**INSTITUTO POLITÉCNICO NACIONAL**

**Unidad Profesional Interdisciplinaria en**

**Ingeniería y Tecnologías Avanzadas**

PROYECTO TERMINAL Il

**“Módulo de consultas federadas geoespaciales en el contexto de la Web de Linked Data para el triple store Apache Marmotta”**

Que para obtener el título de

**“Ingeniero en Telemática”**

Presenta:

**Páez Ortega Oswaldo Emmanuel**

Asesores:

**Dr. Luis Manuel Vilches Blázquez  
Dra. Cyntia Eugenia Enríquez Ortiz**

# Resumen

El presente proyecto terminal se propone diseñar, implementar y caracterizar un módulo de consultas federadas de datos geoespaciales para un *triple store* que no tenga implementado dicho módulo, en específico, a la plataforma Apache Marmotta cuya arquitectura y funcionamiento están basados en los estándares SPARQL y GeoSPARQL. Además, con el propósito de que los usuarios finales visualicen e interactúen con los resultados desplegados por el módulo de consultas federadas geoespaciales, se usó una aplicación Web para que los resultados recuperados de la Web de *Linked Data* pudieran ser visualizados y explorados.

# Palabras clave

Módulo, consultas federadas, *triple store*, datos geoespaciales, Apache Marmotta, *Linked Data*, SPARQL, GeoSPARQL, Web Semántica, aplicación.

ÍNDICE GENERAL

[1 Resumen 2](#_Toc40282846)

[2 Palabras clave 2](#_Toc40282847)

[1 Capítulo I: Panorama general 9](#_Toc40282848)

[1.1 Introducción 9](#_Toc40282849)

[1.2 Planteamiento del problema 9](#_Toc40282850)

[1.3 Justificación 11](#_Toc40282851)

[1.4 Propuesta de solución 12](#_Toc40282852)

[1.5 Alcances (Resultados esperados) 22](#_Toc40282853)

[1.6 Metodología 22](#_Toc40282854)

[1.7 Objetivo general 23](#_Toc40282855)

[1.7.1 Objetivos específicos 23](#_Toc40282856)

[2 Capítulo II: Estado del arte 25](#_Toc40282857)

[2.1 Trabajos a nivel Internacional 25](#_Toc40282858)

[2.1.1 Querying Geospatial Data over the Web: a GeoSPARQL Interface 25](#_Toc40282859)

[2.1.2 Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL 25](#_Toc40282860)

[2.1.3 Strategies for Executing Federated Queries in SPARQL1.1 25](#_Toc40282861)

[2.1.4 Linking UK Gorvernment Data 25](#_Toc40282862)

[2.1.5 A parallel approach for improving Geo-SPARQL query performance 25](#_Toc40282863)

[2.1.6 DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data 26](#_Toc40282864)

[2.1.7 Explotación de información en el dominio geo-hídrico ecuatoriano utilizando tecnología semántica 26](#_Toc40282865)

[2.1.8 Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data 26](#_Toc40282866)

[2.1.9 Answering geospatial queries over relational data 26](#_Toc40282867)

[2.1.10 Benchmarking Commercial RDF stores with Publications Office Dataset 27](#_Toc40282868)

[2.1.11 Geographica: A Benchmark for Geospatial RDF Stores 27](#_Toc40282869)

[2.1.12 Geoyasgui: The GeoSPARQL query editor and result visualizer 27](#_Toc40282870)

[2.2 Trabajos a nivel Nacional 28](#_Toc40282871)

[2.2.1 Linked Open Data en la Biblioteca Digital Semántica Académica 28](#_Toc40282872)

[2.2.2 Enfoque semántico para el descubrimiento de recursos sensible al contexto sobre contenidos académicos estructurados con OAI-PMH 28](#_Toc40282873)

[2.2.3 Facilitador de contenido móvil para el viajero basado en servicios de localización y Web Semántica 28](#_Toc40282874)

[2.3 Trabajos desarrollados en UPIITA 28](#_Toc40282875)

[2.3.1 Recuperación de información geográfica utilizando similitud semántica 28](#_Toc40282876)

[2.4 Software similar 29](#_Toc40282877)

[3 Capítulo III: Marco teórico 30](#_Toc40282878)

[3.1 Web semántica 30](#_Toc40282879)

[3.2 Linked Data 31](#_Toc40282880)

[3.3 RDF (Resource Description Framework) 32](#_Toc40282881)

[3.4 URI (*Uniform Resource Identifier*) 32](#_Toc40282882)

[3.5 RDF *triple store* 33](#_Toc40282883)

[3.6 SPARQL 33](#_Toc40282884)

[3.7 SPARQL endpoint 34](#_Toc40282885)

[3.8 GeoSPARQL 34](#_Toc40282886)

[3.9 Arquitectura SOA 35](#_Toc40282887)

[3.10 Protocolo HTTP 36](#_Toc40282888)

[3.11 REST 36](#_Toc40282889)

[3.12 JSON 37](#_Toc40282890)

[3.13 Traductor 38](#_Toc40282891)

[3.14 Compilador 38](#_Toc40282892)

[3.15 Intérprete **¡Error! Marcador no definido.**](#_Toc40282893)

[3.16 Jerarquía de lenguajes: 38](#_Toc40282894)

[3.17 Partes de un compilador 38](#_Toc40282895)

[4 Capítulo IV: Análisis del sistema 41](#_Toc40282896)

[4.1 Descripción general 41](#_Toc40282897)

[4.2 Perspectiva del producto 41](#_Toc40282898)

[4.3 Características de los usuarios 41](#_Toc40282899)

[4.4 Restricciones 41](#_Toc40282900)

[4.5 Suposiciones y dependencias 42](#_Toc40282901)

[4.6 Requisitos específicos 42](#_Toc40282902)

[4.7 Requerimientos funcionales 42](#_Toc40282903)

[4.8 Requerimientos no funcionales 50](#_Toc40282904)

[4.9 Interfaces de hardware 51](#_Toc40282905)

[4.10 Interfaces de software 51](#_Toc40282906)

[4.11 Interfaces de comunicación 51](#_Toc40282907)

[5 Capítulo V: Diseño del sistema 52](#_Toc40282908)

[5.1 Caso de uso 52](#_Toc40282909)

[5.2 Diagramas de clase 53](#_Toc40282910)

[5.3 Diagramas de estado 54](#_Toc40282911)

[5.3.1 Diagrama de estado para la aplicación web – Usuario 54](#_Toc40282912)

[5.3.2 Diagrama de estado para la aplicación web – Administrador 55](#_Toc40282913)

[5.3.3 Diagrama de estado para módulo de consultas en Apache Marmotta 56](#_Toc40282914)

[5.4 Diagramas de secuencia 57](#_Toc40282915)

[5.4.1 Diagrama de secuencia para aplicación web (Usuario). 57](#_Toc40282916)

[5.4.2 Diagrama de secuencia para aplicación Web (Administrador) 59](#_Toc40282917)

[5.4.3 Diagrama de secuencia para el módulo. 60](#_Toc40282918)

[5.5 Secuencia de interfaces 63](#_Toc40282919)

[6 Referencias 64](#_Toc40282920)

Índice de figuras

[Figura 1 Analogía entre MySQL y Apache Marmotta. 10](#_Toc40282921)

[Figura 2 Elementos que contiene Apache Marmotta considerados en el proyecto. 10](#_Toc40282922)

[Figura 3 Apache Marmotta no soportaba consultas federadas. 10](#_Toc40282923)

[Figura 4 Diagrama de Apache Marmotta haciendo una consulta individual (línea sólida) y consultas federadas (líneas punteadas). 12](#_Toc40282924)

[Figura 5 Consulta geoespacial simple. 12](#_Toc40282925)

[Figura 6 Diagrama a bloques de una consulta geoespacial federada. **¡Error! Marcador no definido.**](#_Toc40282926)

[Figura 7 Diagrama a bloques de la implementación del módulo de consultas federadas. 20](#_Toc40282927)

[Figura 8 Propuesta de visualización. 22](#_Toc40282928)

[Figura 1 Diagrama propuesto por Tim Berners-Lee mostrando las tecnologías que conforma la Web Semántica. 30](#_Toc40282929)

[Figura 2 La nube de datos *Linked Data.* 31](#_Toc40282930)

[Figura 3 Grafo de RDF. 32](#_Toc40282931)

[Figura 4 Diagrama de cómo está compuesto un URI. 32](#_Toc40282932)

[Figura 6 Resultado de la consulta en SPARQL. 33](#_Toc40282933)

[Figura 7 Representación de datos espaciales. 34](#_Toc40282934)

[Figura 8 Resultado de la consulta en GeoSPARQL. 35](#_Toc40282935)

[Figura 9 Caso de uso del diseño del sistema. 52](#_Toc40282936)

[Figura 10 Diagrama de clases. 53](#_Toc40282937)

[Figura 11 Diagrama de estados - aplicación Web – usuario. 54](#_Toc40282938)

[Figura 12 Diagrama de estados - Aplicación Web Administrador. 55](#_Toc40282939)

[Figura 21 Diagrama de estados - Módulo de consultas federadas. 56](file:///C:\Users\zacat\Documents\10mo%20Semestre%20UPIITA\PTII\Proyecto-Terminal-2\Tercer%20Parcial\Documento.docx#_Toc40282940)

[Figura 22 Diagrama de secuencia - aplicación Web – usuario común. 57](file:///C:\Users\zacat\Documents\10mo%20Semestre%20UPIITA\PTII\Proyecto-Terminal-2\Tercer%20Parcial\Documento.docx#_Toc40282941)

[Figura 15 Diagrama de secuencia - Aplicación Web - usuario común. 58](#_Toc40282942)

[Figura 16 Diagrama de secuencia - Aplicación Web - usuario común, continuación. 58](#_Toc40282943)

[Figura 17 Diagrama secuencia aplicación Web - Administrador. 59](#_Toc40282944)

[Figura 18 Diagrama de secuencia para módulo de consultas federadas geoespaciales 60](#_Toc40282945)

[Figura 19 Diagrama secuencia inicio sesión. 61](#_Toc40282946)

[Figura 20 Diagrama de secuencia - selección de modo. 61](#_Toc40282947)

[Figura 21 Diagrama de secuencia - validación consulta. 62](#_Toc40282948)

Índice de tablas

[Tabla 1. Relación entre variables dependientes e independientes en el *Benchmarking*. 20](#_Toc40282949)

[Tabla 2 Tabla comparativa de software similar a Apache Marmotta. 29](#_Toc40282950)

[Tabla 1 Característica de usuario normal. 41](#_Toc40282951)

[Tabla 2 Características de usuario administrador. 41](#_Toc40282952)

[Tabla 3 Requerimiento funcional establecer comunicación 42](#_Toc40282953)

[Tabla 4 Requerimiento funcional validar conexión. 42](#_Toc40282954)

[Tabla 5 Requerimiento funcional selección modo operación. 43](#_Toc40282955)

[Tabla 6 Requerimiento funcional modo *dataset*. 43](#_Toc40282956)

[Tabla 7 Requerimiento funcional selección *dataset*. 43](#_Toc40282957)

[Tabla 8 Requerimiento funcional cargar datos. 44](#_Toc40282958)

[Tabla 9 Requerimiento funcional modo consulta. 44](#_Toc40282959)

[Tabla 10 Requerimiento funcional enviar consulta. 44](#_Toc40282960)

[Tabla 11 Requerimiento funcional validar consulta. 45](#_Toc40282961)

[Tabla 12 Requerimiento funcional recibir resultados. 45](#_Toc40282962)

[Tabla 13 Requerimiento funcional visualizar datos. 45](#_Toc40282963)

[Tabla 14 Requerimiento funcional explorar datos. 46](#_Toc40282964)

[Tabla 15 Requerimiento funcional iniciar de sesión. 46](#_Toc40282965)

[Tabla 16 Requerimiento funcional iniciar de sesión. 46](#_Toc40282966)

[Tabla 17 Requerimiento funcional validar usuario. 47](#_Toc40282967)

[Tabla 18 Requerimiento funcional validar nuevo usuario. 47](#_Toc40282968)

[Tabla 19 Requerimiento funcional dar de baja a usuario. 47](#_Toc40282969)

[Tabla 20 Requerimiento funcional cargar consulta federada. 48](#_Toc40282970)

[Tabla 21 Requerimiento funcional extracción URI y argumentos. 48](#_Toc40282971)

[Tabla 22 Requerimiento funcional Consulta *triple store*. 48](#_Toc40282972)

[Tabla 23 Requerimiento funcional procesar resultados. 49](#_Toc40282973)

[Tabla 24 Requerimiento funcional guardar datos. 49](#_Toc40282974)

[Tabla 25 Requerimiento no funcional Disponibilidad. 50](#_Toc40282975)

[Tabla 26 Requerimiento no funcional Usabilidad. 50](#_Toc40282976)

[Tabla 27 Requerimiento no funcional Interfaz de la aplicación. 50](#_Toc40282977)

[Tabla 28. Requerimiento no funcional Confidencialidad 51](#_Toc40282978)

índice de fragmentos de código

# Capítulo I: Panorama general

## Introducción

La creación de la Web, llevada a cabo por Tim Berners Lee, y la popularidad que alcanzó, provocó que los usuarios se interesaran en aportar contenido de toda índole en poco tiempo, sin prestar atención a desarrollar un conjunto de buenas prácticas, la cuales sirvieran como referencia para los usuarios al momento de crear y subir contenido a la Web. Debido a esta omisión, la posibilidad de tener una web inteligente se volvería difícil de lograr, esto a consecuencia de que las computadoras no son capaces de interpretar ni de hacer inferencias en el contenido de la Web [1]. Sin embargo, se propuso una evolución que le permitiría a la Web tener un contexto y significado en el contenido que alberga en ella; es aquí donde surgió la Web Semántica. Con esta propuesta se pretendió que el contenido en la Web pudiera ser interpretado por las computadoras a nivel semántico [2]. A partir de este acontecimiento surgieron servidores de *triple store* cuya información que almacenan son tripletas en documentos del tipo Marco de Descripción de Recursos (RDF, por sus siglas en inglés) que describen entidades y relaciones en la Web Semántica, a través de grafos y a su vez, surgieron las plataformas de *Linked Data* (LDP, por sus siglas en inglés) las cuales son herramientas que son capaces de manipular dichas entidades y las relaciones existentes entre ellas.

Las *triple stores* están basadas en el protocolo y lenguaje de consultas para RDF [3], *SPARQL* por sus siglas en inglés, que es el lenguaje estandarizado de consultas para bases de datos de tipo RDF. Este concepto también es utilizado en el dominio de la Web Semántica para datos geoespaciales, donde para la realización de consultas se utiliza el estándar *GeoSPARQL* [4]. Este estándar, en conjunto con tecnologías propias de la Web Semántica, ha sido aplicado en problemas de logística, hidrología, turismo, entre otros [5]. Estos ejemplos, con frecuencia, presentan propuestas donde se realizan consultas únicamente a un *triple store* y éste se encarga de devolver la información almacenada con características geoespaciales. No obstante, existen algunas propuestas donde se han realizado algunos ejemplos *ad hoc* de consultas federadas en el ámbito de los datos geoespaciales [6]. Sin embargo, el estado del arte actual presenta una importante limitación, ya que existen *triple stores* que no permiten la realización de consultas federadas a través de múltiples *triple stores* que presenten de información geoespacial (conforme a *GeoSPARQL*) en el contexto de la nube de *Linked Data*.

