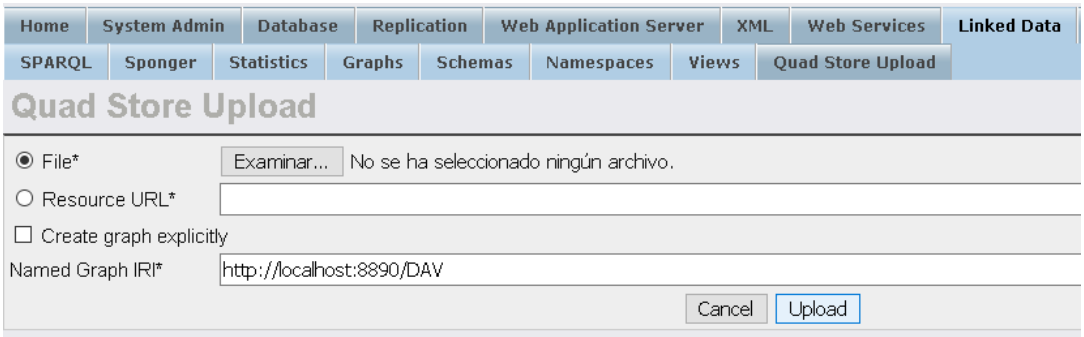

Figura 23. Cargar ontología a virtuoso
Fuente: El autor
Elaboración: El autor

Para verificar si la ontología esta subida en virtuoso, ingresamos en la pestaña “Graphs” donde se muestran todos los grafos ingresados con su ontología hasta el momento como se puede ver en la figura 24, este grafo será necesario luego en el API para poder seleccionar la ontología sobre la cual se va a trabajar.

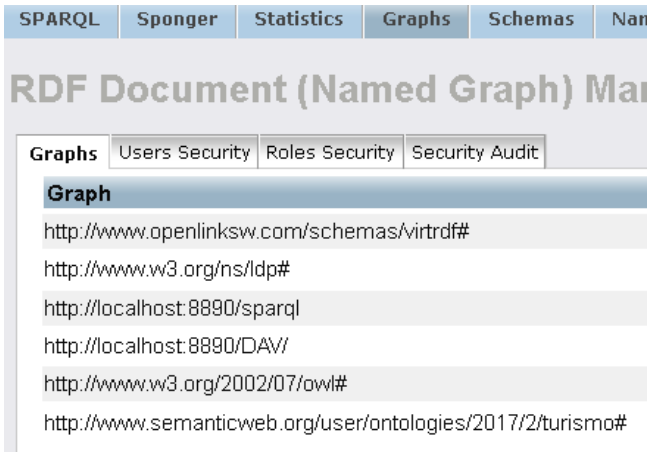

Figura 24. Visualizador de grafos en virtuoso
Fuente: El autor
Elaboración: El autor

CAPÍTULO 3: DESARROLLO

En este capítulo se muestra el desarrollo tanto de la aplicación móvil y web; como del API. Tomando como referencia el diseño de la solución que se planteó en el capítulo anterior. El desarrollo está identificado de acuerdo a los módulos de la aplicación y el API, y a partir de los módulos se determina cada una de las iteraciones. La configuración de la librería sparqllib.php que es consumida por el API también se describe para conocer cómo realizar la conexión con el endpoint de virtuoso.

3.1. Aplicación móvil en Android Studio

De acuerdo al plan de entregas planteado, el desarrollo de la aplicación se establece en 4 iteraciones, los cuales son explicados en base a los módulos que comprenden a la aplicación móvil, se debe tomar en cuenta que así como se muestra en la tabla 6 del capítulo anterior, algunos de los módulos se llevaran a cabo en paralelo.

La aplicación móvil se llevara a cabo en Android estudio. Para lo cual se inicia creando un nuevo proyecto que en este caso toma el nombre de ApiApp, la figura 25 muestra donde crear el nuevo proyecto en Android Studio, para esto previamente se debe instalar el plugin Android Developer Tool (ADT), el SDK de Android, generalmente Android Studio suele venir ya con todos estos plugins, pero se puede agregar otros SDK o ADT que ya estén configurados.

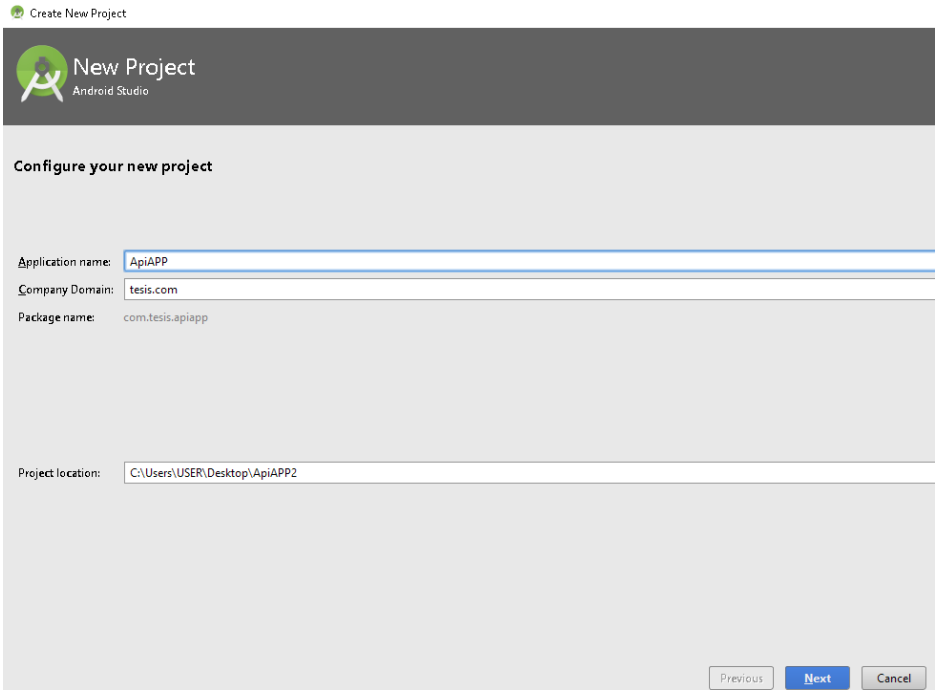

Figura 25. Creación de un proyecto de Android Studio.
Fuente: El autor
Elaboración: El autor

En el SDK, se debe tener alguna de las versiones sobre la cual se va a trabajar. La figura 26 muestra las versiones disponibles, para lo cual se debe seleccionar y descargar la necesaria, en el caso de la aplicación ApiApp la versión seleccionada es Android 4.2 (Jelly Bean).

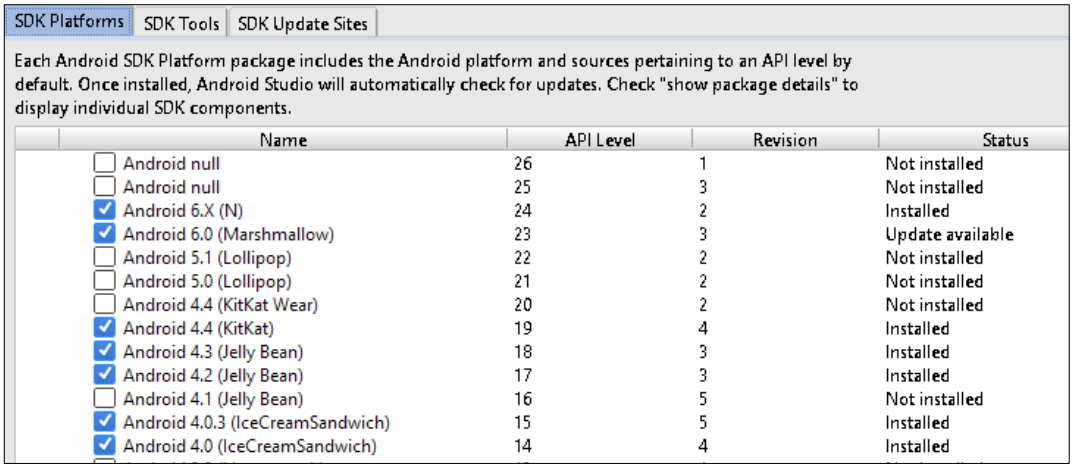

Figura 26. Versiones SDK.
Fuente: El autor
Elaboración: El autor

Por defecto, el ADT y el SDK generan la estructura necesaria para el desarrollo de la aplicación, en la figura 27 se puede ver la estructura, la cual separa algunos de los archivos necesarios como son la vista, configuración de la aplicación y las clases que se conectan al API y lleva los resultados a la vista.

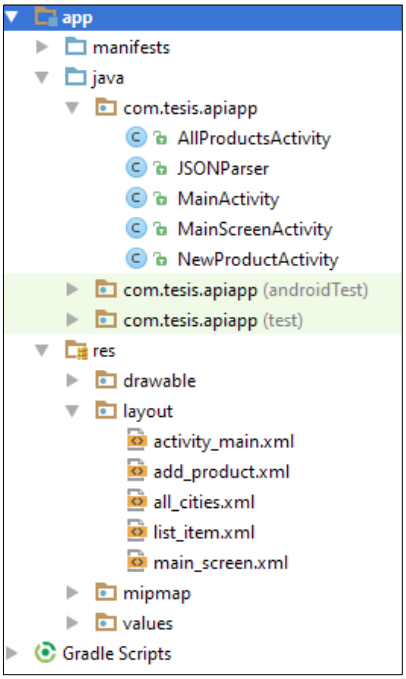

Figura 27. Estructura de archivos del proyecto Android.
Fuente: El autor
Elaboración: El autor

3.2. Módulo de consulta

Para el módulo de consulta se crea la clase y vista que permite mostrar todos los datos ingresados en la ontología.