## Planteamiento del problema

Se ha demostrado que aproximadamente el 80% de los datos tienen relación con una ubicación geográfica [7]. Esta es una de las razones por la que las herramientas que traten con datos geoespaciales están en constante actualización. El caso de la implementación del *Linked Data*, no es la excepción. Investigadores, empresas y organizaciones gubernamentales usan la nube del *Linked Data* para el estudio y administración de su información [8]. Actualmente existen diversos motores de *triple store* que no son capaces de hacer consultas a diversas fuentes de información geográfica a la vez, es decir, consultas federadas a *triple stores* de datos geoespaciales conforme a los estándares *SPARQL* 1.1 y *GeoSPARQL*.

Existen diversas empresas y organizaciones que se encargan de desarrollar herramientas para la Web Semántica y *Linked Data* para la manipulación y almacenamiento de datos semánticos. Una de estas organizaciones, es la organización sin fines de lucro *Apache Software Foundation* (ASF) la cual ofrece diversas herramientas para diferentes necesidades en cuanto a software se refiere. Ejemplos de estas herramientas son servidores Web, *frameworks*, bases de datos, entre otras. Para el mundo del *Linked Data* y Web Semántica, ASF también tiene su plataforma y es conocida como Apache Marmotta. Para contextualizar a Apache Marmotta, una analogía con bases de datos SQL se muestra en la figura 1.

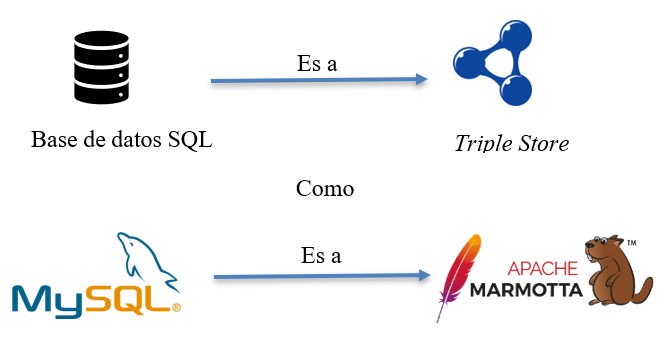
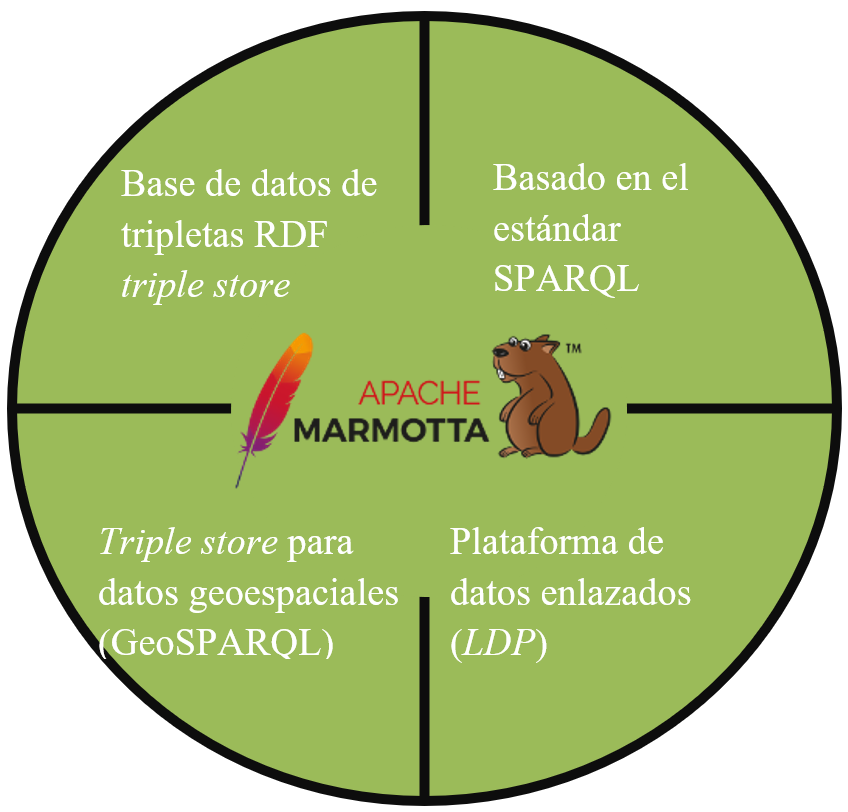


Figura 1 Analogía entre MySQL y Apache Marmotta.

La plataforma Apache Marmotta cuenta con diversas características, entre las cuales, fueron 3 las relevantes para este proyecto: es una LDP, es un *SPARQL* *endpoint* y también es una base de datos para tripletas RDF, *triple store*. En el caso particular de este proyecto, el trabajo está completamente desarrollado sobre Apache Marmotta. Tal y como se indica en la figura 2, entre los elementos que contiene este *triple store,* no se encuentra un módulo de consultas federadas de datos geoespaciales.



y SPARQL *endpoint.*

Figura 2 Elementos que contiene Apache Marmotta considerados en el proyecto.

La plataforma funciona con los estándares *SPARQL* y *GeoSPARQL* los cuales establecen la manipulación correcta de datos en la nube del *Linked Data* y datos geoespaciales respectivamente. Esto quiere decir que tales características ya existen, pero lo que aún no se había implementado es la capacidad de hacer consultas federadas. Como prueba, la figura 3 muestra una captura de pantalla del sitio oficial de Apache Marmotta[[1]](#footnote-1) encerrado en un rectángulo rojo, indicando que la plataforma aún no era capaz de hacer consultas federadas.

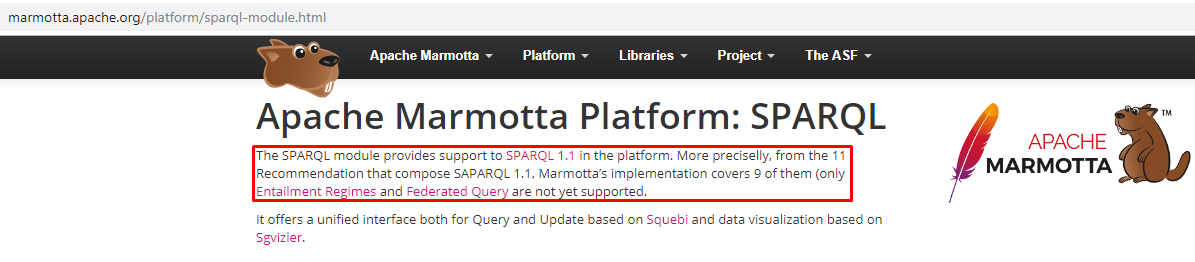


Figura 3 Apache Marmotta no soportaba consultas federadas.

Sin embargo, hubo aspectos que se debieron cumplir para que el proyecto fuera realidad. El primer punto que se abordó es que Apache Marmotta [9] está escrito en Java por lo que se tuvo que comprender cómo es que está diseñado y construido. Las librerías, objetos y el paradigma de programación implementados son ejemplos de los retos que se enfrentaron. Igualmente, se tuvieron que atender los protocolos que fueron indispensables para que Apache Marmotta funcionara junto al módulo desarrollado.

Así mismo, puesto que se escribieron algoritmos y código, dicho código debía ser eficiente y mantenible. El desarrollo fue optimizado a través de la múltiples pruebas llevadas a cabo y que están descritas en el escenario de pruebas.

Otros aspectos que se consideraron fue la funcionalidad del módulo, la cual permitió que las consultas federadas se desplegaran en la nube de *Linked Data* y que el usuario final pudiera recuperar información geoespacial de forma federada. Lo anterior se logró mediante la combinación de respuestas de las diversas fuentes consultadas y el procesamiento de la información para que los resultados no fueran repetidos u omitidos, y la correcta manipulación de datos geoespaciales para poderlos desplegar sobre una aplicación Web. En este sentido, existen diversas opciones para visualizar los datos tanto públicas como libres. ArcGIS con su API que permite embeber su herramienta de visualización de geometrías o GeoJson.io que ofrece la opción de dibujar también las geometrías sobre un mapa. En este sentido, existen diversas alternativas para lograr esta meta, un ejemplo es la herramienta *Map4RDF* [10] cuya función es visualizar y explorar *datasets* RDF cuya información sea geométrica o *GeoYASGUI* [11]que es un editor y visualizador de consultas basadas en *GeoSPARQL*. Por tanto, cabe aclarar que se reutilizará una herramienta de este estilo solo para visualizar los datos recuperados por el módulo de consultas federadas geoespaciales.

De esta forma, la intención del presente proyecto es contribuir a las herramientas *open source* usadas en la Web Semántica. La pregunta que estuvo presente en el desarrollo del proyecto fue: ¿Cuánta información revelará una consulta geoespacial federada frente a una consulta a un único repositorio? Con la intención de cumplir tal propósito la pregunta que surge es: ¿Cuánta información adicional revelará una consulta geoespacial federada frente a una consulta a un único repositorio?

## Justificación

Ante este escenario, el desarrollo e implementación del módulo que solucione dicha problemática fue clave para que el software Apache Marmotta tenga la capacidad de hacer consultas a repositorios remotos y obtener mejores resultados en las búsquedas.

Un aspecto que se consideró para justificar el desarrollo del módulo propuesto en este proyecto es que las bases de datos no solo son del tipo distribuidas, sino que también existen aquellas que están bajo el esquema de bases de datos federadas. En el escenario de la Web Semántica, a este tipo de bases de datos se les denomina *federated* *triple store*. Con esta característica también se involucra el hecho de que los servidores estén en diferentes localizaciones geográficas, por lo que si no se realizan consultas federada a una *triple store* federada, probablemente se estaría obteniendo una porción de información. Por el contrario, si la consulta realizada es federada, se obtendría una respuesta completa que agrupa los resultados de las múltiples *triple store*. Además, considerando la característica de que la información proviene de distintos proveedores de información, el desarrollo de este módulo permitirá que los expertos en las múltiples disciplinas que tiene asociada la información geográfica [6] puedan abordar sus problemáticas con una nueva herramienta que permita integrar y enriquecer sus análisis y estudio.

A pesar de que Apache Marmotta es un software desarrollado por Apache, aún tiene características por incorporar al sistema. Ejemplo de estas características es que al estar basado en el estándar *SPARQL* 1.1, este documento estipula que hay 11 características por cumplir para que un *triple store* se considere completo conforme a esta versión del estándar, la cual es *SPARQL* 1.1. Apache Marmotta carece de 2 características: consultas federadas y regímenes de vinculación [12]; las consultas federadas es una característica que se implementó en el presente proyecto terminal con el fin de ofrecer una herramienta *open source* a desarrolladores e investigadores que usen la Web Semántica en sus trabajos e investigaciones sin invertir dinero.

## Propuesta de solución

En la actualidad, existen diversos motores *triple store,* aunque gran parte de ellos [13] solo pueden hacer consultas de forma individual sobre aquellos conjuntos de datos con características geoespaciales que tienen almacenados. Así, la herramienta que se desarrolló en el proyecto terminal propone una solución a este problema brindando la posibilidad de realizar consultas distribuidas a conjuntos de datos geoespaciales presentes en la nube de *Linked Data* para el *triple store* Apache Marmotta. De esta manera, se pretende aprovechar los beneficios que dicha plataforma ofrece para avanzar en el estado del arte y contribuir con el desarrollo y progreso de la Web Semántica geoespacial.

En la figura 4 se muestra con una línea sólida una consulta simple a un solo *triple store*, mientras que, en las líneas punteadas representan la capacidad de consultar diversos *triple store* a la vez, funcionalidad que se implementó al software Apache Marmotta en este proyecto.

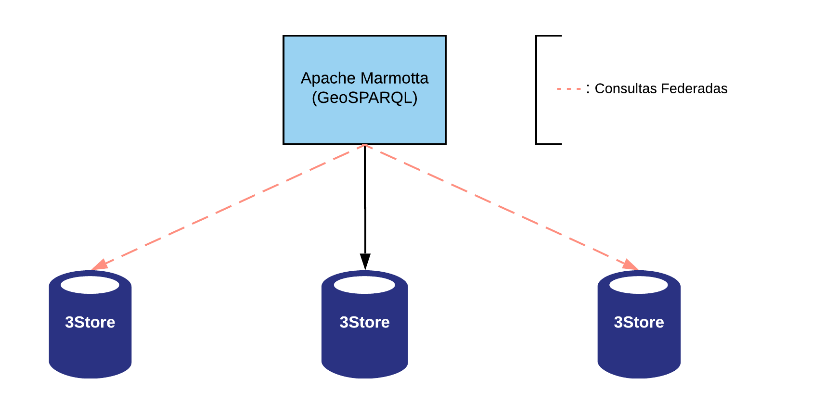


Figura 4 Diagrama de Apache Marmotta haciendo una consulta individual (línea sólida) y consultas federadas (líneas punteadas).

Detallando la figura 4, en la figura 5 se muestra un diagrama a bloques que indica el proceso de una consulta geoespacial simple, es decir, una consulta a un solo *triple store*. Con base a los estándares *SPARQL,* *GeoSPARQL* y la documentación de Apache Marmotta, la consulta según los operadores usados retornará una geometría o un booleano.

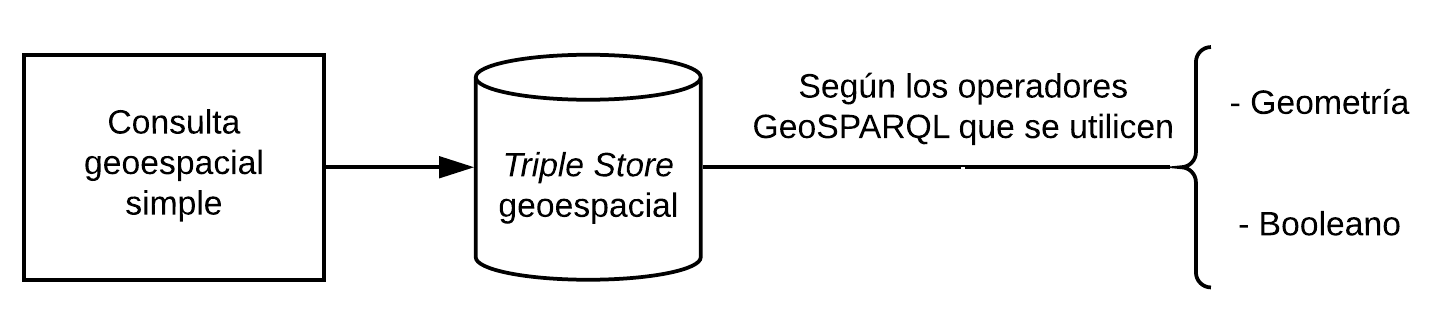


Figura 5 Consulta geoespacial simple.

La intención del módulo desarrollado es que las consultas en Apache Marmotta no se limiten en un solo *triple store,* sino que consulten a todas las *triple store* que estén disponibles y que presenten información con características geoespaciales. Una vez que se consulten a todas las bases de datos geoespaciales, las respuestas tendrán que ser procesadas de tal forma que no exista datos repetidos. El resultado retornado podrá contener más datos y se menciona que se puede ya que los datos que se le soliciten a un cierto *triple store* dependerán si el endpoint cuenta con ellos o no.

El desarrollo e implementación del módulo de consultas federadas geoespaciales están basados en los estándares *SPARQL* y *GeoSPARQL*, es decir, el funcionamiento, desarrollo y resultados cumplen con las características estipuladas en los documentos *SPARQL 1.1 Federated Query* [3] y *A Geographic Query Language for RDF Data* [14].

Apache Marmotta está basado en el lenguaje de consultas de la Web Semántica *SPARQL* y es capaz de hacer consultas geográficas *SPARQL* cuya tecnología es mejor conocida como *GeoSPARQL*. El módulo que ha sido desarrollado está integrado al código fuente del motor para lograr que éste sea capaz de hacer múltiples consultas en vez de una sola además de funcionar con datos geoespaciales según el estándar *GeoSPARQL*. La figura 7 muestra una representación de los módulos de Apache Marmotta; en verde se encuentra el lenguaje y protocolo *SPARQL*, en amarillo la extensión *GeoSPARQL* y en rojo se muestra el módulo de consultas federadas implementado.



Figura 6 Diagrama a bloques de la implementación del módulo de consultas federadas.

Para la caracterización del módulo desarrollado, se propuso llevar a cabo un conjunto de pruebas descritas en el escenario de pruebas. Esto a su vez permitió realizar un *benchmarking* contra otras plataformas similares para determinar sus fortalezas y debilidades respecto a dichas plataformas.

Se analizaron 3 artículos que abordan métodos de evaluación en sistemas de consultas federadas. En ellos se hablan de variables independientes y dependientes. Las variables independientes son aquellas características que debe de ser especificadas con el fin de asegurar que sean replicables en los escenarios de evaluación mientras que las variables dependientes son aquellas que serán medidas en la evaluación. La tabla 1 muestra la relación existente entre ellas

Tabla 1. Relación entre variables dependientes e independientes en el *Benchmarking*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variables independientes | | Variables dependientes | | |
| Tiempo de selección del *SPARQL endpoint* | Tiempo de ejecución | Completitud de la respuesta |
| Consulta | Forma del plan de ejecución | Si | Si | Si |
| Número de patrones de tripletas básicas en la consulta | Si | Si | Si |
| Instancias y posición en las tripletas | Si | Si | No |
| Datos | Tamaño del RDF *dataset* | No | Si | No |
| Características estructurales del *dataset* | No | Si | No |
| Tipo de partición | Si | Si | Si |
| Distribución de datos | Si | Si | Si |
| Plataforma | Memoria RAM | Si | Si | No |
| Número de procesadores | Si | Si | No |
| Administración de memoria caché | Si | Si | No |
| SPARQL endpoint | Número de *SPARQL endpoints* | Si | Si | Si |
| Tipo de *SPARQL endpoints* | Si | Si | No |
| Distribución de las transferencias | Si | Si | Si |
| Latencia de red | Si | Si | Si |
| Retraso inicial del *SPARQL endpoint* | Si | Si | No |

La tabla 1 muestra cuáles son las variables fueron debidamente evaluadas en el *benchmarking* (variables dependientes), y su impacto en las características específicas (variables independientes).Los parámetros considerados en la caracterización son los mismos que se utilizaron para efectuar el *benchmarking*:

* Tiempo de selección del *SPARQL endpoint*
* Tiempo de ejecución.
* Completitud de respuesta.

Los parámetros pueden ser leído con más detalle en los siguientes documentos:

* *FedBench: A Benchmark Suite for Federated Semantic Data Query Processing* [15]
* *SP2Bench: A SPARQL Performance Benchmark* [16]
* *The berlin sparql benchmark* [17]

Con los datos recabados y una vez desarrollado el módulo, se pudo redactar la documentación de la herramienta que permita su correcto uso, así como su compartición con la comunidad relacionada a la Web Semántica y *Linked Data*.

Además, el módulo se complementó con una herramienta para que los usuarios finales pudieran interactuar con los resultados de las consultas geoespaciales federadas. Actualmente ya existen herramientas que permiten visualizar datos geoespaciales, entonces con el único fin de demostrar que el módulo funciona correctamente, por ende, como otra alternativa de comprobación de funcionamiento, este trabajo presenta los resultados obtenidos usando una de esas herramientas que permite desplegar dicha información y analizarla de forma amigable sobre una aplicación Web. Una propuesta de visualización se muestra en la figura 12 donde se aprecian puntos en color azul que representan la información geoespacial recuperada de múltiples *triple store*.

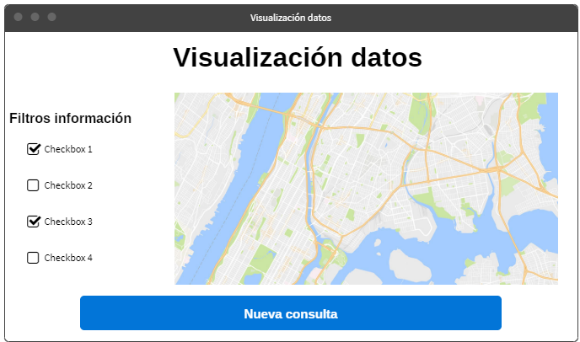


Figura 7 Propuesta de visualización.

## Alcances (Resultados esperados)

Al no haber existido un módulo de consultas federadas geoespaciales para Apache Marmotta, ningunos de los alcances propuestos habían sido elaborados hasta el momento. Los alcances planteados para este trabajo fueron:

* El *triple store* Apache Marmotta en conjunto con el módulo desarrollado es capaz de hacer consultas federadas geoespaciales con base a los estándares *SPARQL* y *GeoSPARQL*.
* Se realizó un *benchmarking* basado en las propuestas descritas en [15], [16] y [17].
* La caracterización está delimitada por los resultados arrojados en el *benchmarking*.
* La comprobación y explotación de datos geoespaciales devueltos por el módulo de consultas federadas en Apache Marmotta se efectuó mediante una aplicación Web donde el usuario puede escoger *datasets* precargados o escribir una consulta federada geoespacial para poder visualizar los datos solicitados.
* Ya que el módulo está actualmente desarrollado, implementado y haber sido puesto a prueba con las pruebas descritas en el escenario de pruebas, se propuso a la organización *Apache Software Foundation* que el módulo sea incorporado en el sistema Apache Marmotta de forma oficial mediante un *pull request* en su repositorio oficial.

**Limitaciones**

* Se necesita una computadora con conexión a Internet, sistema operativo Unix-Linux Ubuntu para instalar Apache Marmotta y el módulo y un navegador web para visualizar la aplicación web.
* El módulo está integrado al código fuente del motor Marmotta por lo que no es un *framework* ni una API. Su uso está completamente dentro del entorno Apache Marmotta. La herramienta de visualización *Leaflet* no forma parte del desarrollo si no que es un recurso para la visualización de los resultados de las consultas embebido en la aplicación Web.
* No se busca competir contra los otros *triple store* similares ya que solo se busca incorporar una nueva funcionalidad a Apache Marmotta.
* El proyecto no trata sobre efectuar una búsqueda semántica inter espacial.

## Metodología

La toma de decisión sobre qué metodología se usaría en el proyecto, la decisión fue basada en el artículo de Alfonso Fuggetta, quien propone que el desarrollo de software *open source* debe de llevarse a cabo con metodologías que sean de rápido prototipado cuyo desarrollo sea evolutivo e incremental. Fuggetta propone que las metodologías como espiral y las ágiles son las adecuadas [18].

En la metodología tipo espiral se consideran 4 fases: Determinación de objetivos, análisis de riesgo, desarrollo y prueba y planificación. La ventajas que ofrece esta metodología es que es muy flexible en cuanto a los cambios que requiera el cliente, implementación, reutilización de software e incorporación de objetivos de calidad en el desarrollo del proyecto. Sin embargo, las desventajas que existen en la metodología es que el tiempo de desarrollo es ambiguo ya que al estar haciendo los cambios que el cliente requiera puede nunca terminar y, por ende, si no se realizó un análisis y diseño correcto del sistema, puede negativamente afectar por completo al proyecto.

Las metodologías que Fuggetta también propone son las ágiles. Si bien la más popular es *SCRUM*, se optó por considerar la metodología *Extreme Programmig (XP)* ya que se adaptó al desarrollo del proyecto y al contexto de este. La metodología *XP* se adapta para equipos que cuentan con pocas personas y en proyectos donde el desarrollo del sistema sea de forma incremental.

En *XP* se consideran roles dentro del equipo: Clientes, programadores, *testers*, *coach* y mánager. En esta metodología, al igual que espiral, también se manejan 4 prácticas: Planeación, diseño, codificación y pruebas. Una de las ventajas que la metodología en espiral ofrece, *XP* también lo proporciona y es la característica de ser una metodología recursiva. Cada vez que se terminen las 4 fases, se planea la siguiente etapa del proyecto para que la siguiente etapa se concluyan las historias pendientes y así poder avanzar con las siguientes

*XP* está basada en historias que el cliente propone con ayuda de los *testers*, las cuales son pequeñas tareas que el sistema debe de hacer. Estas historias son desarrolladas por el equipo de programación mediante pruebas unitarias y los *testers* se encargan de validar el correcto desarrollo y funcionamiento de estas. Esta característica de *XP* fuerza al equipo a terminar todas las historias antes de continuar con las siguientes. Cada vez que se terminan las historias correspondientes a la actual fase del proyecto, se procede a integrarlo al proyecto maestro. La práctica de integración evita que los problemas de comunicación e implementación entre interfaces se presenten al implementar cada prueba unitaria aprobada por los *testers* de forma individual en vez de integrar una característica al proyecto por completo.

Ya que *XP* es la metodología que mejor se adaptó a las condiciones y contexto del desarrollo del proyecto, se determinó usar esta metodología.

## Objetivo general

Desarrollar un módulo de consultas geoespaciales federadas para el *triple store* Apache Marmotta, con el propósito de contribuir al avance de las tecnologías usadas en la Web Semántica y proveer una alternativa *open source* diferente a los *triple store* existentes.

### Objetivos específicos

* Implementar, con base en los estándares *SPARQL*, *GeoSPARQL*, así como auxiliándose de otras tecnologías involucradas en la Web Semántica y *Linked Data,* un módulo de consultas federadas para el *triple store* Apache Marmotta*.*
* Evaluar el rendimiento de las consultas hechas por el módulo desarrollado (*benchmarking*).
* Comparar el *triple store* Apache Marmotta con otros *triple store* auxiliándose de la caracterización y *benchmarking* del módulo.
* Caracterizar el módulo de consultas federadas.
* Explotar lar características geoespaciales de los datos obtenidos del despliegue de consultas federadas mediante una aplicación Web para poder visualizarlos y explorarlos.

# Capítulo II: Estado del arte

Los trabajos que se presentan en este capítulo o bien usan las herramientas del *Linked Data* o dan un enfoque de cómo se usan las tecnologías, pero ninguna aborda una propuesta similar a la considerada en este trabajo.

## Trabajos a nivel Internacional

### *Querying Geospatial Data over the Web: a GeoSPARQL Interface*

En este artículo [19] se describe cómo es que Nancy, Ralhp y Dave crearon e implementaron una interfaz para datos *GeoSPARQL* llamada *GeoQuery TOOL*. Esta interfaz intuitiva pretende hacer que las consultas geoespaciales sean más fáciles de hacer al implementar listas en su interfaz para poder escoger atributos y operadores espaciales. Con base en los datos de entrada que haya ingresado el usuario, *GeoQuery* genera código *GeoSPARQL* automáticamente, realiza la consulta usando el *triple store* Parliament y es desplegado en una aplicación web en vez de utilizar el Sistema de Información Geográfica (GIS, por sus siglas en inglés).

### *Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL*

Este trabajo de Robert y Dave [20] se presentan razones por las cuáles hay que usar *GeoSPARQL*, su estado del arte en la industria y en la investigación, y su implementación de *GeoSPARQL* en el *triple store* Parliament. Explican conceptos geoespaciales tales como las diferencia entre una característica y una geometría, qué es un sistema de referencia de coordenadas (CRS, por sus siglas en inglés) y las relaciones topológicas que existen. En esta última mencionan 8 operaciones básicas y sus 2 variantes Egenhofer y RCC8. Ambas expresan las mismas operaciones al ser equivalentes. En general, este documento da las herramientas para comprender *GeoSPARQL*, así como su uso e implementación con la intención de que empresas u organizaciones consideren adoptar esta tecnología.

### *Strategies for Executing Federated Queries in SPARQL1.1*

En esta propuesta [21] se analizan diferentes estrategias para implementar consultas federadas con la intención de evitar los límites que un *endpoint* presenta. Las estrategias que proponen están basadas en la versión de *SPARQL* 1.1 mediante descomposición de consultas federadas. En este artículo se describe la sintaxis de *SPARQL* mediante teoría de conjuntos, así como la evaluación de las estrategias propuestas probando los teoremas propuestos. Por último, los autores muestran la mejora de resultados habiendo implementado sus estrategias.

### *Linking UK Gorvernment Data*

En este trabajo [22] se establecen los casos de uso para la adopción de los principios de *Linked* Data para la publicación de datos públicos del gobierno de Reino Unido. Además, los autores plantean los beneficios de usar *Linked Data*. En sí, el trabajo pretende convencer a empresas, centros de estudio y a desarrolladores a empezar a usar *Linked Data*. En el documento se abordan los temas de datos públicos del gobierno y la responsabilidad que deben existir para su publicación, patrones de diseño, tópicos imprescindibles para abordar la publicación de datos relacionados con información estadística y geoespacial. Por último, presentan las tecnologías disponibles en el contexto *Linked Data* para que desarrolladores de software puedan crear nuevas herramientas.

### *A parallel approach for improving Geo-SPARQL query performance*

Esta investigación [23] expone el problema actual que existe en las consultas geoespaciales que involucran complejas relaciones topológicas y que en conjunto a las bases de conocimiento las cuales no están indexadas, generan ineficiencia en consultas de sus datos. Entonces, en el documento aparte de exponer el problema, proponen una estrategia para disminuirlo, que consiste en el uso de cómputo en paralelo y la creación virtual de índices de la base de conocimiento. En el desarrollo del documento, se muestran el antes y el después de los resultados obtenidos al usar su estrategia sobre consultas en la ciudad de Connecticut, EUA.

### *DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data*

El documento [24] propone una nueva forma de hacer un *benchmarking* con el propósito de demostrar que los *triples stores* existentes no son tan homogéneos como los otros *benchmarking* lo muestran, es decir, el rendimiento depende de los parámetros usados por lo que una consulta no puede mostrar la misma eficiencia respecto a otra consulta.

Se ponen a prueba 4 *triple store* populares Virtuoso, Sesame, Jena-TDB y BigOWLIM. A lo largo del documento se muestra el desarrollo del *benchmarking*, desde la generación del conjunto de datos de DBpedia, hasta los resultados obtenidos. En este trabajo los autores miden la cantidad de consultas por segundo (QpS, por sus siglas en inglés) en 4 casos: 10%, 50%, 100% y 200% del conjunto de datos generados previamente.

A parte del *benchmarking*, se presenta una discusión donde se habla sobre las fortalezas y debilidades de cada *triple store* al hacer múltiples y simples consultas con ellos. También se presentan trabajos relacionados y los retos por vencer en el campo de los *benchmarking* en *triple stores*.