3.2.1. Iteración 1 - Clase AllCitiesActivity.java.

Esta clase permite obtener desde el API todos los datos de la ontología que están cargados en Virtuoso. Lo primero es establecer un cuadro de dialogo para los mensajes, inicializar un objeto que nos permita manejar los datos de respuesta del API, luego indicar lista que contendrá dichos elementos y finalmente un JSONArray donde se cargaran los datos como un arreglo, la figura 28 expresa la inicialización de los elementos.

```
1.  // Progress Dialog
2.  private ProgressDialog pDialog;
3.
4.  // Creating JSON Parser object
5.  JSONParser jParser = new JSONParser();
6.
7.  ArrayList<HashMap<String, String>> productsList;
8.
9.  // products JSONArray
10. JSONArray products = null;
```

Figura 28. Declaración de parámetros.
Fuente: El autor
Elaboración: El autor

Luego se define la url donde se encuentra el API, de donde se obtendrán los datos, la figura 29 es un ejemplo de la URL donde se realizara la petición. La acción de petición se realiza en la siguiente iteración ya que está dentro de las actividades por debajo de los mensajes de ejecución del proceso de consulta.

```
1. private static String url_all_products =
   "http://192.168.1.7/ontoAPI/core/examples/basic.php";
```

Figura 29. URL del API para consulta.
Fuente: El autor
Elaboración: El autor

La función onCreate hace un llamado a la vista donde se va a mostrar los datos, y se inicializa la lista donde se mostraran los resultados. Luego de inicializar se ejecuta la función donde los datos se cargaran en la lista para luego ser visualizada, figura 30.

```
1. public void onCreate(Bundle savedInstanceState) {
2.   super.onCreate(savedInstanceState);
3.   setContentView(R.layout.all_cities);
4.   // Hashmap for ListView
5.   productsList = new ArrayList<HashMap<String, String>>();
6.   // Loading products in Background Thread
7.   new LoadAllProducts().execute();
8.   // Get listview
9.   ListView lv = getListView();
10. }
```

Figura 30. Llamado a la vista listar y a funciones principales.
Fuente: El autor
Elaboración: El autor

3.2.2. Iteración 2 - Clase AllCitiesActivity.java.

Dentro de la función LoadAllProducts, lo primero es cargar un mensaje previo mientras se cargan los datos, para esto la figura 31 muestra la carga de mensaje que se está ejecutando operaciones por debajo.

```
1. protected void onPreExecute() {
2.   super.onPreExecute();
3.   pDialog = new ProgressDialog(AllProductsActivity.this);
4.   pDialog.setMessage("Cargando ciudades. Espere...");
5.   pDialog.setIndeterminate(false);
6.   pDialog.setCancelable(false);
7.   pDialog.show();
8. }
```

Figura 31. Mensaje de carga de datos.
Fuente: El autor
Elaboración: El autor

Finalmente se ejecuta la función de los procesos principales, dentro del cual se hace la petición a la URL que se inicializo donde está ubicado el API, los datos obtenidos están en formato JSON. La figura 32 corresponde a la función principal de ejecución que se ejecuta por debajo. Los datos pasan al arreglo donde se va separando para mostrarse en la lista que pasa a la vista.

```
protected String doInBackground(String...args) {
 // Building Parameters
 List < NameValuePair > params = new ArrayList < NameValuePair > ();
 // getting JSON string from URL
 JSONObject json = jParser.makeHttpRequest(url_all_products, "GET", params);
  try {
  products = json.getJSONArray(TAG_PRODUCTS);
  // looping through All Products
  for (int i = 0; i < products.length(); i++) {
      JSONObject c = products.getJSONObject(i);
      JSONObject n = c.getJSONObject("p");
      // Storing each json item in variable
      String id = n.getString(TAG_PID);
      String name = n.getString(TAG_NAME);
      // creating new HashMap
      HashMap < String, String > map = new HashMap < String, String > ();
      map.put(TAG_PID, id);
      map.put(TAG_NAME, name);
      productsList.add(map);
     }
    } catch (JSONException e) {
     e.printStackTrace();
    }
     return null;
   }
```

Figura 32. Función de llamado al API para mostrar datos.
Fuente: El autor
Elaboración: El autor

El resultado se puede ver en la figura 33, donde se muestra una lista de las ciudades ingresadas en la ontología.

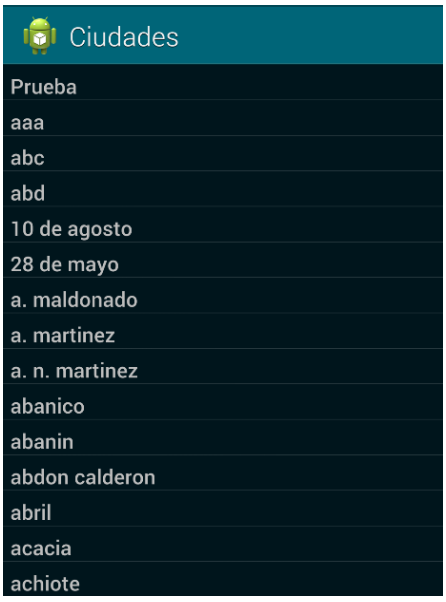

Figura 33. Vista del módulo consultar.
Fuente: El autor
Elaboración: El autor

3.3. Módulo de agregar

Este módulo permite ingresar nuevos datos, los cuales serán enviados desde la aplicación a la ontología mediante un API.

3.3.1. Iteración 1 – NewCityActivity.java.

Esta clase permite crear una nueva ciudad, para lo cual estos datos se enviaran al API para luego ser guardados en la ontología en Virtuoso. Dentro de esta clase lo primero es inicializar un cuadro de dialogo para los mensajes, como es el caso en la figura 34.

```
1. // Progress Dialog
2. private ProgressDialog pDialog;
3.
4. JSONParser jsonParser = new JSONParser();
```

Figura 34. Inicializar parámetros para agregar.
Fuente: El autor
Elaboración: El autor

Luego se especifica la url donde se consumirán los atributos de entrada para luego generarlos de forma dinámica y la url donde se encuentra el API. La url de API es en donde se enviaran los datos para almacenarse, la figura 35 es un ejemplo de donde se consumen los atributos y donde se realizan el almacenamiento de los datos.

```
1. //atributos para generar entrada de datos
2. private static String url =
   "http://192.168.1.7/ontoAPI/core/APP/consulta_bd.php"
   ;
3. // url para crear nuevas ciudades
4. private static String url_create_product =
   "http://192.168.1.7/ontoAPI/core/examples/create.php"
```

Figura 35. URLs para conectar con API modulo agregar.
Fuente: El autor
Elaboración: El autor

Se hace un llamado a la función onCreate, esta función llama a la vista donde se ingresaran los datos que se va a guardar en la ontología. Pero primero se hace un llamado al API para obtener los atributos que se mostraran en la vista para ingresar los valores, se debe asignar permisos antes de hacer la petición de estos parámetros, así como se puede ver en la figura 36, donde antes de hacer el llamado al API se declara permisos para no tener ningún error.

```
1. public void onCreate(Bundle savedInstanceState) {
2.   super.onCreate(savedInstanceState);
3.   setContentView(R.layout.add_product);
4.   StrictMode.ThreadPolicy policy = new
   StrictMode.ThreadPolicy.Builder().permitAll().build();
5.   StrictMode.setThreadPolicy(policy);
6.   JSONArray prod = null;
7.   List < NameValuePair > param = new ArrayList <
   NameValuePair > ();
8.   JSONObject data_bd = jsonParser.makeHttpRequest(url,
   "POST", param);
```

Figura 36. Función onCreate obtener atributos para ingresar.
Fuente: El autor
Elaboración: El autor

Al obtener los atributos desde el API, y como se muestra en la figura 37 se procede listarlos en la vista para que se pueda ahí ingresar luego los valores.

```
1. prod = data_bd.getJSONArray("atributos_bd");
2.
3. LinearLayout layout = new LinearLayout(this);
4. layout.setOrientation(LinearLayout.VERTICAL);
5.
6. Button btnCreateProduct = new Button(this);
7. for (int k = 0; k < prod.length(); k++) {
8.  LinearLayout row = new LinearLayout(this);
9.  row.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
   LayoutParams.WRAP_CONTENT));
10.    JSONObject c = prod.getJSONObject(k);
11.    String n = c.getString("ATR_NOMBRE");
12.
13.    for (int i = 0; i < 1; i++) {
14.     TextView TxtView = new TextView(this);
15.     TxtView.setLayoutParams(new
   LayoutParams(LayoutParams.WRAP_CONTENT,
   LayoutParams.WRAP_CONTENT));
16.     TxtView.setText(n);
17.     TxtView.setPadding(10, 10, 10, 10);
18.     TxtView.setTextSize(20);
19.     row.addView(TxtView);
20.
21.     EditText EdtTag = new EditText(this);
22.     EdtTag.setLayoutParams(new
   LayoutParams(LayoutParams.WRAP_CONTENT,
   LayoutParams.WRAP_CONTENT));
23.     EdtTag.setId(k);
24.     EdtTag.setWidth(2000);
25.     EdtTag.setHeight(200);
26.     EdtTag.setSingleLine(true);
27.     TxtView.setTextSize(20);
28.     row.addView(EdtTag);
29.    }
30.    layout.addView(row);
31.   }
```

Figura 37. Función onCreate listar atributos para ingresar datos.
Fuente: El autor
Elaboración: El autor

Luego de ingresar todos los campos de texto para mostrar los nombres de los atributos y los espacios de entrada de texto, se crea un botón así como se ve en la figura 38, el cual enviara todos los campos a almacenar.

```
1. btnCreateProduct.setText("Crear Ciudad");
2. btnCreateProduct.setId(R.id.btnCreateProduct);
3. layout.addView(btnCreateProduct);
```

Figura 38. Función onCreate botón de almacenamiento.
Fuente: El autor
Elaboración: El autor

Al tener creado el botón se realizar el evento, el cual leerá todos los atributos ingresados para luego poder ya ser enviados al API. En la figura 39 se puede ver como se envía los parámetros ingresados y campos que se desea ingresar.

```
1. btnCreateProduct.setOnClickListener(new View.OnClickListener() {
2.
3.  @Override
4.  public void onClick(View view) {
5.   JSONArray prod = null;
6.
7.   List < NameValuePair > param1 = new ArrayList < NameValuePair >
   ();
8.   JSONObject data_bd = jsonParser.makeHttpRequest(url, "POST",
   param1);
9.   try {
10.      prod = data_bd.getJSONArray("atributos_bd");
11.      ArrayList < String > list = new ArrayList < String > ();
12.      for (int k = 0; k < prod.length(); k++) {
13.       EditText input = (EditText) findViewById(k);
14.       String name = input.getText().toString();
15.       list.add(name);
16.      }
17.      JSONArray jsArray = new JSONArray(list);
18.
19.      new CreateNewProduct(jsArray, prod).execute();
20.     } catch (JSONException e) {
21.      e.printStackTrace();
22.     }
23.    }
24.   });
```

Figura 39. Función onCreate evento de almacenamiento de datos.
Fuente: El autor
Elaboración: El autor