### Explotación de información en el dominio geo-hídrico ecuatoriano utilizando tecnología semántica

El documento [6] presenta una propuesta de cómo aprovechar los datos geográficos en el dominio geo hídrico. Se apoyan en la única herramienta *open source,* Parliament, para llevar a cabo las consultas geográficas de forma federada.

Presentan el escenario del dominio geo hídrico ecuatoriano y la manera de cómo explotan los datos mediante 3 repositorios de *triple store*.

Cuando terminan de obtener los resultados de las consultas, estos son visualizados con el apoyo de una herramienta llamada MAP4RDF.

Cabe decir que en este trabajo no se hace ninguna implementación o desarrollo sobre un *triple store*, solo utilizan ejemplos de consultas y obtienen datos.

### *Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data*

Es este documento [25] presentan una herramienta Web llamada *Sextant* que permite la exploración de datos enlazados geoespaciales para la creación, compartición y edición colaborativa mediante la combinación de datos geoespaciales enlazados y otro tipo de información disponible en archivos de formato OGC.

### *Answering geospatial queries over relational data*

En este artículo [26] los autores parten del hecho de que los datos geoespaciales son comúnmente almacenados en sistemas manejadores de bases de datos, *DBMS* por sus siglas en inglés, del tipo geoespacial. Para llevar a cabo dicha tarea, se deben de convertir esos datos en datos RDF y almacenarlos en *triple store* cada vez que nuevos datos llegan. Esta labor resulta ser monótona, generando apatía en los administradores de dichas bases de datos para actualizarlas. Si bien existe el paradigma de administración de datos denominada Acceso a Datos Basado en Ontologías, *OBDA* por sus siglas en inglés, que se encargan de ofrecer consultas *SPARQL* en formato SQL, no existe soporte para datos geoespaciales; esto implica no hacer consultas actualizadas o no obtener resultados de consultas geoespaciales. La solución por parte de los autores es habilitar consultas del tipo *GeoSPARQL* *en el aire* al no convertir los datos consultados a RDF y después almacenarlos en un *triple store*. Como resultado, se pueden hacer consultas sin tener que hacer conversiones y almacenamiento, solo puras consultas.

### *Benchmarking Commercial RDF stores with Publications Office Dataset*

Los autores presentan un *benchmark* [27] para RDF *stores* usando *datasets* de la oficina de publicaciones, *PO* por sus siglas en inglés. En la comparación se miden 4 características: *bulkloading*, escalabilidad, estabilidad y ejecución de consultas. De la misma *PO* se utilizaron *datasets* normalizados y no normalizados. Usando lo mismos *datasets* se construyeron nuevos datos para medir la escalabilidad; en estas consultas de dividieron en 2 categorías: consultas instantáneas, consultas que involucran operadores primitivos, y consultas analíticas con el propósito de medir la calidad de servicio y no solo la velocidad. Se concluye que el rendimiento no es homogéneo entre sistemas y que la calidad y velocidad de resultados dependen de diversos parámetros como el tipo de consultas, características de base de datos o *hardware* utilizado.

### *Geographica: A Benchmark for Geospatial RDF Stores*

Se consideró por parte de los autores de este artículo [28], que no existía un *benchmark* que evaluara *triple store* geoespaciales que fuese usado de una manera conocida, por lo que el propósito de haber desarrollado el *benchmark* *Geographica* fue contribuir al estado de arte. En *Geographica* se usaron datos sintéticos y datos del mundo real para probar la funcionalidad ofrecida y el rendimiento de RDF *stores* geoespaciales; se usaron en específico Strabon, Parliament y uSeekM los cuales eran los más completos para la evaluación. El desarrollo de esta comparativa fue con el fin de ofrecer una metodología que permitiera evaluar RDF *stores* de una mejor manera que propuestas previas. Ellos usaron 2 *workloads* de datos: sintético y del mundo real. Con ellos evaluaron eficiencia de funciones primitivas espaciales y el rendimiento de los RDF *stores* en *reverse geocoding, map search y browsing.* Los autores buscan, como trabajo futuro, expandir el *benchmark* para cubrir el estándar *GeoSPARQL* de una manera completa y probar a *Geographica* en un ecosistema centralizado y distribuido.

### *Geoyasgui: The GeoSPARQL query editor and result visualizer*

Los autores abordan una problemática presente en los editores y evaluadores de consultas de *GeoSPARQL* [11]*,* al trabajar con el *Land Registry and Mapping Agency,* mejor conocido como *Kadaster,* quienes publican una gran cantidad de *datasets* entre los cuales son publicados de diversas maneras y en una cantidad considerable, muchos de esos datos son geoespaciales. Cabe decir que *Kadaster* publica sus datos basados el estándar *GeoSPARQL* como *Linked Open Data*. Básicamente lo que hace el sistema es evaluar las consultas al ser enviadas a un *SPARQL* *endpoint* donde se evalúa el álgebra respecto a la colección de datos almacenados en el *endpoint,* esto con el objetivo de optimizar la consulta y después el *endpoint* devuelve los resultados de la consulta en un formato estandarizado (XML, JSON, CSV/TSV). En cuanto a la edición de las consultas, los autores se basaron en trabajos previos: *YASQE, YASR, YASGUI* pero usando los componentes de *GeoSPARQL,* dando así una edición de consultas de datos geoespaciales ofreciendo un visualizador de consultas, un *feedback* directamente al usuario al autocompletar y resaltar sintaxis de la consulta, y un servicio Web que une los elementos anteriores.

## Trabajos a nivel Nacional

### *Linked Open Data* en la Biblioteca Digital Semántica Académica

En este trabajo [29] se describe y analiza cómo es que el *Linked Open Data* es aplicado en las bibliotecas digitales semánticas. En el documento se presenta un marco teórico explicando las tecnologías que son necesarias para el trabajo, tales como XML, RDF, *SPARQL*, *triple store.* Además, se describen las herramientas con las que cuentan en la UNAM para el desarrollo de la biblioteca digital. También presentan iniciativas de *Linked Open Data* en Alemania, Reino Unido y en México. Para concluir su documento, dan una justificación del porque su proyecto es innovador al usar nuevas alternativas de publicación, búsqueda y recuperación de información.

### Enfoque semántico para el descubrimiento de recursos sensible al contexto sobre contenidos académicos estructurados con OAI-PMH

En este trabajo [30], describen un enfoque que considera los recursos de información estructurados con el Protocolo para Cosecha de Metadatos de la Iniciativa de Archivos Abiertos, OAI-PMH por sus siglas en inglés, representación ontológica y su uso en aplicaciones de recuperación de información. Los autores presentan los conceptos a tomar en cuenta como son OAI-PMH, Dublin-Core, sensibilidad al contexto, ontologías. En el trabajo se usó Apache Jena, el cual es un *triple store* donde llevaron a cabo sus pruebas. Se presenta una metodología para consultar y obtener información. Después muestran sus resultados mostrando un grafo que describe la relación entre las instancias de ejemplo que propusieron. Para finalizar, abordan trabajos relacionados, conclusiones y el trabajo a futuro para mejorar su propuesta.

### Facilitador de contenido móvil para el viajero basado en servicios de localización y Web Semántica

En esta tesis [31] desarrollada en el CIC del IPN se propone el diseño e implementación de un facilitador de contenido móvil usando técnicas de la Web Semántica y datos geográficos con *GeoSPARQL* para recomendaciones de alimentación, hospedaje y sitios de interés a turistas. El proyecto se basó en lenguajes de la Web Semántica como RDF, OWL y *SPARQL*. Fue desarrollado sobre un *servlet* de Java en conjunto a servicios basado en geolocalización. Se basó en una metodología dividida en 5 etapas: conceptualización, recuperación de términos, recuperación de información turística y su presentación. Al ser un proyecto que pretende estar disponibles en cualquier dispositivo móvil, se hicieron pruebas experimentales de su servicio web, de la aplicación móvil, de la recuperación de información turística y la forma de presentar los datos. Por último, en la conclusión se hace una comparación del proyecto respecto a una popular aplicación llamada *TripAdvisor,* con la principal característica de que en el proyecto desarrollado muestra resultados precisos, limitados, pero correspondían a la zona donde el usuario se encuentra, mientras que la aplicación comercial en diversos casos devolvía resultados que no estaban relacionados a la consulta hecha.

## Trabajos desarrollados en UPIITA

### Recuperación de información geográfica utilizando similitud semántica

Trabajo hecho por 2 estudiantes [32] donde proponen el uso de un sistema de información geográfica (GIS) y un sistema de geoposicionamiento global (GPS) para identificar sitios de interés alrededor de la zona de donde se encuentre el usuario usando técnicas de similitud semántica, en específico la teoría de confusión. El proyecto fue probado en los alrededores de la IPN Zacatenco para encontrar banco, hospitales y lugares para comer y entretenerse.

## Software similar

Actualmente, no existen más que 4 *triple* *store* que se asemejan en condiciones a Apache Marmotta. Estos cuatro *triple store* implementan *GeoSPARQL* y están actualmente activos. A continuación, se describirán un poco cada uno de ellos

* Parliament
  + Licencia: *BSD License.*
  + Última actualización: mayo 2019.
  + Lenguajes: Java, C++.
* GraphDB
  + Licencia: Comercial.
  + Última actualización: agosto 2019.
  + Lenguaje: Java.
* Openlink Virtuoso
  + Licencia: Comercial.
  + Última actualización: octubre 2018
  + Lenguaje: C.
* Apache Jena
  + Licencia: Apache 2.
  + Última actualización: mayo 2019.
  + Lenguaje: Java.

La tabla 2, mostrada a continuación, se muestran 4 *triple store* que existen en la actualidad. Dichas plataformas son las que se usaron en el *benchmarking* contra *Apache Marmotta.*

Tabla 2 Tabla comparativa de software similar a Apache Marmotta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Consultas Federadas | GeoSPARQL | Status | Libre o pago |
| *Parliament* | Si | Si, en versión 2.7.4 | Activo | Libre |
| *GraphDB* | Si | Si | Activo | Ambos |
| *OpenLink Virtuoso* | Si | Si | Activo | Ambos |
| *Apache Jena* | Si | Si | Activo | Libre |
| *Apache Marmotta* | No | Si | Activo | Libre |

Tal como se mencionó en la justificación del proyecto, desarrollar herramientas que contribuyan al desarrollo de la Web Semántica es primordial, ya que son pocas las plataformas libres para los usuarios que se apegan a los protocolos *SPARQL* y *GeoSPARQL*. Solamente 2 plataformas son completamente libres; los demás *triple store* en la tabla 2 y los que no fueron incluidos en la tabla, son de pago, no se apegan a los protocolos o no tienen las funcionalidades que las plataformas de la tabla 2 ofrecen.

# Capítulo III: Marco teórico

Los conceptos descritos en esta sección fueron fundamentales en el presente proyecto terminal para mostrar los elementos que lo componen. RDF, URI, *triple store* y *SPARQL* *endpoint* son conceptos fundamentales que se trabajaron a lo largo del desarrollo del módulo; tales conceptos explican los objetos con los que se tuvieron que abordar para este proyecto terminal.

En lo que respecta a *Linked Data* y Web Semántica, estos son conceptos que se tuvieron que contemplar, ya que no se debe de perder de vista sobre qué contexto se estuvo desarrollando el módulo, porque bien se pudo hablar de datos cualquiera pero no fue así, el proyecto terminal está construido bajo estos conceptos.

Con respecto a *SPARQL* y *GeoSPARQL*, estos fueron los estándares que se siguieron y cumplieron. Si no hubiera sido así, simplemente no se hubiera podido nada que funcione con el *triple store* Apache Marmotta o cualquier otra plataforma que esté basada en cualquiera de los dos estándares.

## Web semántica

La Web semántica es el grupo de actividades que el consorcio de la *World Wide Web* con la intención de construir tecnologías que permitan publicar datos para que puedan ser legibles por computadoras a través de conceptos que definan los objetos de la Web, semántica que permita a las máquinas interpretar los conceptos de los objetos, metadatos (RDF) como recurso para describir a los objetos y ontologías para establecer la relación existente entre los objetos de la Web. En la figura 9 se muestra la pila que Tim Berners-Lee propone para describir la Web Semántica en cuanto a elementos que lo conforman.

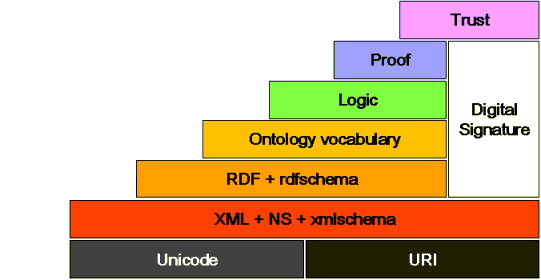


Figura 8 Diagrama propuesto por Tim Berners-Lee mostrando las tecnologías que conforma la Web Semántica.

Describiendo de abajo hacia arriba, en el primer nivel se encuentran las tecnologías que permiten la identificación de cada entidad (Unicode y URI). En el segundo nivel se encuentran las tecnologías que le dan estructura, pero no significado, al documento publicado en la web (XML, NS, XML *schema*). A partir del tercer nivel al sexto, la pila muestra una firma digital (columna a la derecha) para garantizar la autenticidad de cada entidad. En el tercer nivel están las tecnologías que describen el contenido del documento, tal que permiten a la mayoría de las computadoras entender el significado de la entidad en cuestión (RDF y RDF schema). En el cuarto nivel, si bien no es una tecnología, se encuentra un documento o archivo que define las relaciones entre entidades mediante ontologías y reglas de inferencia (comúnmente se implementa el lenguaje OWL para lograrlo). En el quinto nivel se encuentra la lógica, la cual reúne las diversas ontologías usadas al igual que las reglas de los lenguajes que se usaron para describir y estructurar la entidad; en este nivel se llevan a cabo las inferencias y se les da un significado a los datos. En el sexto nivel, se realiza la prueba, es decir, el cómo se hicieron las inferencias y el origen de los datos. Por último, se encuentra la confianza, la confianza de que el sistema es capaz de funcionar correctamente, de que el sistema pueda explicar qué hace, del origen de las fuentes de datos y servicios, así como la tecnología e interfaz de usuario.

La Web Semántica se puede ver como un todo y el *Linked Data* como los elementos que la conforman puesto que proporciona sustento y las bases para que la Web Semántica sea construida correctamente [33].

## Linked Data

El *Linked Data* es un conjunto de buenas prácticas para publicar y conectar datos estructurados en la web. Los principios [34] que tiene asociado son los siguientes:

* URI para identificar entidades en el mundo.
* HTTP que es el mecanismo con el que se recuperan recursos o descripciones de recursos.
* Protocolo SPARQL y archivos RDF para estructurar, consultar y enlazar los objetos presentes en la nube del *Linked Data*.

Ya que el *Linked* Data está basado en web, la diferencia entre sitios de Internet comunes y los basados en *Linked* Data es que mientras que los HTML simples en la web son conectados mediante hipervínculos comunes, Linked Data se basa en documentos que albergan datos en formato RDF.

En la figura 10 se observa la nube de datos *Linked Data,* la cual existe en que existe en *DBpedia*, donde cada burbuja representa un *triple store*. Cada color implica un dominio de datos distinto tales como gobierno, ciencias, multimedia entre otros. El dominio que compete al actual proyecto son los geoespaciales.

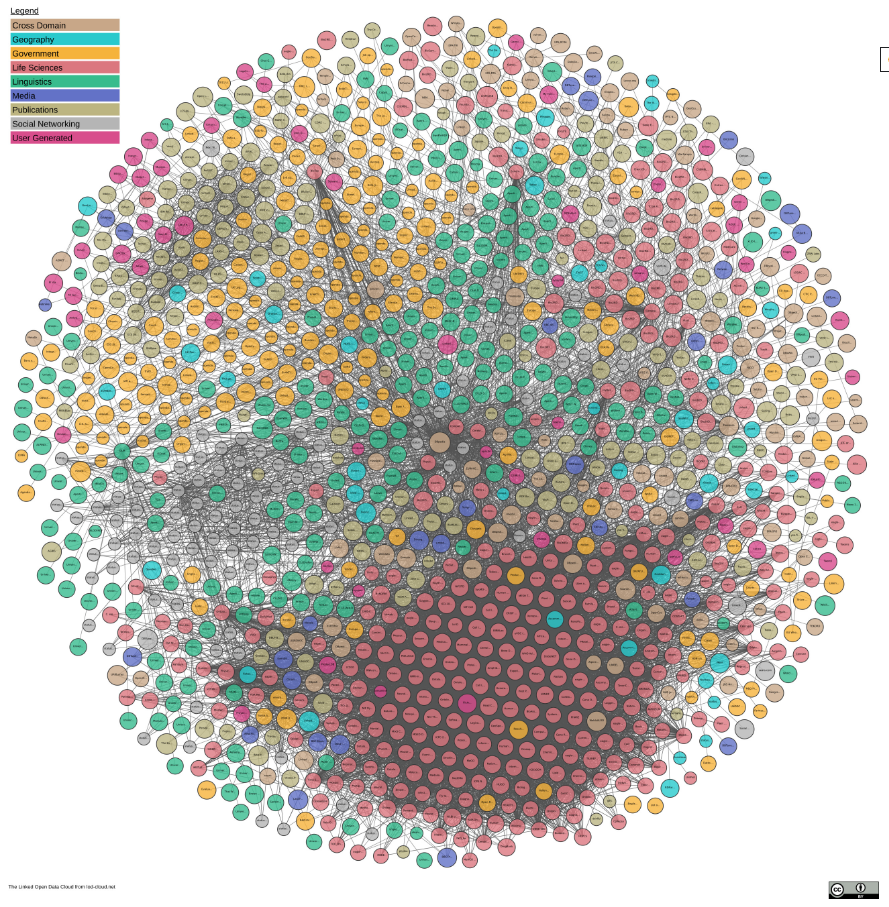


Figura 9 La nube de datos *Linked Data.*

## RDF (Resource Description Framework)

El *framework* de descripción de recurso, RDF por sus siglas en inglés, es un modelo estándar para el intercambio de datos en la web. La característica de RDF es que extiende las estructuras de enlaces de la web, al usar URI tanto para nombrar relaciones entre cosas como para los puntos finales de las relaciones, a veces referido como “*triple*”. El modelo RDF permite representar los datos y la relación existente entre ellos mediante ontologías a través de relaciones semánticas. La relación semántica que guardan los *RDF* es: sujeto, predicado y objecto. El sujeto y el predicado de un *triple* son URI que identifican a cada uno. El predicado especifica como el sujeto y el objeto están relacionados, y también es representado por un URI. Esta característica provee un modelo de datos basado en grafos [35]. La figura 11 muestra un ejemplo de cómo diferentes sitios se enlazan entre sí mediante sus respectivas URI. Los vértices son los objetos y sujetos mientras que las aristas son los predicados, cada una asociada a una URI.

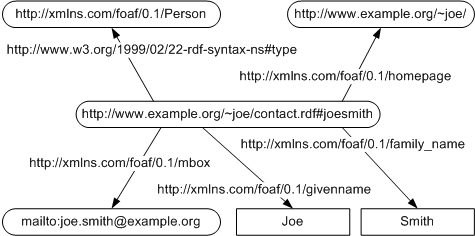


Figura 10 Grafo de RDF.

## URI (*Uniform Resource Identifier*)

Identificador de recursos uniforme, URI por sus siglas en inglés, es una cadena ASCII que identifica recursos de información en la Web Semántica.

Tal y como se observa en la figura 12, una URI puede estar compuesto de un localizador de recursos uniforme (URL, por sus siglas en inglés), de un nombre de recursos uniforme (URN, por sus siglas en inglés) o de ambos [36].

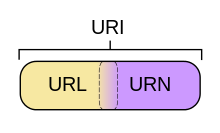


Figura 11 Diagrama de cómo está compuesto un URI.

Un ejemplo que se puede mostrar sobre la composición de una URI es la figura 13 en la que las letras en negritas denotan la ubicación donde el recurso está albergado como URL, y en letra normal, el nombre del recurso como URN.

**http://sitiointernet.com/autor/bibliografia.html**#posts

Figura 5 Ejemplo URI

## RDF *triple store*

Es un tipo de base de datos basada en grafos de tripletas RDF [37], por lo que, al ser una base de datos basada en grafos, el *triple store* puede ser vista como una red de objetos enlazados. El *triple store* al ser una herramienta de la Web Semántica, las entidades que conforman a la base de datos, tripletas RDF, son representadas como sujeto, predicado y objeto o también puede ser considerada como sujeto, predicado y etiqueta.

## SPARQL

*SPARQL* es un acrónimo para el Protocolo *SPARQL* y Lenguaje de Consultas RDF, por sus siglas en inglés, y es un protocolo y lenguaje de consultas para *Linked Data* en la web o bases de datos semánticas basadas en grafos (*RDF triple stores*) [3]. *SPARQL* está diseñado y respaldado por el consorcio de la web (W3C).

El siguiente código muestra cómo hacer una consulta *SPARQL* de los músicos mexicanos famosos que ya han muerto y que están sobre el *triple store* de DBpedia.

PREFIX dcterms: <http://purl.org/dc/terms/>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT ?musico ?nombreMusico ?fechaFallecimiento

WHERE {

?musico dcterms:subject

<http://dbpedia.org/resource/Category:Mexican\_musicians>;

rdfs:label ?nombreMusico ;

dbp:birthDate ?fechaNacimiento ;

dbp:deathDate ?fechaFallecimiento .

FILTER (LANG(?nombreMusico) = "es")

}

La figura 14 muestra el resultado de la consulta

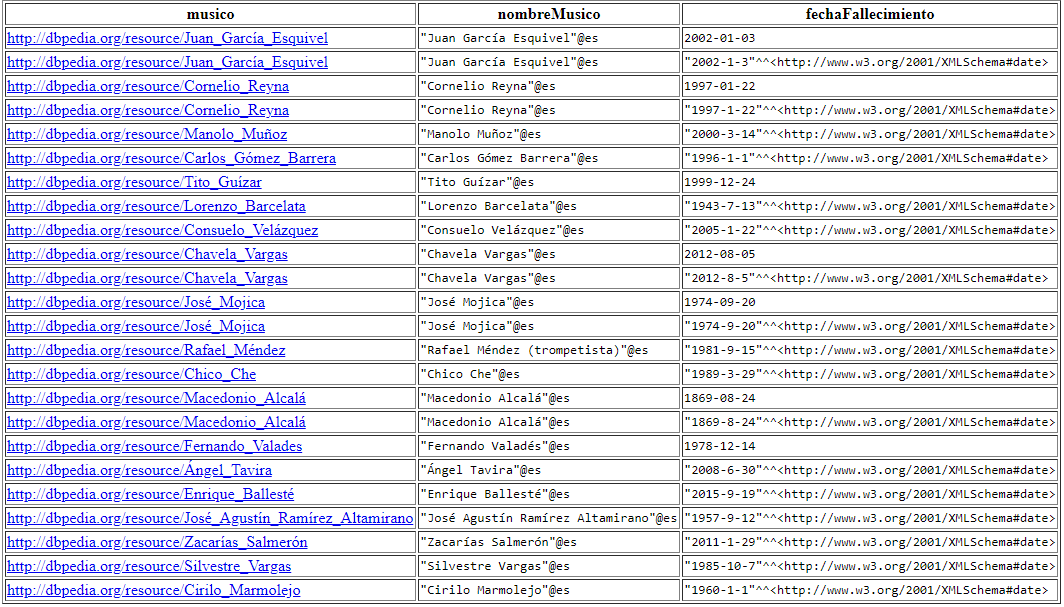


Figura 12 Resultado de la consulta en SPARQL.

Actualmente se encuentra en su segunda versión, *SPARQL* 1.1 [3], y ya describe una extensión para explícitamente delegar subconsultas a diferentes *SPARQL* *endpoint*. Esta característica es conocida como consulta federada.

## SPARQL endpoint

Se le denominar *SPARQL endpoint* al identificador único de recursos, URI por sus siglas en inglés, asociado al servidor HTTP que ofrece y devuelve peticiones HTTP para peticiones provenientes de clientes que usan el protocolo SPARQL [38]. Un ejemplo de *SPARQL endpoint* es DBpedia [39] el cual es uno de los *endpoints* más famoso ya que en ella se albergan aproximadamente 4.58 millones de objetos en su base de datos de conocimiento.

## GeoSPARQL

Es un lenguaje de consultas geográficas para datos RDF en la Web Semántica estandarizado por el *Open Geospatial Consortium* (OGC) [4]*.* *GeoSPARQL* define un vocabulario para la representación de datos en RDF y también define una extensión para el lenguaje de consultas geoespaciales *SPARQL*. Esta extensión de *SPARQL* para geo datos es útil en la solución de problemas de logística, hidrología y turismo [5].

*GeoSPARQL* busca relaciones topológicas entre objetos que posean datos asociados a una ubicación geográfica. Dicha búsqueda lo realiza mediante tres componentes principales:

* Definición de un vocabulario para representar características, geometrías y sus relaciones (ontologías).
* Conjunto de funciones espaciales para llevar a cabo consultas en *SPARQL.*
* Conjunto de reglas de transformación de consultas.

Un ejemplo de consulta es el siguiente: Determinar los objetos que estén completamente contenidos en la figura A de la figura 15.

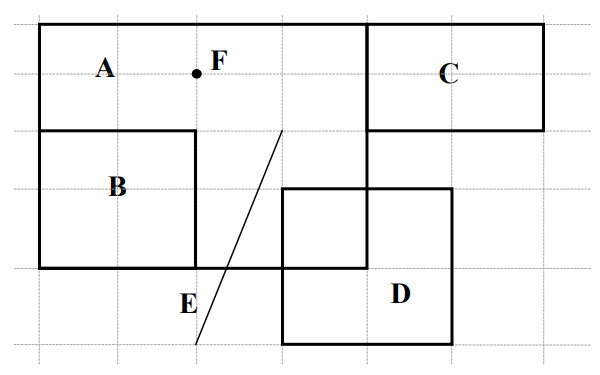


Figura 13 Representación de datos espaciales.

A continuación, la consulta *SPARQL* asociada para resolver ese interrogante:

PREFIX my: <http://example.org/ApplicationSchema#>

PREFIX geo: <http://www.opengis.net/ont/geosparql#>

PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?f

WHERE { my:A my:hasExactGeometry ?aGeom .

?aGeom geo:asWKT ?aWKT .

?f my:hasExactGeometry ?fGeom .

?fGeom geo:asWKT ?fWKT .

FILTER (geof:sfContains(?aWKT, ?fWKT) &&

!sameTerm(?aGeom, ?fGeom))

}

El resultado se muestra en la figura 16.



Figura 14 Resultado de la consulta en GeoSPARQL.

Como era de esperarse, el resultado dice que ‘A’ contiene tanto a ‘B’ como al punto ‘F’

## Arquitectura SOA

La arquitectura orientada a servicios, SOA por sus siglas en inglés, es un concepto en la ingeniería de software cuya intención es reducir costos de implementación, servicios para clientes innovadores, agilidad para adaptar cambios del sistema [40]. Sus características son las siguientes:

* Objetivos de negocio ligados a la infraestructura de tecnologías de información (TI).
* Orientada a la arquitectura de sistemas. Esto busca que la lógica de procesos de negocio no intervenga con la lógica del software de un sistema.
* Separación de objetivos: Dividir objetivos primarios en diferentes características con funcionalidades estrechas tan pequeñas como sean posible.
* Modularidad: La aplicación estará dividida en piezas distinguibles las cuales, desempeñarán una función en específico en el sistema.
* Bajo acoplamiento: Los atributos de los componentes de un sistema no tienen y/o no hacen uso del conocimiento de otros componentes independientes.
* Encapsulación: El acceso a datos con sus respectivas instrucciones de manipulación, las cuales estarán dentro de un paquete, es posible mediante una interfaz independiente.
* Interfaces: Implementación de un pequeño conjunto de interfaces que son mantenidas de manera separada.
* Mensajes: Uso de mensaje que contengan información a ser intercambiada a través de las interfaces mediante una estructura y vocabulario delimitado por un esquema.
* Reutilización: Es la acción de reutilizar un componente múltiples veces.
* Composabilidad: Capacidad de seleccionar componentes y ensamblarlos de diversas maneras que puedan cumplir el objetivo de la aplicación.

## Protocolo HTTP

El protocolo HTTP es un protocolo a nivel de aplicación para colaborar, distribuir sistemas de información de tipo hipermedia (texto, imagen, audio, video, mapas). La comunicación entre sistemas, según el protocolo, está basado en respuestas y peticiones [41].

* El cliente hace una petición HTTP.
* El servidor recibe la petición.
* El servidor procesa la petición.
* El servidor lleva a cabo una respuesta HTTP con la información solicitada o con un mensaje de error.
* El cliente recibe la respuesta.

Así mismo, el protocolo HTTP establece un grupo de métodos de petición, los cuales indican que acción se quiere llevar a cabo sobre un recurso específico. Los métodos son los siguientes [42].

* GET: Pide una representación de un recurso específico. Este método solo debe ser capaz de recuperar datos.
* POST: Crea un nuevo recurso en el servidor.
* PUT: Las representaciones actuales de un recurso son sustituidas por la información que lleva el mensaje, y en caso de no existir, lo crea.
* DELETE: Elimina un recurso determinado.
* HEAD: Petición similar a la de GET, pero sin cuerpo de la respuesta.
* CONNECT: Establece comunicación en 2 vías con el servidor del recurso solicitado. Comúnmente es usado para llevar a cabo una comunicación túnel.
* TRACE: Lleva a cabo un mensaje de prueba de ida y vuelta en toda la ruta hasta el recurso objetivo con el fin de ser un mecanismo de depuración.
* OPTIONS: Método que permite describir las opciones de comunicación para un recurso en específico.
* PATCH: Permite modificar un recurso de manera parcial, a diferencia del método PUT que lo hace sobre todo el recurso.

## REST

La transferencia de estado representacional, REST por sus siglas en inglés, es un tipo de arquitectura que define reglas de comunicación entre sistemas computacionales en la Web [43].

A pesar de que *REST* no es un estándar, si hace uso de ellos

* URL.
* HTTP.
* XML, GIF, JPG, etc. (representación de recursos).
* Extensiones multipropósito de correo de Internet (MIME).