3.3.2. Iteración 2 – NewCityActivity.java.

Al ejecutarse el evento de envió de los parámetros en esta clase se leerán los campos figura 40, en donde se ordenan los valores ingresados, con el parámetro al que corresponden para luego si poder ser enviados hacia el API.

```
1.  class CreateNewProduct extends AsyncTask < ArrayList
   < String > , String, ArrayList < String >> {
2.    JSONArray val, atrib;
3.
4.    public CreateNewProduct(JSONArray valores,
   JSONArray atributos) {
5.     try {
6.      val = new JSONArray(valores.toString());
7.      atrib = new JSONArray(atributos.toString());
8.     } catch (JSONException e) {
9.      e.printStackTrace();
10.       }
11.      }
```

Figura 40. Función onCreate evento de almacenamiento de datos.
Fuente: El autor
Elaboración: El autor

Como actividad que se ejecuta por delante es cargar un mensaje mientras se envía los valores ingresados al API, para esto la figura 41 muestra la carga de mensaje que se ejecuta mientras hay otras operaciones por debajo.

```
1. protected void onPreExecute() {
2.  super.onPreExecute();
3.  pDialog = new ProgressDialog(NewProductActivity.this);
4.  pDialog.setMessage("Creando Ciudad..");
5.  pDialog.setIndeterminate(false);
6.  pDialog.setCancelable(true);
7.  pDialog.show();
8. }
```

Figura 41. Mensaje de guardado de datos.
Fuente: El autor
Elaboración: El autor

Las función que se ejecutan por debajo permite enviar los datos hacia el API, de donde se espera recibir un mensaje de guardado. Es así como se ve en la figura 42 como se envían los parámetros hacia el API, los cuales son leídos dinámicamente según se ingresaron y los campos disponibles por el API.

```
protected ArrayList < String > doInBackground(ArrayList < String > ...args) {
  // Building Parameters
  List < NameValuePair > params = new ArrayList < NameValuePair > ();

  int size = val.length();
  for (int i = 0; i < size; i++) {
   try {
    JSONObject c = atrib.getJSONObject(i);
    String n = c.getString("ATR_NOMBRE");
       params.add(new BasicNameValuePair(n, val.getString(i)));
     } catch (JSONException e) {
      e.printStackTrace();
     }
    }
    // getting JSON Object
    // Note that create city url accepts POST method
    JSONObject json =
  jsonParser.makeHttpRequest(url_create_product, "POST", params);
```

Figura 42. Función de llamado al API para guardar datos.
Fuente: El autor
Elaboración: El autor

Una vez enviado los datos se obtienen respuesta para validar el guardado, con lo cual en caso de no registrar los datos muestra un error, y al guardarse se llamara a la vista del módulo de consulta. La figura 43 muestra la validación del guardado.

```
try {
 int success = json.getInt(TAG_SUCCESS);

 if (success == 1) {
  // successfully created product
  Intent i = new Intent(getApplicationContext(),
  AllProductsActivity.class);
  startActivity(i);

  // closing this screen
    finish();
   } else {
    // failed to create product
    Log.d("Registration Failure!", "No se puede crear la ciudad");
   }
  } catch (JSONException e) {
   e.printStackTrace();
  }
```

Figura 43. Validación de guardado del módulo agregar.
Fuente: El autor
Elaboración: El autor

La vista para el guardado de datos se puede ver en la figura 44, donde existen los campos requeridos para ser almacenados en la ontología.

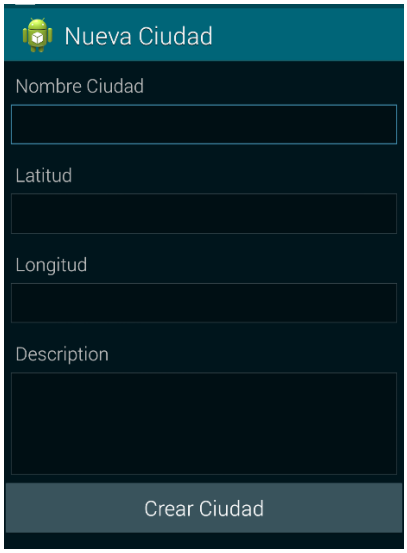

Figura 44. Vista del módulo agregar.
Fuente: El autor
Elaboración: El autor

3.4. Configuración Virtuoso

Para poder realizar las consultas SPARQL en caso de agregar, actualizar o borrar, se requiere permisos configurados correctamente para estás actualizaciones de SPARQL se puedan hacer a través del endpoint. Para ello se debe acceder a Virtuoso como se muestra en la figura 22 y luego de iniciar sesión, seleccionar la pestaña de “System Admin” y luego en la pestaña de “Security” como se muestra en la figura 45.

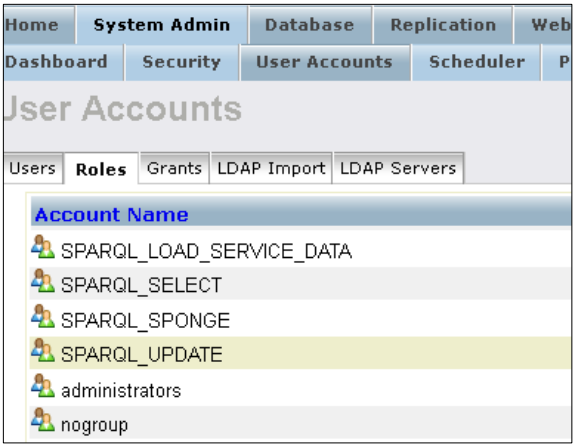

Figura 45. Roles en Virtuoso.
Fuente: El autor
Elaboración: El autor

El rol que se requiere modificar permisos es SPARQL_UPDATE, para ello se edita el rol, la figura 46 muestra que se debe quitar los permisos seleccionados a excepción de SPARQL, con ello se podrán ejecutar las consultas.

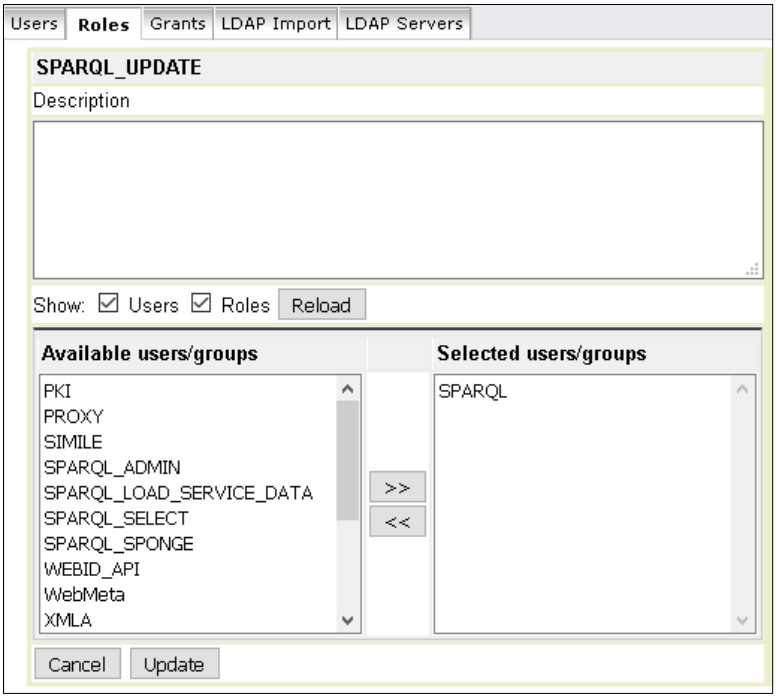

Figura 46. Asignación de permisos a roles.
Fuente: El autor
Elaboración: El autor

3.5. API

Así como en la aplicación y de acuerdo al plan de entrega planteada, el desarrollo del API se establece en 6 iteraciones como se muestra en la tabla 6 del capítulo anterior, algunos de los módulos se llevaran a cabo en paralelo. Además se requiere establecer la parte administrativa del API, para que así este sea dinámico.

3.5.1. API – Información.

Corresponde a el área informativa del API y el punto de partida de la configuración, ya que es en esta parte donde se ingresa primero el grafo al cual se va a hacer referencia para consultar y agregar datos, la figura 47 muestra donde agregar el grafo, el cual se almacenara en la base de datos para luego si poder realizar las configuraciones necesarias.

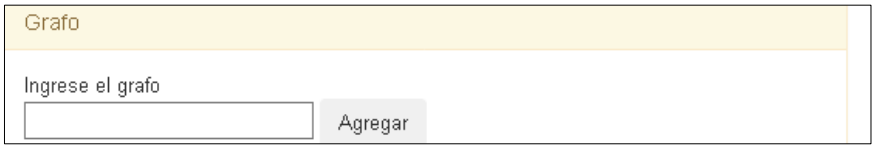

Figura 47. Configuración API agregar grafo.
Fuente: El autor
Elaboración: El autor

Luego de cargar el grafo, automáticamente se listara todas las clases, atributos y URIs que se encuentran ingresados en la ontología, como se ve en la figura 48.

Tabla de Clases

Clase
http://schema.org/City
http://schema.org/Country

Tabla URI

URI
http://schema.org/
http://www.geonames.org/ontology#
http://www.w3.org/2003/01/geo/wgs84_pos#

Tabla atributos

Atributo	Atributo completo
name	http://schema.org/name
name	http://www.geonames.org/ontology#name
lat	http://www.w3.org/2003/01/geo/wgs84_pos#lat
description	http://schema.org/description
countryCode	http://www.geonames.org/ontology#countryCode
long	http://www.w3.org/2003/01/geo/wgs84_pos#long

Figura 48.Campos de la ontología.
Fuente: El autor
Elaboración: El autor

3.5.2. API – Generar Base de datos.

El generar la base de datos permitirá generar luego consultas dinámicas tanto para almacenar como consultar la información introducida en la ontología, esta base de datos administrara el grafo, las clases como ejemplo la figura 49 y los atributos siendo este lo más importante ya que en base a ello se estable que atributos se mostraran para almacenamiento y que atributo se desea consultar.

CLA_ID	CLA_NOMBRE
60	http://schema.org/City
61	http://schema.org/Country

Figura 49.Base de datos ejemplo de almacenamiento de clases.
Fuente: El autor
Elaboración: El autor

En la figura 50 se muestra un ejemplo de los atributos que se agregan en la base de datos, lo cual permite generar las consultas dinámicas mencionadas, así mismo se puede ver los prefijos automáticos, y los atributos que se reconocen con el valor de 1 para realizar las operaciones necesarias.

ATR_ID	ATR_NOMBRE	ATR_COMPLETO	ATR_URI	ATR_PREFIJO	ATR_CONSL	ATR_INGRESAR
141	name	http://schema.org/name	http://schema.org/	A	1	1
142	name	http://www.geonames.org/ontology#name	http://www.geonames.org/ontology#	B	(Null)	(Null)
143	lat	http://www.w3.org/2003/01/geo/wgs84_pos#lat	http://www.w3.org/2003/01/geo/wgs84_pos#	C	(Null)	1
144	description	http://schema.org/description	http://schema.org/	D	(Null)	1
145	countryCode	http://www.geonames.org/ontology#countryCode	http://www.geonames.org/ontology#	E	(Null)	(Null)
146	long	http://www.w3.org/2003/01/geo/wgs84_pos#long	http://www.w3.org/2003/01/geo/wgs84_pos#	F	(Null)	1

Figura 50.Base de datos ejemplo de almacenamiento de atributos.
Fuente: El autor
Elaboración: El autor

Adicional a ello se establece un área para modificar los atributos figura 51, en caso de algún cambio de URI o del nombre del atributo, estas modificaciones se reflejaran tanto en la base de datos, como en la ontología.

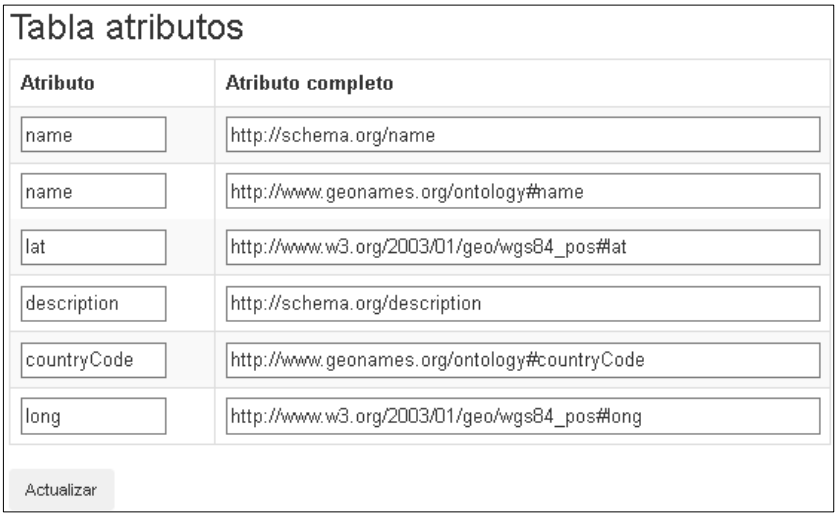

Figura 51.Base de datos ejemplo de almacenamiento de atributos.
Fuente: El autor
Elaboración: El autor

La actualización de estos atributos se puede ver como se realiza en la figura 52, donde primero se elimina y luego se inserta el campo que se desea actualizar.

```
1. /*ACTUALIZAR ONTOLOGIA*/
2. $db = sparql_connect( "http://localhost:8890/sparql" );
3. if( !$db ) { print sparql_errno() . ": " . sparql_error(). "\n";
   exit; }
4.
5. for ($i = 0; $i < $max;$i++) {
6.     $atributo = $_POST['atributo_'.$id_atr[$i]];
7.     $completo = $_POST['comp_atr_'.$id_atr[$i]];
8.
9.     $arrayUP = actualizar_atr($completo,$atributo);
10.       $arrayBD = $row_consulta2[$i]["ATR_COMPLETO"];
11.
12.       if($arrayUP != $arrayBD){
13.           /*CONSULTA Update S*/
14.           $consulta1 = "WITH GRAPH ".$graph;
15.           $consulta1 .= " DELETE { <".$arrayBD."> ?p ?o  . } ";
16.           $consulta1 .= " INSERT { <".$arrayUP."> ?p ?o . } ";
17.           $consulta1 .= " WHERE { <".$arrayBD."> ?p ?o  . } ;";
18.
19.           /*CONSULTA Update P*/
20.           $consulta2 = "WITH GRAPH ".$graph;
21.           $consulta2 .= " DELETE {?s <".$arrayBD."> ?o  . } ";
22.           $consulta2 .= " INSERT {?s <".$arrayUP."> ?o . } ";
23.           $consulta2 .= " WHERE {?s <".$arrayBD."> ?o  . } ;";
24.
25.           $result1 = sparql_query( $consulta1 );
26.           $result2 = sparql_query( $consulta2 );
27.       }
28.   }
```

Figura 52.Actualizar atributos en ontología.
Fuente: El autor
Elaboración: El autor

Luego de actualizar en la ontología se actualiza en la base de datos, figura 53, para ello se recorre toda tabla, para ir actualizando todos los campos en caso de existir cambios en alguno de ellos.

```
1. mysql_select_db($database, $conexion);
2. for ($i = 0; $i < $max; $i++) {
3.   $ATR_ID = $id_atr[$i];
4.  $atributo = $_POST['atributo_'.$id_atr[$i]];
5.  $completo = $_POST['comp_atr_'.$id_atr[$i]];
6.  $atr_completo = actualizar_atr($completo, $atributo);
7.  $atr_uri = generate_prefix($completo);
8.
9.   $query_consulta = "UPDATE atributos SET ATR_NOMBRE = '".$atributo.
10.  "', ATR_COMPLETO = '".$atr_completo.
11.  "', ATR_URI = '".$atr_uri.
12.  "' WHERE ATR_ID = ".$ATR_ID;
13.  $consulta = mysql_query($query_consulta, $conexion) or
     die(mysql_error());
14. }
```

Figura 53.Actualizar atributos en Base de datos.
Fuente: El autor
Elaboración: El autor

3.5.3. API – Configuración del API.

Para la consulta se debe seleccionar el atributo que se desea buscar, para ello solo se puede seleccionar una opción en la columna de consulta. Para agregar se debe seleccionar los atributos que se desea ingresar, la figura 54 es un ejemplo de cómo configurar estos campos, los cuales se listaran luego dinámicamente en la aplicación móvil y web. Además de permitir generar las consultas dinámicas.

Atributo	Atributo completo	Consulta	Agregar
name	http://schema.org/name	◉	☑
name	http://www.geonames.org/ontology#name	○	☐
lat	http://www.w3.org/2003/01/geo/wgs84_pos#lat	○	☑
description	http://schema.org/description	○	☑
countryCode	http://www.geonames.org/ontology#countryCode	○	☐
long	http://www.w3.org/2003/01/geo/wgs84_pos#long	○	☑

Figura 54.Configuración del API.
Fuente: El autor
Elaboración: El autor

3.5.4. Librería.

Para poder realizar consultas SPARQL mediante PHP, se consideró hacer uso de una librería gratuita como se menciono es sparqllib.php, la cual está disponible para ser descargada en su repositorio oficial “https://github.com/cgutteridge/PHP-SPARQL-Lib/”. En base a esta librería se han empleado las siguientes funciones:

- **sparql_get**. Permite obtener el resultado de una consulta que ha sido realizada en SPARQL, su definición es:

```
1.  function sparql_get($endpoint, $sparql) {
2.   $db = sparql_connect($endpoint);
3.   if (!$db) {
4.    return;
5.   }
6.   $result = $db -> query($sparql);
7.   if (!$result) {
8.    return;
9.   }
10.  return $result -> fetch_all();
11. }
```

Figura 55. Función sparql_get.
Fuente: El autor
Elaboración: El autor

- **sparql_errno**. Retorna error en caso de que la consulta no se haya realizado correctamente, se define de la siguiente manera:

```
1. function sparql_errno($db = null) {
2.  return _sparql_a_connection($db) -> errno();
3. }
```

Figura 56. Función sparql_errno.
Fuente: El autor
Elaboración: El autor

- **fields**. Dicha función permite obtener los campos de la consulta SPARQL realizada, y su definición es:

```
1. function fields() {
2.  return $this -> fields;
3. }
```

Figura 57. Función fields.
Fuente: El autor
Elaboración: El autor

3.5.5. API - Modulo Consultar.

El modulo consultar dentro del API es el mediador entre la petición que hace la aplicación y la ontología que se encuentra en Virtuoso.

3.5.5.1. *Iteración 1.*

Primero se requiere llamar a la librería sparqllib.php, la figura 58 muestra cómo se realiza el llamado, y luego se establece la ubicación del endpoint de virtuoso, con ello ya se tendría la conexión lista para cuando la aplicación haga su petición.

```
1. require_once("../sparqllib.php");
2.
3. $db = sparql_connect("http://localhost:8890/sparql");
```

Figura 58. Conexión endpoint API – modulo consultar.
Fuente: El autor
Elaboración: El autor

3.5.5.2. *Iteración 2.*

Luego de establecer la conexión, se realiza la consulta de los atributos de la base de datos, para así poder establecer los prefijos y la consulta se genere dinámicamente, esto se puede ver en la figura 59, esta consulta hace uso de la librería mediante la función sparql_query, la cual retorna la consulta SPARQL ingresada.

```
1. /*grafo*/
2. mysql_select_db($database, $conexion);
3. $query = "SELECT * FROM grafo";
4. $consulta1 = mysql_query($query, $conexion) or die(mysql_error());
5. $row = mysql_fetch_assoc($consulta1);
6. mysql_free_result($consulta1);
7. $graph = "<".$row["GRAFO"]. ">";
8.
9. /*obtener uri para generar prefijos*/
10. $query = "SELECT * FROM atributos WHERE ATR_CONSULTA = '1' ";
11. $consulta1 = mysql_query($query, $conexion) or
    die(mysql_error());
12. $row = mysql_fetch_assoc($consulta1);
13. mysql_free_result($consulta1);
14.
15. /*Establecer prefijos*/
16. sparql_ns("consulta", $row["ATR_URI"]);
17.
18. /*Establecer la consulta*/
19. $where = "consulta:".$row["ATR_NOMBRE"];
20. $sparql = "SELECT ?p FROM ".$graph. "  WHERE { ?s ".$where. " ?p }";
21.
22. $result = sparql query($sparql);
```

Figura 59. Consulta API – modulo consultar.
Fuente: El autor
Elaboración: El autor

3.5.5.3. *Iteración 3.*

Finalmente en este módulo se retorna un error en caso de no existir datos, o se retorna los datos que se extraen de la consulta, la figura 60 muestra como estos datos son retornados en formato JSON.

```
1. if( !$result ) { print sparql_errno() . ": " . sparql_error().
   "\n"; exit; }
2.  echo json_encode($result);
```

Figura 60. Respuesta API – modulo consultar.
Fuente: El autor
Elaboración: El autor

3.5.6. API - Modulo Agregar.

Este módulo permite agregar todos los datos que fueron ingresados en la aplicación, en la ontología que se encuentra en Virtuoso, mediante una consulta SPARQL.