Las características de *REST* son:

* Basado en cliente servidor: Consumo de componentes mediante peticiones.
* Sistemas independientes: Los sistemas computacionales no deben de estar desarrollados en el mismo lenguaje o paradigma de programación mientras estos cumplan el estilo de arquitectura *REST* se puede estableces comunicación entre los sistemas.
* Uso de memoria caché: Con el fin de mejorar la eficiencia de respuestas, los sistemas deben de ser capaces de decidir si las respuestas son o no parte de la memoria caché.
* Recursos etiquetados: Los sistemas deben de identificar a sus recursos con una URL.
* Interfaz uniforme: Cualquier recurso del sistema puede accederse mediante una interfaz genérica. Ejemplo: métodos HTTP.
* Representación de recursos enlazados: Las representaciones de los recursos deben de estar conectados entre sí para que el usuario sea capaz de ir de una representación a otra.
* Capas entre los componentes: Con el fin de otorgar servicios al usuario como seguridad, privacidad, eficiencia de servicio, entre otros, pueden usarse sistemas intermediarios como *gateways,* servidores *proxy* o servidores *caché* por mencionar algunos.

## JSON

La notación de Objetos de JavaScript, *JSON* por sus siglas en inglés, es un formato de intercambio de información basado en texto e independiente de lenguaje. El formato fue un derivado del estándar del lenguaje de programación *ECMAScript* [44].

*JSON* está conformado por las siguientes 2 estructuras:

* Colección de pares nombre/valor: En diversos lenguajes de programación implementan esta estructura como un objeto, diccionario, registro, tabla *hash*, arreglos asociativos o lista de claves.
* Lista ordenada de valores: Se implementa en los lenguajes de programación como vectores, arreglos, secuencias o listas.

Ejemplo de un objeto *JSON* es

{

"Image": {

"Width": 800,

"Height": 600,

"Title": "View from 15th Floor",

"Thumbnail": {

"Url": "http://www.example.com/image/481989943",

"Height": 125,

"Width": 100

},

"Animated" : false,

"IDs": [116, 943, 234, 38793]

}

}

## Traductor

En informática un traductor es un programa que convierte un programa fuente en un programa objeto.



Fig. 9 Traductor.

## Compilador

Es un traductor que convierte lenguaje de medio o alto nivel a lenguaje de máquina o ensamblador. Por ejemplo: C, Java, Pascal, Fortran, entre otros.

## Partes de un compilador

Si bien los compiladores pueden clasificarse en función a su construcción como de una, dos, múltiples pasadas, en función de su construcción o de alguna característica especial. Sin embargo, generalmente se dividen en 2 partes: análisis y síntesis.

La parte de análisis está compuesta por:

* Analizador léxico o *scanner*: Encargado de analizar el código fuente y separar las palabras mediante token los cuales son un conjunto de caracteres como pueden ser palabras reservadas, variables, símbolos aritméticos, de relación entre otros. Elimina comentarios y los tokens son almacenados en una tabla de símbolos.
* Analizador sintáctico o *parser*: Encargado de analizar la correcta agrupación de los tokens en frases gramaticales las cuales pueden ser representadas mediante un *parse tree*.
* Analizador semántico: Encargado de verificar la consistencia de las expresiones y/o significado del código mediante la localización de errores.

La parte de síntesis está compuesta por:

* Generador de código intermedio: Encargado de llevar acabo equivalencias entre el código fuente y código máquina. Usualmente se usan como representación la notación de postfijos, árbol de direcciones de código o un árbol de sintaxis.
* Optimizador de código: Encargado de reducir el código generado por el generador de código intermedio para reducir el número de recursos y mejora la velocidad de ejecución.
* Generador de código: Genera código ensamblador o código ejecutable de tal forma que la máquina pueda interpretar el código fuente [45].

## Jerarquía de lenguajes:

La jerarquía de lenguajes se refiere al nivel de dependencia que tiene un cierto lenguaje con la máquina o procesador. Los lenguajes se clasifican de la siguiente forma:

* Lenguajes de máquina: Son los nativos de cada computadora y describen la arquitectura de esta. Las instrucciones en lenguajes de máquina se representan mediante códigos numéricos. La comunicación entre los lenguajes de máquina y la máquina no son necesarios.
* Lenguajes ensambladores: Son conjuntos de símbolos que representan unívocamente a cada código numérico del lenguaje de máquina asociado. El nombre que reciben las instrucciones de este tipo de lenguajes es nemónico.
* Lenguajes de nivel medio: Son lenguajes que permiten el uso eficiente de lenguajes ensambladores y también sobre las estructuras de control de flujo, de iteración y el orden de un lenguaje de alto nivel.
* Lenguajes de alto nivel de uso general: Este tipo de lenguajes permiten abstraerse de la estructura interna de la computadora al manipular las estructuras de control de flujo, de iteración y secuencia.
* Lenguajes de propósito específico: Son lenguajes de alto nivel que solucionan problemas especiales en determinadas tareas como PostgreSQL para manejar bases de datos.
* Lenguajes de inteligencia artificial: Estos lenguajes permiten declarar explícitamente conocimiento y las estructuras de control de flujo, de iteración y secuencia no son expresadas en el programa por lo que está abstraída la lógica del control de la computadora.

# Capítulo IV: Análisis del sistema

El análisis de requerimientos llevado a cabo en este proyecto está basado en el estándar IEEE 830-1998 [46]. En este capítulo se presenta la descripción del proyecto especificando qué es lo que debe de hacer.

## Descripción general

Se presenta una perspectiva de lo que debe de hacer el producto, las características de los usuarios del proyecto, qué restricciones existen, así como las suposiciones y dependencias que se asumen para que el proyecto funcione.

## Perspectiva del producto

El módulo de consultas que se desarrolló ha permitido que el software Apache Marmotta tenga la capacidad de realizar consultas federas con el objetivo de devolver una respuesta de datos geoespaciales unificada de los diversos *triple store* que se especifiquen en la consulta ingresada.

La aplicación Web permite llevar a cabo las consultas federadas geoespaciales y usa *GeoYASGUI* como herramienta externa para visualizar e interactuar con los resultados en un mapa.

## Características de los usuarios

A continuación, se presentan las características que los 2 tipos de usuarios que pueden hacer uso del sistema.

Tabla 3 Característica de usuario normal.

|  |  |
| --- | --- |
| Tipo de usuario | Normal |
| Formación | Estudiante, profesor y/o investigador |
| Habilidades | Conocimiento *SPARQL* y *GeoSPARQL* |
| Actividades | Iniciar sesión, consultar, visualizar datos, filtrar datos. |

Tabla 4 Características de usuario administrador.

|  |  |
| --- | --- |
| Tipo de usuario | Administrador |
| Formación | Estudiante, profesor y/o investigador |
| Habilidades | Uso de aplicación Web |
| Actividades | Iniciar sesión, administrar usuarios. |

## Restricciones

* Conexión a Internet
* Consultas basadas en los protocolos *SPARQL 1.1* y *GeoSPARQL*
* Las consultas pueden ser federadas
* Consultas a la nube *Linked Data*
* El lenguaje de programación para el desarrollo del módulo que fue implementado en Apache Marmotta es Java.
* Los lenguajes de programación y de marcado para la plataforma fueron
  + HTML
  + CSS
  + JavaScript
  + Java
* La aplicación está basada en métodos HTTP, en REST y JSON.
* Los resultados de las consultas poseen un LIMIT que permite limitar el número de resultados por recuperar que a su vez están asociados a la cantidad de memoria RAM disponible en el sistema. Si no se contara con este mecanismo, es probable que el sistema no pudiera procesar toda la información que Apache Marmotta devolviera al momento de hacer una consulta.

## Suposiciones y dependencias

* Se consideró que la herramienta *GeoYASGUI* al momento de hacer el presente proyecto terminal seguiría usando las tecnologías mencionadas para la aplicación Web.
* Se asumió que el módulo de consultas federadas estuviera ya desarrollado cuando la aplicación Web estuviera lista.

## Requisitos específicos

En esta sección se muestran los requerimientos funcionales y no funcionales que contiene el tanto el módulo de consultas como el de la aplicación Web.

## Requerimientos funcionales

Los requerimientos funcionales para la aplicación Web son los siguientes:

Tabla 5 Requerimiento funcional establecer comunicación

|  |  |
| --- | --- |
| Número de requisito | RF01 |
| Nombre de requisito | Establecer comunicación |
| Tipo | Obligatorio |
| Características | El usuario debe de inicializar la aplicación Web para poder usarla. |
| Descripción del requerimiento | Se debe de establecer comunicación entre el software Apache Marmotta e Internet con la aplicación Web ya que, sin ellos la aplicación Web no puede funcionar. |
| Requerimiento no funcional | RNF01  RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 6 Requerimiento funcional validar conexión.

|  |  |
| --- | --- |
| Número de requisito | RF02 |
| Nombre de requisito | Validar conexión |
| Tipo | Obligatorio |
| Características | La aplicación Web debe validar la conexión. |
| Descripción del requerimiento | Para usar la aplicación Web, se debe de conectar la aplicación con Apache Marmotta e Internet con el fin de avanzar o mostrar un mensaje de error de conexión en la aplicación. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

Tabla 7 Requerimiento funcional selección modo operación.

|  |  |
| --- | --- |
| Número de requisito | RF03 |
| Nombre de requisito | Selección modo operación |
| Tipo | Obligatorio |
| Características | El usuario deberá escoger el modo de uso de la aplicación: Consultar *dataset* o realizar consulta. |
| Descripción del requerimiento | La aplicación Web da la opción de escoger 2 opciones de operación. En la primera, existen varios *datasets* precargados en Apache Marmotta mientras que, en la segunda opción, el usuario puede ingresar una consulta que quiera ser ejecutada en tiempo real. Los resultados de ambas opciones se pueden visualizar en la aplicación Web. |
| Requerimiento no funcional | RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 8 Requerimiento funcional modo *dataset*.

|  |  |
| --- | --- |
| Número de requisito | RF04 |
| Nombre de requisito | Modo *dataset* |
| Tipo | Obligatorio |
| Características | La aplicación Web muestra los *datasets* disponibles en Marmotta. |
| Descripción del requerimiento | Los *datasets* disponibles en Apache Marmotta son desplegados en la aplicación Web para que usuario escoja cual quiere explorar. |
| Requerimiento no funcional | RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 9 Requerimiento funcional selección *dataset*.

|  |  |
| --- | --- |
| Número de requisito | RF05 |
| Nombre de requisito | Selección *dataset* |
| Tipo | Obligatorio |
| Características | El usuario puede escoger los *dataset* disponibles en el sistema para luego ser visualizados |
| Descripción del requerimiento | El sistema Web en conjunto con Apache Marmotta, muestra los *datasets* disponibles de consultas federadas previas y posteriormente, con la herramienta *GeoYASGUI*, visualizar e interactuar con los resultados de la consulta asociada a dicho *dataset*. |
| Requerimiento no funcional | RNF01  RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 10 Requerimiento funcional cargar datos.

|  |  |
| --- | --- |
| Número de requisito | RF06 |
| Nombre de requisito | Cargar *dataset* |
| Tipo | Obligatorio |
| Características | Se carga el *dataset* en la aplicación Web. |
| Descripción del requerimiento | Cuando el usuario escoja cuál *dataset* quiere explorar, la aplicación pide los datos a Apache Marmotta para que sean cargados en la aplicación Web. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

Tabla 11 Requerimiento funcional modo consulta.

|  |  |
| --- | --- |
| Número de requisito | RF07 |
| Nombre de requisito | Modo consulta |
| Tipo | Obligatorio |
| Características | La aplicación Web muestra una caja de texto donde el usuario ingresará una consulta y también ofrece ayuda que le permita al usuario menos experto construir una consulta federada. |
| Descripción del requerimiento | En este modo, el usuario debe ingresar una consulta que en una caja de texto cargada en la aplicación Web o en dado caso que el usuario no sea experto, el usuario puede pedir ayuda al sistema para escribir la consulta federada. |
| Requerimiento no funcional | RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 12 Requerimiento funcional enviar consulta.

|  |  |
| --- | --- |
| Número de requisito | RF08 |
| Nombre de requisito | Enviar consulta |
| Tipo | Obligatorio |
| Características | Botón que le permite al usuario enviar la consulta que quiera ser ejecutada. |
| Descripción del requerimiento | Una vez terminada la tarea de escribir una consulta por parte del usuario, se debe enviar la consulta escrita a Apache Marmotta mediante la acción de hacer click en un botón. |
| Requerimiento no funcional | RNF01  RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 13 Requerimiento funcional validar consulta.

|  |  |
| --- | --- |
| Número de requisito | RF09 |
| Nombre de requisito | Validar consulta |
| Tipo | Obligatorio |
| Características | Validará la consulta enviada a Apache Marmotta. |
| Descripción del requerimiento | El compilador de Apache Marmotta es capaz de identificar qué tipo de consulta fue la que se ingresó. La respuesta puede ser: federada, normal o error.  La aplicación muestra un mensaje de error en caso de que la consulta enviada haya sido incorrecta. |
| Requerimiento no funcional | RNF01  RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 14 Requerimiento funcional recibir resultados.

|  |  |
| --- | --- |
| Número de requisito | RF10 |
| Nombre de requisito | Recibir resultados |
| Tipo | Obligatorio |
| Características | La aplicación Web recibe los resultados de la consulta enviada por el usuario. |
| Descripción del requerimiento | Si la consulta enviada a Apache Marmotta no tuvo algún error, la aplicación Web recibe los datos provenientes de Apache Marmotta. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

Tabla 15 Requerimiento funcional visualizar datos.

|  |  |
| --- | --- |
| Número de requisito | RF11 |
| Nombre de requisito | Visualizar datos |
| Tipo | Obligatorio |
| Características | El usuario puede visualizar los resultados de la consulta ingresada o *dataset* selección*.* |
| Descripción del requerimiento | Una vez que Apache Marmotta haya terminado de hacer la consulta o de cargar el *dataset*, la aplicación web debe dar la opción al usuario de visualizar los datos usando la herramienta *GeoYASGUI*. |
| Requerimiento no funcional | RNF01  RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 16 Requerimiento funcional explorar datos.

|  |  |
| --- | --- |
| Número de requisito | RF12 |
| Nombre de requisito | Explorar datos |
| Tipo | Obligatorio |
| Características | La aplicación Web carga la herramienta *GeoYASGUI.* |
| Descripción del requerimiento | Cuando se tenga el *dataset* seleccionado cargado o los resultados de la consulta, la aplicación Web carga la herramienta *GeoYASGUI* con la que el usuario puede visualizar e interactuar con los datos. |
| Requerimiento no funcional | RNF01  RNF02  RNF03 |
| Prioridad del requisito | Alta/Esencial |

Tabla 17 Requerimiento funcional iniciar de sesión.

|  |  |
| --- | --- |
| Número de requisito | RF13 |
| Nombre de requisito | Iniciar sesión |
| Tipo | Obligatorio |
| Características | El usuario debe de iniciar sesión para acceder al sistema. |
| Descripción del requerimiento | Para poder hacer uso del sistema de exploración de datos, el usuario debe ingresar su usuario y contraseña. |
| Requerimiento no funcional | RNF03  RNF04 |
| Prioridad del requisito | Alta/Esencial |

Tabla 18 Requerimiento funcional iniciar de sesión.