3.5.6.1. *Iteración 1.*

Al igual que en el módulo consultar en el API, lo primero es llamar a la librería sparqllib.php, la figura 61 muestra cómo se realiza el llamado, así también establecer la ubicación del

endpoint de virtuoso, y además en este caso se realiza una validación precia para conocer si hay respuesta del endpoint.

```
require_once( "../sparqllib.php" );

$db = sparql_connect( "http://localhost:8890/sparql" );
if( !$db ) { print sparql_errno() . ": " . sparql_error(). "\n";
exit; }
```

Figura 61. Conexión endpoint API – modulo agregar.
Fuente: El autor
Elaboración: El autor

3.5.6.2. *Iteración 2.*

Establecida la conexión y validando que no existan ningún problema con el endpoint, se realiza un llamado a la base de datos como en la figura 62 para extraer los atributos que serán necesarios para realizar la consulta.

```
mysql_select_db($database, $conexion);
$query = "SELECT * FROM grafo";
$consulta1 = mysql_query($query, $conexion) or
die(mysql_error());
$row = mysql_fetch_assoc($consulta1);
mysql_free_result($consulta1);
$graph = $row["GRAFO"];

/*Obtener el dato principal para almacenar*/
$query = "SELECT * FROM atributos WHERE ATR_CONSULTA = '1' ";
$consulta3 = mysql_query($query, $conexion) or
die(mysql_error());
$row = mysql_fetch_assoc($consulta3);
$atr_guardado = $row["ATR_NOMBRE"];

/*obtener uri para generar prefijos*/
$query = "SELECT * FROM atributos WHERE ATR_INGRESAR = '1' ";
$consulta2 = mysql_query($query, $conexion) or
die(mysql_error());
$row_consulta = array();
while ($row = mysql_fetch_assoc($consulta2)) {
 $row_consulta[] = $row;
}
mysql_free_result($consulta2);
```

Figura 62. Consulta API – modulo agregar extraer atributos de BD.
Fuente: El autor
Elaboración: El autor

Se crea una consulta dinámica en base a los valores ingresados, ya sea mediante la aplicación web o móvil, la cual luego de obtener todos los valores necesarios, se envía la consulta SPARQL, la figura 63 muestra además de la consulta como recibe los parámetros

mediante POST para poder ser almacenados en la ontología con el uso de la función sparql_query de la librería sparqllib.php.

```
$max = sizeof($row_consulta);

/*Prefijos y URI's*/
for ($i = 0; $i < $max; $i++) {
 $ATR_NOMBRE[] = $row_consulta[$i]["ATR_NOMBRE"];
 $ATR_URI[] = $row_consulta[$i]["ATR_URI"];
 $ATR_PREFIJO[] = $row_consulta[$i]["ATR_PREFIJO"];
 sparql_ns($ATR_PREFIJO[$i], $ATR_URI[$i]);
}
sparql_ns("agregar", $graph);
$value_graph = valor_grafo($graph);
/*Establecer la consulta*/
$sparql = "INSERT DATA  { GRAPH <".$graph.
"> { ";
for ($i = 0; $i < $max; $i++) {
 if ($i + 1 != $max) {
  $sparql. = " <".$value_graph.
  ":".$_POST[$atr_guardado].
  "> ".$ATR_PREFIJO[$i].
  ":".$ATR_NOMBRE[$i].
  " '".$_POST[$ATR_NOMBRE[$i]].
  "' . ";
 } else {
  $sparql. = " <".$value_graph.
  ":".$_POST[$atr_guardado].
  "> ".$ATR_PREFIJO[$i].
  ":".$ATR_NOMBRE[$i].
  " '".$_POST[$ATR_NOMBRE[$i]].
  "' ";
 }
}
$sparql. = " }}";

$result = sparql_query($sparql);
```

Figura 63. Consulta API – modulo agregar generar.
Fuente: El autor
Elaboración: El autor

3.5.6.3. *Iteración 3.*

En esta última iteración se valida que si se ha ingresado los valores nos devuelva una respuesta la cual se retorna en formato JSON, así como se ve en la figura 64 como retorna el resultado, para que en la aplicación se pueda ejecutar alguna actividad, en caso de que el almacenamiento sea correcto o un mensaje de existir algún problema.

```
1. $resultado = array();
2. $resultado["success"] = 1;
3. echo json encode($resultado);
```

Figura 64. Respuesta API – modulo agregar.
Fuente: El autor
Elaboración: El autor

3.6. Aplicación Web

De acuerdo a la tabla 6 y conforme a las características de la aplicación móvil, la aplicación web lleva los mismos módulos y se requiere la integración con el API.

3.6.1. Módulo de consulta.

Para el módulo de consulta se trabaja en Front-end y back-end, lo cual permitirá consultar todos los datos ingresados en la ontología y mostrarlos en una lista.

3.6.1.1. *Iteración 1 – Conexión con el API.*

Al ejecutar el evento de llamado de un botón, se realiza el llamado a una función en JavaScript, la cual realiza la conexión con el API, más específicamente con el archivo que retorna los datos de la ontología, como se puede ver en la figura 65. Además se genera la tabla que se presentara en la siguiente iteración que corresponde a mostrar los datos.

```
1. function consultaAPI() {
2.  var xmlhttp = new XMLHttpRequest();
3.  var url = "http://127.0.0.1/ontoApi/core/examples/basic.php";
4.  xmlhttp.onreadystatechange = function() {
5.   if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
6.    var array = JSON.parse(xmlhttp.responseText);
7.    var out = "<table cellpadding='0' cellspacing='0' border='0'
   class='display' id='example' width='100%'>";
8.    out += "<thead><tr><th>Ciudades</th></tr></thead>";
9.    for (var i = 0; i < array.rows.length; i++) {
10.     out += " <tr><td>" +
11.      array.rows[i].p.value +
12.      "</td></tr>";
13.    }
14.    out += "<tfoot><tr><th>Ciudades</th></tr></tfoot>";
15.    out += "</table>";
16.
17.    document.getElementById("myWatch").innerHTML = out;
18.   }
19.  }
20.  xmlhttp.open("POST", url, true);
21.  xmlhttp.send();
22. }
```

Figura 65. Aplicación web conexión al API y generar tabla.
Fuente: El autor
Elaboración: El autor

3.6.1.2. *Iteración 2 - Mostrar los resultados.*

Luego de tener la tabla se presenta en una etiqueta “div”, en donde se listaran los campos solicitados como se puede ver en la figura 66.

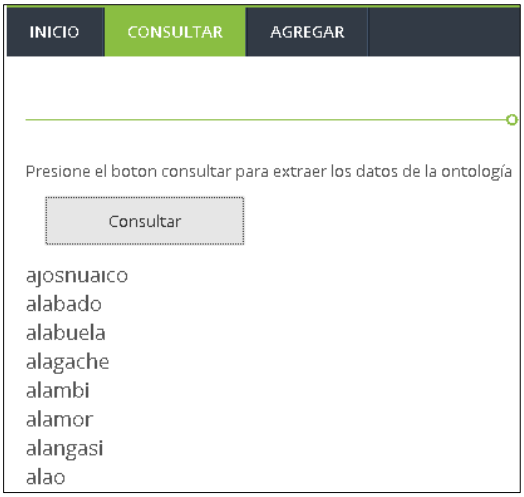

Figura 66. Aplicación web - vista del módulo consultar.
Fuente: El autor
Elaboración: El autor

3.7. Módulo de agregar

Este módulo permite ingresar nuevos datos a la ontología turística, los cuales serán enviados desde la aplicación web a la ontología mediante un API.

3.7.1. Iteración 1 – Conexión al API y configuración dinámica.

Antes de la conexión lo primero es extraer los campos que si están configurados para el almacenamiento en la ontología, figura 67, para ello se consulta la base de datos de la configuración de la ontología, de manera que se generen los campos dinámicamente.

```
require_once('ConnecBD/conexion.php');

mysql_select_db($database, $conexion);
$query_consulta1 = "SELECT * FROM grafo";
$consulta1 = mysql_query($query_consulta1, $conexion) or
die(mysql_error());
$query_consulta2 = "SELECT * FROM atributos WHERE ATR_INGRESAR = '1'";
$consulta2 = mysql_query($query_consulta2, $conexion) or
die(mysql_error());

$row_consulta1 = mysql_fetch_assoc($consulta1);
    $row_consulta2 = array();
    while ($row = mysql_fetch_assoc($consulta2)) {
     $row_consulta2[] = $row;
    }
    $result["grafo"] = $row_consulta1;
    $result["atributos_bd"] = $row_consulta2;
    mysql_free_result($consulta1);
    mysql_free_result($consulta2);

    return $result
```

Figura 67. Extracción de atributos a guardar.
Fuente: El autor
Elaboración: El autor

Luego de tener los campos de la base de datos, se procede a mostrar los en un tabla, así como se muestra en la figura 68, la cual se encuentra dentro de un formulario, que se enviara al momento de ejecutar el evento mediante el botón de almacenamiento.

```
<?php
$max = sizeof($result["atributos_bd"]);

for ($i=0; $i < $max; $i++) {
    echo "<tr>";
    echo "<td
width='153'>".ucwords($result["atributos_bd"][$i]["ATR_NOMBRE"]).
": </td>";
    echo "<td width='31'><label>";
    echo "<input
name='".$result["atributos_bd"][$i]["ATR_NOMBRE"]."' type='text'
id='".$result["atributos_bd"][$i]["ATR_NOMBRE"]."' />";
    echo "</label></td>";
     echo "</tr>";
}
?>
```

Figura 68. Aplicación web - campos a guardar.
Fuente: El autor
Elaboración: El autor

La figura 69, es la vista del usuario donde automáticamente se listan los campos que se encuentren configurados en el API.

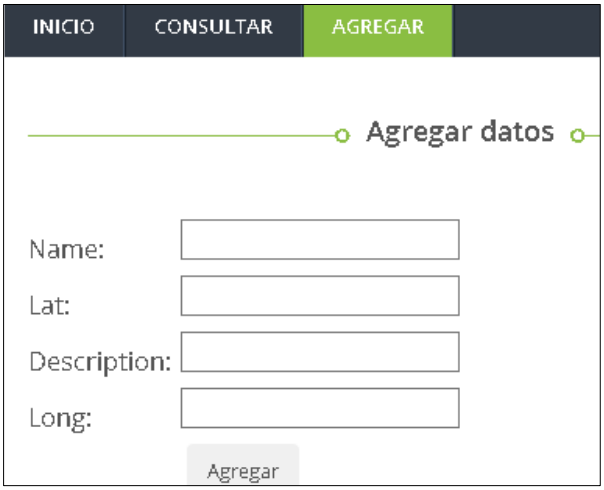

Figura 69. Aplicación web - vista del módulo agregar.
Fuente: El autor
Elaboración: El autor

Finalmente en esta iteración, se procede a la conexión, la cual se realiza de acuerdo a la figura 70, en donde se envía el identificador del formulario de los datos, para luego si poder tener respuesta según el almacenamiento.

```
1. function consultaAPI() {
2.         var xmlhttp = new XMLHttpRequest();
3.         var url =
   "http://127.0.0.1/ontoApi/core/examples/create.php";
4.         var data = new
   FormData(document.getElementById("form_ingreso"));
5.         xmlhttp.onreadystatechange=function() {
6.             if (xmlhttp.readyState == 4 && xmlhttp.status ==
   200) {
7.                 var array = JSON.parse(xmlhttp.responseText);
8.                if(array.success == 1){
9.                    document.getElementById("myWatch").innerHTML
   = "Guardado correcto";
10.                }else{
11.                    document.getElementById("myWatch").innerHTML
   = "Problema al guardar datos";
12.                }
13.             }
14.     }
15.     xmlhttp.open("POST", url, true);
16.     xmlhttp.send(data);
17.     }
```

Figura 70. Aplicación web - vista del módulo consultar.
Fuente: El autor
Elaboración: El autor

3.7.2. Iteración 2 – Respuesta de almacenamiento.

La respuesta del almacenamiento se establece de acuerdo a la respuesta del API, para ello en la figura 70, se puede ver como se especifica 2 tipos de respuesta para el

almacenamiento de los datos, los cuales son presentados en la vista del usuario, como se puede ver en la figura 71, que en este caso se enviaron datos de prueba y no hubo ningún inconveniente en el almacenamiento.

Figura 71. Aplicación web – Respuesta vista agregar.
Fuente: El autor
Elaboración: El autor

CAPÍTULO 4: PRUEBAS E IMPLEMENTACIÓN

En el presente capitulo pretende llevar a cabo un plan de validación y pruebas, en el cuál se define la fase de validación, pruebas y la implementación, para determinar los resultados obtenidos y proceder a concluir, se procede a realizarlas pruebas de: validación, aceptación, rendimiento, integridad de datos, funcionamiento y así mismo pruebas de caja negra y caja blanca.

4.1. Propósito de prueba

El presente plan, tiene el propósito de llevar a cabo una estrategia de pruebas y validaciones, cuya finalidad es reducir aquellos riesgos debido a fallas q se presenten, así como también asegurar el objetivo planteado en el inicio del proyecto.

4.2. Elementos de prueba

Para efectuar las pruebas se ha considerado el siguiente parámetro:

- Dispositivo de prueba: dispositivo móvil Android (Consultas API).

4.2.1. Dispositivo de prueba.

Para poder realizar las pruebas de la aplicación móvil, se empleó un dispositivo Android con las siguientes características:

- Marca: Samsung
- Modelo: Galaxy S5 SM-G900A
- Conexión a internet:
    - 3G HSDPA 42.2Mbps / HSUPA 5.76Mbps
    - 4G LTE Cat. 4
    - Wi-Fi 802.11 a/b/g/n/ac; DLNA; Wi-Fi Direct; banda dual
- Procesador: Qualcomm MSM8974AC Snapdragon 801 quad-core 2.5 GHz.
- Almacenamiento: 16 GB.
- Memoria RAM: 2 GB.
- Sistema: Android 4.4.2 KitKat.

4.3. Objetivos de prueba

La elaboración del plan de pruebas tiene como objetivos:

- Evaluar la transferencia de parámetros para solicitar una consulta u operación.
- Presentación de los datos al realizar la consulta de listar.
- Confirmar el acceso correcto de los parámetros al ingresar datos.

4.4. Plan de Pruebas

Para la aplicación desarrollada y para poder realizar el plan de pruebas, se ha empleado consultas al API a nivel local para verificar así los resultados, los cuales son luego presentados en la aplicación.

Una vez realizadas las consultas a nivel local para almacenamiento y consultas directamente en el API, se procede a realizar la verificación en la aplicación para determinar que las operaciones se ejecuten de la misma manera.

Los perfiles de usuarios considerados para los tipos de pruebas para todos los casos son usuarios con dispositivos móviles.

4.5. Pruebas de Caja Blanca

Las pruebas de caja blanca, se centran en aquellos detalles procedimentales del software, es decir que el diseño está ligado hacia el código fuente. Para realizar estas pruebas que generalmente las lleva quien está desarrollando, se selecciona diversos parámetros que serán la entrada para verificar todos aquellos flujos que ejecuta la aplicación y comprobar que retorna los resultados adecuados o requeridos por la aplicación.

Dentro de las principales técnicas del diseño de las pruebas de caja blanca que se pueden emplear en base a los diagramas de flujo establecidos en el capítulo 2 que corresponde a diseño de solución, se tiene:

- Prueba de condición
- Prueba de flujo de datos
- Prueba de bucles

4.5.1. Prueba de condición.

Esta prueba consiste en que cada condición que se encuentre en el código, tanto de la aplicación como en el API, no contenga ningún error. Por ejemplo una condición sencilla corresponde a una variable lógica o sea una expresión relacional que también puede ser procedida con un operador AND.

Una expresión relacional se representa de la siguiente manera:

- E1 <operador_relacional> E2.

De dicha expresión se tiene que E1 y E2 son expresiones aritméticas y por otro lado se tiene <operador_relacional> el cual puede ser: “<, <=, =, ≠, > o >=”.

Algo que también se evalúa son las condiciones compuestas, las cuales se forman por dos o más condiciones simples, operadores lógicos o paréntesis. Entre los operadores lógicos permitidos para las condiciones compuestas se tiene OR, AND y NOT. Las condiciones que no poseen expresiones relacionales se las conoce como expresiones lógicas.

En si este método de prueba de caja blanca, nos permite evaluar cada condición que se ejecute dentro de nuestra aplicación o API, garantizando así que las mismas no contengan errores.

4.5.2. Prueba de flujo de datos.

Para esta prueba se debe seleccionar los posibles caminos de prueba que presente la aplicación o el API, para estos flujos se ha considerado los diagramas de flujo del capítulo de diseño de solución. La estrategia de esta prueba es comprobar cuan útiles son los caminos de pruebas de la aplicación y el API, en donde existen sentencias o bucles anidados.

Con esta prueba se concluye que gracias a la misma permite ejercitar las condiciones lógicas, es decir, llevar a cabo esta prueba permite comprobar todas las condiciones tanto de la aplicación y el API. Afortunadamente no se ha encontrado ningún problema en ninguna condición, por lo cual no hay problemas en ninguna de las partes antes mencionadas.

4.5.3. Prueba de bucles.

Las pruebas de bucles, son técnicas que nos permite verificar la validez de la construcción de ciertos bucles en el desarrollo de la aplicación o el API, un bucle simple se puede identificar simplemente, ya que se presenta de acuerdo a la figura 72.

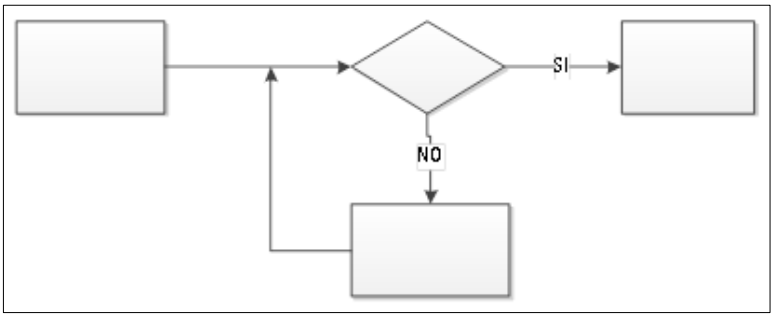

Figura 72. Estructura de un bucle simple.
Fuente: El autor
Elaboración: El autor

Para realizar las pruebas se a considerados solo trabajar en bucles simples, ya que son los que se ha podido identificar de acuerdo a los diagramas. A estos bucles se aplica el

siguiente conjunto de pruebas, donde se identifica a n como el máximo número de pasos q son permitidos por un bucle simple.

- Omitir por completo el bucle.
- Solo un pasó por el bucle.
- Dos pasos por el bucle.
- m pasos por el bucle, donde m<n.
- n = 1, n, n + 1 pasos por el bucle.

4.6. Pruebas de Caja Negra

En el desarrollo y algunas de las pruebas realizadas, se ha realizado pruebas de caja negra, las cuales tienen como finalidad el validar ciertos elementos desde el punto de vista de todas las entradas que recibe y la respuesta que todo el proceso genera, sin considerar el funcionamiento interno. Es decir se relaciona más a las pruebas de funcionalidad, ya que como se sabe en caja negra el interés principal, es la interacción que tiene con el medio que le rodea, de manera que lo importante en estas pruebas es el cómo realiza las funciones y no el como lo hace. Para estas pruebas se definen tanto las entradas y los resultados, de cierta manera se considera conocer más la interfaz que el funcionamiento interno.

Generalmente en programación modular, el programa se encuentra dividido en diversos módulos, para lo cual cada módulo es considerado una caja negra y cada una de estas cajas forman parte de un sistema global. Es por ello que se lleva a tener una independencia de cada caja y en este caso cada módulo representa independencia, pero para realizar la implementación de estos módulos se de conocer como estos se comunican así como se mencionó en base a su interfaz, pero no se requiere saber cómo operan internamente.

4.7. Pruebas de validación

Las pruebas de validación se realizaran sobre el dispositivo, para lo cual se requiere conexión a internet del dispositivo, para que este se pueda conectar con el API, de manera que las pruebas se realizaran tanto en consultas para almacenamiento, como consultas para obtener los datos registrados en la ontología.

4.7.1. Aplicación Móvil.

El primer paso es iniciar la aplicación, la figura 73 corresponde a una captura de las opciones que hacen un llamado a los módulos de consultar o agregar en la aplicación.

Figura 73. Vista inicial de la aplicación.
Fuente: El autor
Elaboración: El autor

4.7.2. Prueba 1: Consultar datos.

Una de las opciones es ingresar a ver ciudades, con lo cual se realiza la petición al API y se realiza una consulta para obtener los datos, la figura 74 muestra un mensaje mientras se realiza la petición, para luego si proceder a listar todos los datos, así como se mostró en la figura 26.