|  |  |
| --- | --- |
| Número de requisito | RF14 |
| Nombre de requisito | Validar nuevo usuario |
| Tipo | Obligatorio |
| Características | Se corrobora que el correo y contraseña sean correctos. |
| Descripción del requerimiento | Para dar acceso al sistema, se verifica en el *backend* de la aplicación Web, que el correo y contraseña sean correctos para brindar el acceso a la aplicación Web. |
| Requerimiento no funcional | RNF03  RNF04 |
| Prioridad del requisito | Alta/Esencial |

Tabla 19 Requerimiento funcional validar usuario.

|  |  |
| --- | --- |
| Número de requisito | RF15 |
| Nombre de requisito | Registrar usuarios |
| Tipo | Obligatorio |
| Características | El administrador de la aplicación Web debe de registrar a los usuarios en la aplicación Web para que puedan acceder a cualquier funcionalidad de esta. |
| Descripción del requerimiento | Para que los usuarios puedan acceder a la aplicación Web, ellos tendrán que proporcionar correo electrónico, una contraseña y nivel de experiencia en SPARQL y GeoSPARQL para que el administrador de la aplicación los registre en el sistema. |
| Requerimiento no funcional | RNF03  RNF04 |
| Prioridad del requisito | Alta/Esencial |

Tabla 20 Requerimiento funcional validar nuevo usuario.

|  |  |
| --- | --- |
| Número de requisito | RF16 |
| Nombre de requisito | Validar nuevo usuario |
| Tipo | Obligatorio |
| Características | El *backend* debe corroborar si los datos del nuevo usuario son válidos. |
| Descripción del requerimiento | Cada vez que el administrador de la aplicación Web ingrese datos para registrar a un nuevo usuario, el *backend* de la aplicación corrobora que el correo ingresado no exista. |
| Requerimiento no funcional | RNF01  RNF02  RNF03  RNF04 |
| Prioridad del requisito | Alta/Esencial |

Tabla 21 Requerimiento funcional dar de baja a usuario.

|  |  |
| --- | --- |
| Número de requisito | RF16 |
| Nombre de requisito | Dar de baja a usuario |
| Tipo | Obligatorio |
| Características | Mediante la aplicación Web, el administrador puede dar de baja al usuario. |
| Descripción del requerimiento | El administrador debe tener la posibilidad de dar de baja mediante el correo y contraseña del usuario para que posteriormente el usuario no pueda acceder de nuevo a la aplicación. |
| Requerimiento no funcional | RNF01  RNF02  RNF03  RNF04 |
| Prioridad del requisito | Alta/Esencial |

Los requerimientos funcionales para el módulo de consultas federadas son los siguientes:

Tabla 22 Requerimiento funcional cargar consulta federada.

|  |  |
| --- | --- |
| Número de requisito | RF17 |
| Nombre de requisito | Cargar consulta federada |
| Tipo | Obligatorio |
| Características | Apache Marmotta le hace llegar la consulta federada al módulo de consultas que se va a desarrollar. |
| Descripción del requerimiento | Para que se puedan extraer los URI y argumentos de la consulta federada, la consulta proveniente del *SPARQL endpoint* debe de llegar al módulo de consultas federadas. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

Tabla 23 Requerimiento funcional extracción URI y argumentos.

|  |  |
| --- | --- |
| Número de requisito | RF18 |
| Nombre de requisito | Extracción URI y argumentos |
| Tipo | Obligatorio |
| Características | Se extraen los URI y argumentos de la consulta federada |
| Descripción del requerimiento | Para poder llevar a cabo la consulta federada, se deben de extraer los URI de los *triple store* y argumentos de la consulta que se van a realizar. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

Tabla 24 Requerimiento funcional Consulta *triple store*.

|  |  |
| --- | --- |
| Número de requisito | RF19 |
| Nombre de requisito | Consulta *triple store* |
| Tipo | Obligatorio |
| Características | Se consulta el *triple store* asociado a cada URI con sus respectivos argumentos. |
| Descripción del requerimiento | Para llevar a cabo una consulta federada, se debe de consultar a todos los *triple store* extraídos de la consulta federada para obtener respuestas con base a los argumentos también extraídos de la consulta. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

Tabla 25 Requerimiento funcional procesar resultados.

|  |  |
| --- | --- |
| Número de requisito | RF20 |
| Nombre de requisito | Procesar resultados |
| Tipo | Obligatorio |
| Características | Los resultados retornados por los diferentes *triple store* son unificados. |
| Descripción del requerimiento | El módulo desarrollado procesa todas las respuestas de cada *triple store* consultado con el fin de eliminar resultados repetidos y devolver una respuesta unificada. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

Tabla 26 Requerimiento funcional guardar datos.

|  |  |
| --- | --- |
| Número de requisito | RF21 |
| Nombre de requisito | Guardar datos |
| Tipo | Obligatorio |
| Características | Los resultados de las consultas federadas se guardan en el *kiwi triple store* para que el usuario pueda usarlos posteriormente. |
| Descripción del requerimiento | Por cada consulta federada exitosa, Apache Marmotta guarda el resultado de la consulta en el *triple store* de Marmotta, *kiwi triple store*, para que pueda posteriormente hacer uso de los datos de su consulta. |
| Requerimiento no funcional | RNF01 |
| Prioridad del requisito | Alta/Esencial |

## Requerimientos no funcionales

Tabla 27 Requerimiento no funcional Disponibilidad.

|  |  |
| --- | --- |
| Número de requisito | RNF01 |
| Nombre de requisito | Disponibilidad |
| Características | El software Apache Marmotta e Internet deben estar accesibles cuando un usuario de la la aplicación Web lo requiera. |
| Descripción del requerimiento | Cada vez que la aplicación Web necesite hacer uso de algún recurso de Apache Marmotta o de Internet, deben de estar disponibles para hacer uso de ellos. |
| Prioridad del requisito | Alta/Esencial |

Tabla 28 Requerimiento no funcional Usabilidad.

|  |  |
| --- | --- |
| Número de requisito | RNF02 |
| Nombre de requisito | Usabilidad |
| Características | El usuario cuenta con alguna leyenda que proporcione información de cómo usar la aplicación Web o llevar a cabo una consulta. |
| Descripción del requerimiento | Debido a los requerimientos funcionales disponibles en la aplicación Web, se muestra información en la aplicación en donde sea pertinente de tal forma que le permita al usuario entender cómo debe de ser usada la aplicación.  También ofrece ayuda para llevar a cabo una consulta federada en caso de que el usuario haya ingresado una consulta federada de manera incorrecta. De igual forma, la aplicación Web ayuda al usuario menos experto a construir una consulta federada mediante el despliegue de mensajes en la misma página que ofrezca ayuda al usuario, mejor conocido como *tooltips*. |
| Prioridad del requisito | Alta/Esencial |

Tabla 29 Requerimiento no funcional Interfaz de la aplicación.

|  |  |
| --- | --- |
| Número de requisito | RNF03 |
| Nombre de requisito | Interfaz de la aplicación |
| Características | Todas las funcionalidades que ofrece la aplicación Web se podrán acceder desde la interfaz visual. |
| Descripción del requerimiento | La interfaz visual en la aplicación web permitirá al usuario usar la aplicación en conjunto con Apache Marmotta de manera sencilla. |
| Prioridad del requisito | Alta/Esencial |

Tabla 30. Requerimiento no funcional Confidencialidad

|  |  |
| --- | --- |
| Número de requisito | RNF04 |
| Nombre de requisito | Confidencialidad |
| Características | La seguridad de los datos proporcionados por los usuarios está garantizada en el sistema. |
| Descripción del requerimiento | La información ofrecida por los usuarios para darse de alta en el sistema está segura para que nadie más pueda hace uso de ella. |
| Prioridad del requisito | Alta/Esencial |

## Interfaces de hardware

Para hacer uso de la aplicación Web es necesario tener equipo de cómputo con las siguientes especificaciones:

* 8 GB en memoria RAM.
* 500 MB de almacenamiento.
* Mouse.
* Teclado.
* Adaptador de Red.
* Monitor.
* Procesador doble núcleo o superior.

## Interfaces de software

Para poder usar en la aplicación Web, se debe contar con las siguientes características (AM y AppWeb):

* Sistema operativo Ubuntu.
* Java JDK 6 o superior.
* PostgreSQL
* MySQL
* Express JS
* Leaflet
* Servidor de aplicaciones Java (Tomcat 7.X o Jetty 6.X).
* Navegador Web (Google Chrome, Firefox, Edge, Safari u Opera).

## Interfaces de comunicación

La comunicación entre la aplicación Web y Apache Marmotta es mediante métodos HTTP y el estilo de arquitectura de software es *SOA* y *REST.* Mientras que, para hacer uso de la aplicación Web, se necesita conexión a Internet.

# Capítulo V: Diseño del sistema

En este capítulo se detalla cómo se implementó el módulo de consultas federadas en Apache Marmotta y cómo se desarrolló la aplicación Web. El diseño se empieza abordando el cómo y en donde se debía implementar el módulo en Apache Marmotta desde un punto de vista de compiladores. Después se presentan las clases e interfaz que se implementaron en el módulo de consultas federadas con su respectiva justificación. Por otra parte, se presenta el diagrama que describe la arquitectura de la aplicación Web, fragmentos de código y sus diagramas respectivos.

## Diseño del módulo de consultas como compilador.

En el marco teórico se indicó cómo es que funciona un compilador ya que la implementación del código de consultas se asemeja a implementar una funcionalidad de un compilado y a continuación se entenderá el porqué.

El desarrollo del proyecto se empezó por entender cómo es que un objeto es capaz de recibir una cadena de texto con una serie de instrucciones y luego tener la capacidad de procesarlo de tal forma que diera como resultado lo indicado en dicha cadena. Lo anterior surge a partir de la necesidad de saber cómo es que se piensa implementar una nueva funcionalidad en Apache Marmotta quien a partir de consultas SPARQL, es capaz de procesarlas, extraer información de los repositorios necesarios y luego devolver el resultado de la consulta.

El paso que siguió en el desarrollo fue la manera de modificar el código de Apache Marmotta de tal manera que se le diera una nueva funcionalidad. Tomando como referencia a MySQL, dicho lenguaje está construido sobre código C y C++ [<https://dev.mysql.com/doc/connector-cpp/8.0/en/connector-cpp-introduction.html>], Apache Marmotta Marmotta está desarrollado en Java. Después la pregunta que surgió ¿En qué parte había que modificar el código de Apache Marmotta?

Comprendido lo anterior, ¿Qué significa que Marmotta esté escrito en otro lenguaje que no es *SPARQL*? Significa que las funcionalidades presentes en Marmotta están desarrolladas y compiladas con Java por lo que había que editar el código de su repositorio para poder implementar el módulo de consultas.

En esta parte del desarrollo es donde los compiladores hicieron sentido ya que tanto Apache Marmotta como MySQL usaban lenguajes ajenos para su desarrollo y los lenguajes que se emplean en su desarrollo son compilados. Con base al marco teórico, los lenguajes se clasifican en las siguientes categorías:

* Lenguajes naturales.
* Lenguajes de inteligencia artificial.
* Lenguajes de propósito específico.
* Lenguajes de medio nivel de uso general.
* Lenguajes de nivel medio.
* Lenguajes ensambladores.
* Lenguajes de máquina.

Con base al protocolo *SPARQL*, se llegó a la conclusión que *SPARQL* es un lenguaje de programación de propósito específico ya que fue desarrollado para procesar datos en formato RDF y que, a partir de enunciados de tipo sujeto, predicado y objeto, hacer consulta a bases de datos cuya información alojada sea RDF, es decir, a *triple stores*. El software Apache Marmotta utiliza el lenguaje Java para llevar a cabo la ejecución de todas sus funcionalidades.

Citando nuevamente el marco teórico, entre los lenguajes de programación existen los compilados y los interpretados y dado la característica de Apache Marmotta y el uso de Java para su desarrollo, se concluye que *SPARQL* en Marmotta es compilado.

El paso siguiente fue averiguar cómo es que el sistema de Apache Marmotta funciona. El estilo de arquitectura de software de Apache Marmotta está inspirado en el modelo de software Orientado a Servicios [***referencia a página de Marmotta***], SOA por sus siglas en inglés, por lo que había que determinar cómo es que las distintas funcionalidades de Marmotta conviven entre ellas. Apache Marmotta está desarrollado con la herramienta Maven, la cual es un sistema de administración y compresión de Software, con la que todos los paquetes [***Ref a MAVEN***], clases, interfaces [ADDDD MOOORE] se organizan y conviven en el sistema. [***ADD cómo se conecta unos con los otros***.]

El siguiente paso fue determinar exactamente en qué parte del código de Apache Marmotta había que modificar para implementar la característica por lo que se procedió a comprender cómo es que Marmotta funciona.

La pregunta que surgió después fue ¿Qué implica que *SPARQL* sea un lenguaje de propósito específico y que Apache Marmotta use Java para ejecutar todas las instrucciones declaradas en una consulta SPARQL? La respuesta es que el módulo de consultas federadas tenía que ser empleado en un compilador por lo que había que editar el código de un compilador para realizar el proyecto.

La conclusión anterior permitió darle un enfoque distinto al proyecto ya que solo era cuestión de entender cómo es que un compilador funciona y después extrapolarlo al sistema Apache Marmotta.

La tarea siguiente fue comprender cómo funciona un compilador. Un compilador está construido básicamente en 2 partes que a su vez se dividen en 3 partes cada uno y son:

* Analizador
  + Analizador léxico o *Scanner*.
  + Analizador sintáctico o *Parser.*
  + Analizador semántico.
* Síntesis.
  + Conversor de código intermedio.
  + Optimizador de código.
  + Generador de código.

En la siguiente figura se muestra un diagrama de bloques que muestra cómo es que se organiza un compilador.

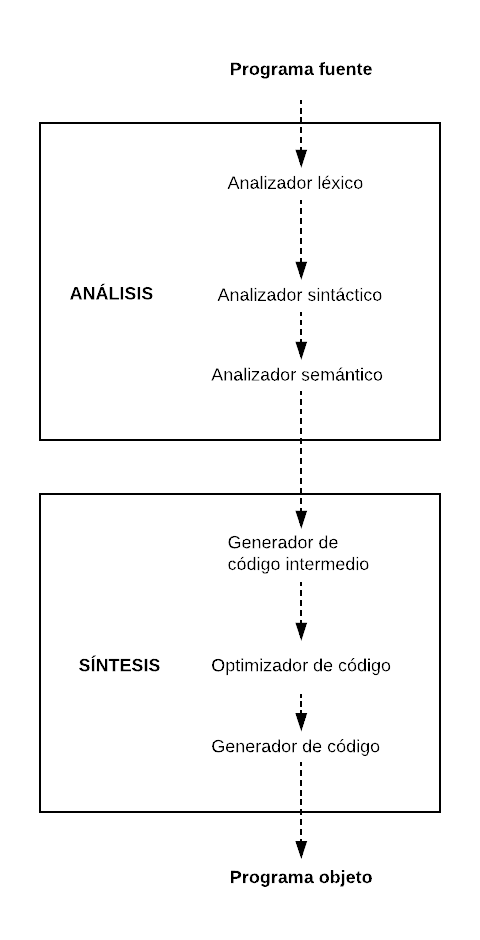
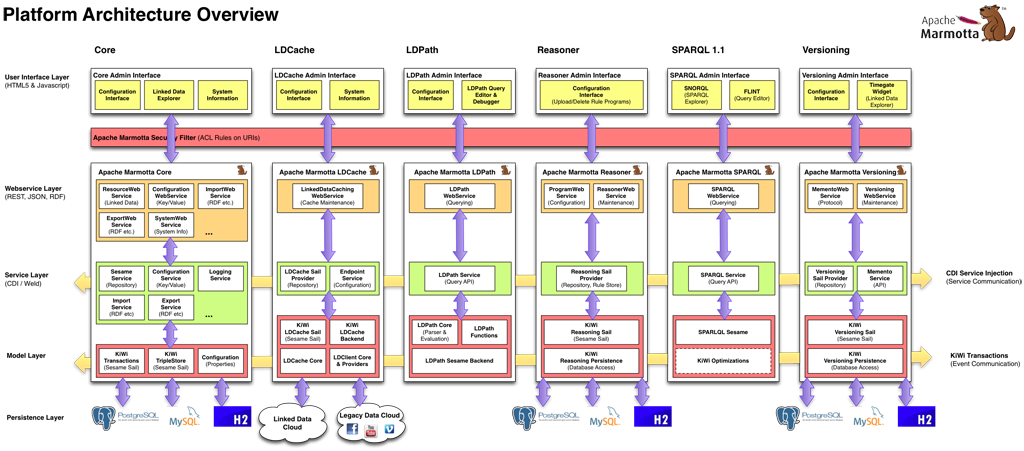


Figura 15 Diagrama a bloques de un compilador.