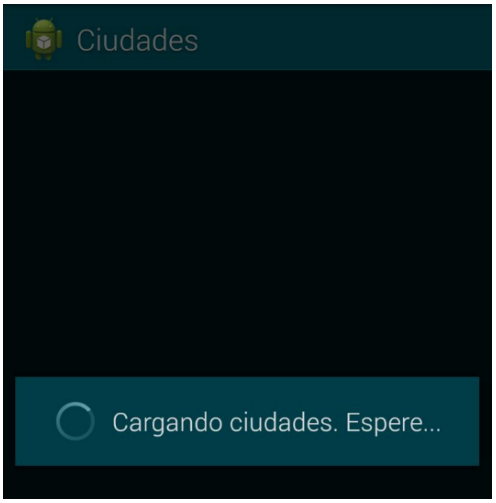

Figura 74. Prueba de validación 1 – Mensaje petición de ciudades
Fuente: El autor
Elaboración: El autor

El proceso a seguir mientras se realiza esta petición de consulta es:

- Comprueba conexión a internet.
- Realizar petición al API
- Respuesta del API.
- Listar datos de la consulta.

4.7.3. Prueba 2: Agregar parámetros.

La otra opción es ingresar a agregar ciudad, para este caso se envían parámetros al API, en donde se ingresan como una consulta, la figura 75 muestra un mensaje de cuando se ejecuta por debajo el envío de los parámetros como son nombre de la ciudad, latitud, longitud y una descripción. Una vez ingresado los parámetros se espera una respuesta ya sea para ingresar a la vista de todas las ciudades, o una alerta de problema de registro.

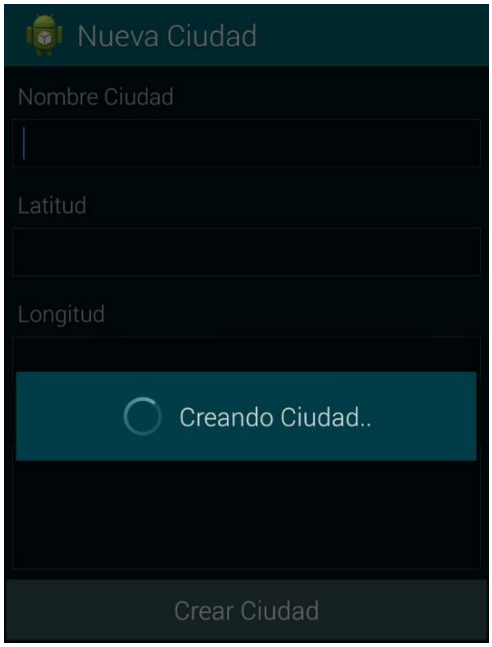

Figura 75. Prueba de validación 2 – Mensaje almacenar ciudad
Fuente: El autor
Elaboración: El autor

El proceso a seguir mientras se realiza esta petición de agregar parámetros es:

- Comprueba conexión a internet.
- Enviar parámetros al API
- Respuesta del API.
- Cambio de vista según la respuesta.

4.8. Pruebas de Aceptación

Las pruebas de aceptación se generaron en base a los entregables que se desarrollaban durante los procesos de iteración y la prueba de validación está orientada sobre el aplicativo móvil.

En estas pruebas se analizan los entregables (módulos) que se crearon en el desarrollo, así como en la aplicación móvil y en la integración con el API.

4.8.1. Aplicación Móvil.

Las pruebas de aceptación están basadas en algunos escenarios posibles, a los cuales el usuario puede hallar al momento que inicia la aplicación. Para las pruebas de aceptación de la aplicación se establecieron las tablas 8, 9 y 10.

Tabla 8. Prueba de aceptación 1 – Conexión con el API

Caso de Prueba de Aceptación 01	
Código: PA01	**Nombre:** Conexión al API
Descripción: Cuando se ejecuta las operaciones de consulta o agregar enviando parámetros, se debe validar que exista conexión a internet para ejecutar las consultas.	
Condiciones de ejecución: El dispositivo dispone del hardware necesario para la ejecución.	
Entrada / Pasos de ejecución: • Consultar ciudades / agregar ciudades. • Comprueba la conexión con el API. • Si existen, comprueba que estén activados. **Opción 1:** o Si están activados, procede a ejecutar las consultas. **Opción 2:** o Si están desactivados, muestra un mensaje para activar la conexión a internet.	
Resultado esperado: Los adaptadores WIFI o Datos están activados.	
Evaluación: Satisfactoria.	

Fuente: El autor
Elaboración: El autor

Tabla 9. Prueba de aceptación 2 – Consultar datos.

Caso de Prueba de Aceptación 02	
Código: PA02	**Nombre:** Consultar datos
Descripción: Al ingresar a ver ciudades, se conectara al API para obtener los datos.	
Condiciones de ejecución: El dispositivo dispone de conexión a internet.	
Entrada / Pasos de ejecución: • Se ingresa a ver ciudades. • Se realiza una petición al API • El API ejecuta una consulta a la ontología. • El API retorna datos • Los datos son estructurados como una lista. **Opción 1:** o Si hay conexión, procede a listar todos los datos. **Opción 2:** o Si no hay conexión, solicitar conexión a internet o Si no hay datos, se muestra la lista vacía	
Resultado esperado: Se listen las ciudades ingresadas en la ontología.	
Evaluación: Satisfactoria.	

Fuente: El autor
Elaboración: El autor

Tabla 10. Prueba de aceptación 3 – Agregar datos.

Caso de Prueba de Aceptación 03	
Código: PA03	**Nombre:** Agregar datos.
Descripción: Al ingresar a agregar ciudad, se conectara al API para guardar los datos.	
Condiciones de ejecución: o Se ha iniciado la aplicación. o No haber eliminado la aplicación de la multitarea del dispositivo.	
Entrada / Pasos de ejecución: • Se ingresa a agregar ciudad. • Se envía los parámetros al API • El API ejecuta una consulta para enviar los parámetros a la ontología. • El API retorna mensaje a la aplicación. • **Opción 1:** o Si hay conexión, se guardan los datos, se carga la vista de listar ciudades. **Opción 2:** o Si no hay conexión, solicitar conexión a internet o Si no hay parámetros, no ejecutar la consulta en el API.	
Resultado esperado: Se almacenen los datos y se pueda visualizar en la lista de ciudades ingresadas.	
Evaluación: Satisfactoria.	

Fuente: El autor
Elaboración: El autor

4.9. Pruebas de Rendimiento

Las pruebas de rendimiento se realizan para determinar el rendimiento de la aplicación en el dispositivo móvil.

4.9.1. Aplicación Móvil.

Dentro de la aplicación móvil se evaluaron los siguientes aspectos:

- Consumo de datos o WIFI.
- Espacio de almacenamiento interno.
- Uso de memoria RAM.

Para establecer estos aspectos, se empleó herramientas nativas del sistema operativo Android.

4.9.1.1. *Consumo de datos móviles*

La aplicación móvil tiene un consumo bajo de datos, la Figura 76 muestra el consumo de WIFI, para las pruebas que se realizaron tanto en el ingreso, como en las consultas para obtener los datos de la ontología.

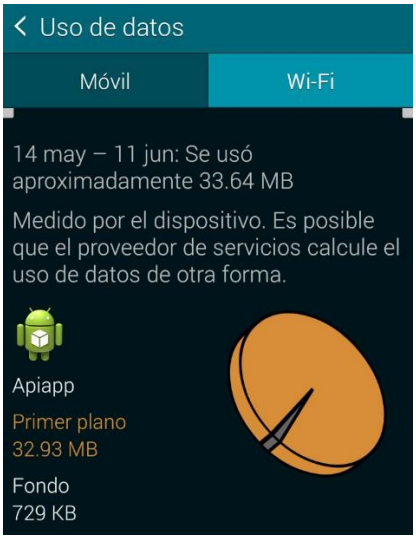

Figura 76. Pruebas de rendimiento - Consumo de datos móviles.
Fuente: El autor
Elaboración: El autor

El consumo de datos realizado es de 32.93 MB, durante el periodo de un mes, considerando que se realizaron aproximadamente 30 peticiones y la ontología contiene gran cantidad de datos registrados.

4.9.1.2. *Espacio de almacenamiento interno*

Haciendo uso de una herramienta nativa dentro de Android como es gestión de aplicaciones, la aplicación posee un tamaño total de 4.52 MB, una vez instalada. La Figura 77 permite ver a detalle el espacio que hace uso la aplicación.

Figura 77. Pruebas de rendimiento - Tamaño de la aplicación.
Fuente: El autor
Elaboración: El autor

4.9.1.3. *Uso de memoria RAM*

El uso que se le da a la aplicación, constituye el consumo de memoria RAM en el dispositivo móvil, por lo que el dispositivo debe contar con cierta cantidad de memoria RAM libre para la ejecución de aplicaciones.

El sistema operativo Android hace uso del 50% o 70% de memoria RAM de un dispositivo móvil, de lo cual el restante está destinado para el uso de aplicaciones.

Para la aplicación generada la figura 78 muestra el consumo de memoria RAM, de la aplicación, cuyo valor es aproximadamente 10 MB, un consumo realmente mínimo.

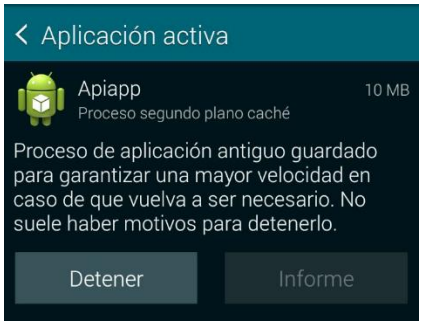

Figura 78. Pruebas de rendimiento - Consumo de memoria RAM.
Fuente: El autor
Elaboración: El autor

4.10. Pruebas de Integridad de datos

4.10.1. Objetivo.

Asegurar la extracción correcta de datos desde la ontología turística de prueba que se encuentra en Virtuoso, mediante el API hasta el dispositivo móvil.

4.10.2. Técnica.

Revisar la ontología y la consulta, para definir qué datos serán extraídos al momento de realizar la consulta desde el API, al hacer la petición desde la aplicación.

4.10.3. Población.

Las pruebas se realizan por administrador y usuarios con acceso a dispositivo móviles:

- Administrador:
    - Revisar directamente en el API el formato y los datos disponibles
- Usuarios con dispositivo móvil:
    - Solicitar acceso a los datos mediante la aplicación.

4.10.4. Escenarios de prueba.

Para esta prueba se requiere especificar que hay dos situaciones tanto para el API y para la aplicación:

- Consumo de datos desde el API
    - Ejecutar consultas desde el API
    - Verificar el retorno en un formato útil para la aplicación
- Consumo de datos desde la aplicación

4.10.5. Escenario 1.a.

Consumo de datos desde el API, ejecutar consultas desde el API.

4.10.6. Resultados.

La consulta que se realiza en el API es introducida en la librería sparqllib, esta consulta puede ser revisada en el endpoint.

4.10.7. Escenario 1.b.

Consumo de datos desde el API, verificar el retorno en un formato útil para la aplicación.

4.10.8. Resultados.

El formato de respuesta se puede ver en la figura 79, donde los resultados se están presentando correctamente en JSON, formato que está leyendo la aplicación.

{"rows":[{"p":{"type":"uri","value":"Prueba"}},{"p":{"type":"uri","value":"aaa"}},{"p":{"type":"uri","value":"abc
agosto","datatype":"http:\/\/www.w3.org\/2001\/XMLSchema#string"}},{"p":{"type":"literal","value":"28 de
{"p":{"type":"literal","value":"a. maldonado","datatype":"http:\/\/www.w3.org\/2001\/XMLSchema#string"}}

Figura 79. Respuesta API formato JSON.
Fuente: El autor
Elaboración: El autor

4.10.9. Escenario 2.

Consumo de datos desde la aplicación

4.10.10. Resultados.

Como se pudo observar anteriormente en la figura 27, los datos extraídos desde el API en formato JSON, son listados para ser entendibles por cualquier usuario.

4.11. Pruebas de Funcionamiento

El objetivo de esta prueba es el poder asegurar la funcionalidad de la aplicación, la carga de los datos desde la ontología y envió de los parámetros desde la aplicación hacia el API.

4.11.1. Técnica.

Para las pruebas de funcionamiento se ejecutan las funciones del dispositivo móvil, para determinar la respuesta del API en los resultados se verifica lo siguiente:

- Qué los datos del API puedan ser listados en la aplicación
- Qué todos los parámetros ingresados en la aplicación se almacenen en la ontología.

4.11.2. Resultado.

- Todas las pruebas planeadas han sido ejecutadas.
- No hay ningún problema con la respuesta del API, el JSON es compatible con la aplicación.
- Se almacena todos los campos ingresados, especificando siempre el a quien pertenece el parámetro con el atributo en la ontología.

En el proceso de las pruebas de funcionamiento, se pudo comprobar que tanto el ingreso de datos, como el consumir los datos del API se ejecuta con normalidad y todo es correcto.

4.12. Implementación

El objetivo de la implementación es obtener un aplicativo que pueda interactuar sin ningún problema con un API, para poder realizar las funciones de consulta y almacenamiento de datos en la ontología turística establecida para pruebas. Ya que la integración entre el API con la aplicación se lleva a cabo en el desarrollo, en la implementación solo se presenta el cómo llevar a cabo las opciones y la configuración del API y la aplicación.

4.12.1. Configuración aplicación móvil.

Para obtener datos e ingresar desde el API se requiere identificar la url del mismo, y luego de ello solo se requiere realizar la configuración del API, ya que este permite mostrar los atributos dinámicamente.

4.12.2. Configuración API.

Se requiere configurar el API, empezando por el ingreso del grafo, luego se genera la base de datos la cual permitirá administrar las consultas y los atributos que se desea ingresar o consultar para obtener la información. De igual manera en la configuración se puede actualizar atributos o URIs en caso de ser necesario.

Es recomendable realizar la configuración de virtuoso que se explicó en el capítulo de Desarrollo, debido a que de no realizarse no se podrá almacenar datos, ya que se requiere dar permisos para poder acceder al endpoint para acciones que requiera actualizar datos que estén en la ontología.

TRABAJOS FUTUROS

El trabajo desarrollado puede llegar a ser empleado como base para futuras investigaciones y aplicaciones, entre estas tenemos algunas ideas que se exponen a continuación:

- Aplicaciones turísticas: Consumo de datos desde ontologías que contengan información de cada región, y almacenamiento de lugares nuevos para crecimiento de las aplicaciones.
- Aplicaciones para Agentes Inteligentes para la Construcción de Sistemas de Información Aplicados al Turismo: Consumir datos de ontologías que permitan dar respuesta de manera automática a interrogantes definidas, en base a datos ingresados previamente o en tiempo real.
- Minería de datos aplicados a turismo: Crear un buscador en línea de sitios turísticos en base de los datos turísticos almacenados, utilizando un modelo de predicción de minería de datos.
- Aplicaciones móviles: Integrar opciones adicionales que permitan brindar mayor información o ser más accesible a la información.

CONCLUSIONES

Acabado el proyecto, se puede llegar a las siguientes conclusiones:

- La investigación de tecnologías para el desarrollo de APIs, consumo y almacenamiento de datos RDF a la nube, ha permitido conocer diversas propuestas, las cuales han permitido establecer una solución óptima para completar el desarrollo del API, y que el consumo y envió así mismo sea lo más rápido.
- Se creó una ontología turística para realizar las pruebas de consumo y envió de datos RDF, la cual se cargó luego en el repositorio semántico virtuoso, para realizar las consultas dinámicas desde el API, de acuerdo a las actividades que se soliciten en la aplicación móvil o web.
- La aplicación móvil es compatible con dispositivos Android versión 4.0 o superior, la cual debe tener conexión a internet, para poder así extraer y enviar datos en tiempo real hacia el API.
- El API mantiene la comunicación entre el aplicativo móvil o web, y la ontología turística, de manera que todas las peticiones del aplicativo son ejecutadas mediante el API, transformando el lenguaje natural a consultas que son ejecutadas en SPARQL, dando respuesta en un formato que la aplicación móvil o web pueda presentar los resultados.
- Se generó un panel de administración del API, para poder gestionar los atributos que se desea almacenar o mostrar, y las consultas de insertar y consumir se generen dinámicamente.
- Se implementó una base de datos MySQL, la cual almacena las configuraciones del panel de administración del API, para poder gestionar los atributos.
- Se desarrolló una aplicación web, con las mismas características de la aplicación móvil, la cual presentaba el dinamismo para el ingreso y consumo de datos, según las configuraciones en el panel de gestión del API.

RECOMENDACIONES

Tomando como referencia el desarrollo de este proyecto y conclusiones, se plantean algunas recomendaciones en caso de futuras implementaciones y/o trabajos relacionados:

- Emplear documentación oficial y hacer uso de las versiones más recientes que tiene el SDK de Android, para no tener problemas de compatibilidad con dispositivos.
- Revisar la documentación para poder hacer uso de la librería “sparqllib.php”, para aprovechar todas las funciones y de igual manera establecer nuevas funcionalidades.
- Investigar trabajos relacionados en base al desarrollo del API que consuma y almacene datos RDF para dispositivos móviles y web, para definir nuevas tecnologías que permitan incrementar funcionalidades.
- En las aplicaciones Android que se van a desarrollar, emplear una metodología ágil, por la variabilidad que existen en los requerimientos y los ajustes a los cambios.
- Aprovechar los datos generados en formato JSON, para generar nuevas opciones de visualización, que sean más útiles para el usuario.
- Integrar nuevas funcionalidades en la aplicación móvil y web, para mejorar la forma en que el usuario percibe la información y asegurar el crecimiento de la aplicación.
- Validar la ontología en caso de crearse y verificar que se pueda realizar consultas antes de realizar pruebas con el consumo desde móvil o web.
- En caso de registrar nuevas funcionalidades, que requiera el uso de consultas SPARQL, se debe optimizar de manera que sean configurables en el API, y así no se requiera que se modifique el código al gestionar nuevos atributos.

BIBLIOGRAFÍA

Aguirre, L., & Sinche, H. (2013). Diseño de una Aplicación Móvil para la consulta Académica de la FIIS-UTP.

Android Developers. (2011). What is Android.

Brähler, S. (2010). *Analysis of the Android Architecture*. *Os.Ibds.Kit.Edu*. Retrieved from http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf

Descamps-Vila, L., Casas, J., Conesa, J., & Pérez-Navarro, A. (2011). Cómo introducir semántica en las aplicaciones SIG móviles : expectativas , teoría y realidad. V Jornadas SIG Libre de Girona, (1).

González, E. (2016). Enriquecimiento automatico de un Data Lake con metadatos.

González, S. (2015). Trabajo de Fin de Grado Publicación de datos sociosanitarios : Una API basada en Open Linked Data.

Lamas, M. I. (2006). Lenguajes de consulta para documentos RDF, 87. Retrieved from http://openaccess.uoc.edu/webapps/o2/bitstream/10609/640/1/38449tfc.pdf

Mosquera, P. (2014). Desarrollo De Nueva Funcionalidad E Interfaz Web Y Mó Vil Para La Api Mappingapi2.

Puerta, J. M. (2015). Desarrollo de una API para la descripción y gestión de Servicios Web REST.

Rudas, J. S., Gómez, L. M., & Toro, A. O. (2013). Revision sistemática de literatura. Caso de estudio: Modelamiento de un par deslizante con fines de predecir desgaste. *Prospect*, *11*(1), 50–58. http://doi.org/10.15665/rp.v11i1.27

Hitz, H.G., Leitner, R., & M. (2006). Web Engineering: the Discipline of Systematic Development of Web Applications. Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki.

Mora, S. L. (2013). Programación de aplicaciones web: historia, principios básicos y clientes web. (Editorial Club Universitario, Ed.)Editorial Club Universitario. (Vol. 53). Alicante: Gamma. http://doi.org/10.1017/CBO9781107415324.004

Pino, M. P. (2014). Extensión Semántica de OML.

Puerta González, J. M. (2015). Desarrollo de una API para la descripción y gestión de Servicios Web REST.

Rudas, J. S., Gómez, L. M., & Toro, A. O. (2013). Revision sistemática de literatura. Caso de estudio: Modelamiento de un par deslizante con fines de predecir desgaste. Prospect, 11(1), 50–58. http://doi.org/10.15665/rp.v11i1.27

Unifying, P. C. R. A., Apis, W., & Data, L. (2013). RDF-REST : A Unifying Framework for Web APIs and Linked Data To cite this version : RDF-REST : A Unifying Framework for Web APIs and Linked Data, 10–19.

Volz, R., Oberle, D., Staab, S., & Motik, B. (2003). KAON SERVER -- A Semantic Web Management System. Scenario, 20–24. Retrieved from http://www2003.org/cdrom/papers/alternate/P029/p29-volz.html