Tras una serie de preguntas y respuestas con la persona José Ortiz de la Universidad de la Cuenca en Ecuador, quien fue la persona asignada por el asesor Luis al alumno, indicó que a nivel compilador la sección en la que se debía de trabajar es en la sección de análisis ya que ahí es donde se identifican las palabras reservadas, que los tipos de datos sean los adecuados y que la sintaxis sea correcta. Por otra parte, para la sección de la funcionalidad se debía de trabajar en la clase de Apache Marmotta encargada de la evaluación de las consultas. Tal como se mencionó, Apache Marmotta usa Maven como herramienta de organización por lo que solo fue cuestión de averiguar cuál es el paquete de Marmotta encargado de evaluar las consultas.

Después de la búsqueda en el proyecto y basándose en la figura ***PLATFORM\_ARCHITECTURE\_OVERVIEW*** [<https://marmotta.apache.org/platform/index.html>] se concluyó que el paquete encargado de evaluar las consultas es: *evaluation* el cual está contenido en otros paquetes, tal como se observa en la figura ***ARBOL\_PAQUETES\_AM***.



Una vez averiguado lo anterior, se procedió a la implementación del módulo de consultas federadas. Se procedió a preguntar nuevamente a José Ortiz el cómo se podía empezar a realizar dicha actividad y comentó que Apache Marmotta usa código de OpenRDF[***REFERENCIA\_OPENRDF***] en su sistema por lo que me sugirió que hiciera el mismo análisis que se hizo en Apache Marmotta, pero ahora en OpenRDF ya que OpenRDF sí tiene la característica de hacer consultas federadas.

El *triple store* Apache Marmotta, tal y como se mencionó previamente, su estilo de arquitectura está inspirado en SOA, por lo que solamente había que trabajar con el paquete adecuado para agregar una nueva funcionalidad. Sin embargo, para poder implementar código en Marmotta hay que cumplir ciertas características para poder editar cualquier cosa.

Las características están delimitadas por las interfaces de la clase quienes son las encargadas de definir las firmas de los métodos que deben de existir en las clases que implementen dichas interfaces. Entiéndase a la palabra firma a la declaración de los métodos pero sin cuerpo del código.

La implementación de tales archivos parte de la clase existente de Apache Marmotta llamada ***KiWiEvaluationStrategy.java*** ya que dicha clase es la encargada de evaluar las consultas. Originalmente contaba con NNNN clases de las cuales, ninguna se encarga de procesar consultas federadas.

IMAAAAAAAGEN

La clase hace uso de un concepto que existe en el paradigma de programación orientado a objetos y es la sobrecarga de métodos, es decir, en una misma clase pueden existir más de un método con el mismo nombre pero con diferentes argumentos. Tal característica es la que se aprovechó en el proyecto ya que algo característico de los métodos de la clase KiWiEvaluationStrategy es que el tipo de datos que regresa después de evaluar una consulta son los mismos por lo que solo fue cuestión de definir qué argumentos son los que el método de consultas federadas debería recibir.

Nuevamente, se tuvo una plática con José Baculima para preguntar qué tipo de datos recibe la clase KiWiEvaluationStrategy y de qué forma se podía empezar a implementar código. Él sugirió que se revisara cómo es que OpenRDF procesa las consultas federadas ya que Apache Marmotta hace uso de librerías de OpenRDF para su estrategia de evaluación, entonces restaría determinar qué tipo de datos

Posteriormente, haciendo el mismo análisis de compilador en OpenRDF, se determinó que el la librería encargada de la estrategia de evaluación es la sesame-queryalgebra-evaluation. Ahí se halla una clase similar a la de KiWiEvaluationStrategy y contiene el método encargado de hacer consultas avanzadas en OpenRDF. Los datos que recibe tal método son:

* Service service.
* BindingSet bindings.

A consecuencia de que Apache Marmotta usa OpenRDF como un auxiliar de funcionalidades, la implementación del método de consultas federadas se pudo hacer sin demasiado inconveniente al tener casi el mismo tipo de firma de los métodos presentes en Apache Marmotta a excepción de los argumentos del método.

Consecuentemente, solo faltaba definir las clases e interfaces necesarias por extraer de OpenRDF y luego implementarlas en Apache Marmotta.

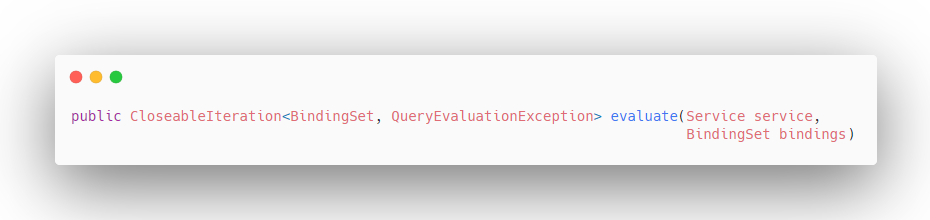
Se tuvieron que implementar las siguientes clases e interfaz en un nuevo paquete llamado ConsultasFederadas:

* Clase: FederatedServiceManager.java.
* Clase: ServiceFallbackIteration.
* Clase: SPARQLFederatedService.
* Interfaz: FederatedService.

A continuación se muestran los método más representativos de cada uno y su justificación para ser incluidos en el proyecto.

### Evaluador de consulta

El método que se encarga de empezar a evaluar las consultas federadas en la clase KiWiEvaluationStrategy tiene, al igual que los demás método de la clase, *evaluate* y su firma se muestra en la siguiente fracción de código.



Fragmento de código 1 Firma de método encargado de evaluar consultas federadas.

El método de que se encarga de evaluar la consulta federada hace uso de 1 clase y de la única interfaz del presente proyecto y son:

* Clase FederatedServiceManager.
* Interfaz FederatedService.

El método recibe 2 argumentos:

* Service service: Objeto que contiene toda la información de la sesión de la conexión entre Apache Marmotta y el *SPARQL endpoint* remoto.
* BindingSet bindings: Objeto que contiene todos los enunciados de la consulta federada.

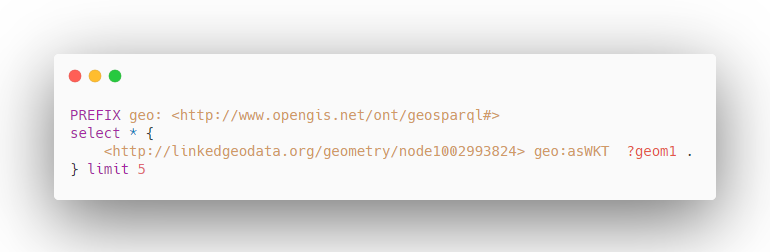
Lo primero que se hace en el método *evaluate* es extraer toda la información del SPARQL endpoint remoto como es la referencia al objeto que establece la conexión al repositorio remoto, extracción de la URI de tal servicio.

Luego, se crea una instancia con todos sus métodos y atributos de la clase importada FederatedServiceManager a través de la URI que fue previamente extraída y es en esta acción en la que se hace uso de la Interfaz FederatedService y de la clase FederatedServiceManager ya que la interfaz es la encargada de definir qué métodos son válidos de la clase FederatedServiceManager, el cual, debido a la implementación del presente proyecto, son todos. En la siguiente fracción de código se muestra la creación y asignación dela instancia del repositorio remoto.



Fragmento de código Variable que contiene la referencia del SPARQL endpoint.

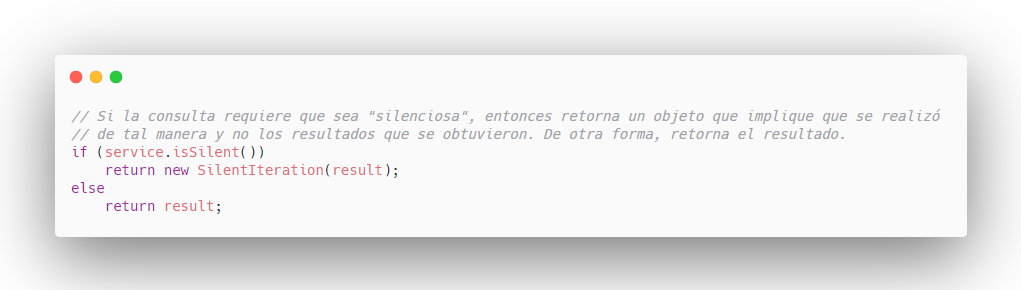
Después, se manipulan los *bindings* para determinar si son variables libres o no. Las variables libres son aquellas que no son requeridas por la consulta y estas pueden ser variables intermedias y finales que no son seleccionadas en la sentencia *SELECT*. Posteriormente, se extraen aquellos nodos que son definidos previamente, es decir, en aquellos casos en que el usuario define valores que son constantes y no variable. Por ejemplo, en el siguiente fragmento de código se muestra una consulta cuyo objeto y predicado están definidos por URI ya establecidas.



Fragmento de código 2 Ejemplo de consulta SPARQL con valores previamente definidos.

Después, a partir de la cantidad de variables libres, se determina si la consulta es de tipo ASK o SELECT. Si el ASK, significa que la cantidad de variables libres es cero y es tal cantidad ya que las variables que no son descartadas es cero, y debido a que la instrucción ASK consiste en recolectar toda la información que se ejecute en la consulta, la instrucción ASK corresponde tal escenario. Si es distinto el escenario, significa que se está preguntando solamente por algunas variables por lo que habrá variables libres en el proceso.

En caso de que la consulta sea SELECT, se puede silenciar la consulta mediante la palabra reservada SILENT con la intención de no retornan ningún resultado y ninguna excepción. Cuando tal caso ocurre, se encapsula el resultado de mediante la clase SilentIteration. El siguiente fragmento de código muestra cómo se lleva a cabo.



Fragmento de código 3 Código encargado de regresar los resultados de manera silenciosa o normal.

Lo anterior es lo más representativo del método *evaluate* ya que se muestran sus funciones características y cómo es que tiene interacción con una clase e interfaz del proyecto, sin embargo, hay más funciones que no se muestran en esta sección del documento por cuestiones estéticas del mismo, así que, si se require ahondar más sobre el código de *evaluate* en la clase KiWiEvaluationStrategy, se sugiere ir al anexo **####** donde podrá ver todo el código de la clase y Con el fin de dar un panorama más general del método *evaluate*, se realizó el diagrama de flujo del método y de sus funciones auxiliares el cual puede ser visitado en el anexo **####.**

## Diagrama de paquete

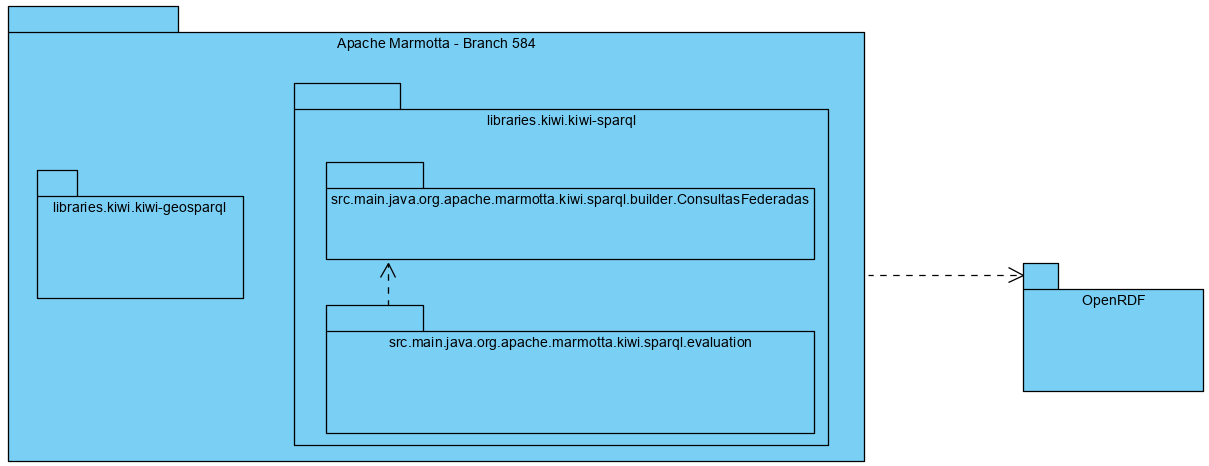
En la presente sección se muestran los diagramas de paquete que representan las estructuras de las dependencias entre los diferentes paquetes del módulo de consultas federadas y los paquetes de Apache Marmotta. Y de igual forma, se muestra en un diagrama las dependencias que se tiene entre la aplicación Web y librerías que fueron necesarias para su desarrollo.

### Diagrama de paquetes – Apache Marmotta y OpenRDF

Apache Marmotta basa parte de su código en el *triple store* OpenRDF por lo que es común ver importación de paquetes de dicha librería en las diversas clases de Apache Marmotta.

Lo primero que se observa de la siguiente figura es que el *triple store* Apache Marmota depende del *triple store* Open RDF al tener una flecha cortada que parte del paquete Apache Marmotta cuya rama es la 584, ya que es la que cuenta con las funcionalidades de *GeoSPARQL*, hacia el paquete OpenRDF ya que cuenta con ciertas funciones que Apache Marmotta no posee.

Lo último por destacar es que el módulo de consultas federadas y el módulo que se encarga de evaluar las consultas tienen en común el mismo paquete padre y esto es debido ambos pertenecen al protocolo *SPARQL* 1.1 y al estilo de arquitectura de software que Apache Marmotta usa.



### Diagrama de paquetes – Aplicación Web

El siguiente diagrama muestra qué dependencias tiene la aplicación Web para que funcione. Las tecnologías base de la aplicación Web son JavaScript, SQL, HTML y CSS sin embargo, se tuvieron que usar librerías para simplificar el desarrollo de la aplicación Web y cumplir los requerimientos tanto funcionales como no funcionales.

Para el servidor Web se usó JavaScript y el *framework* llamado *Express* ya que *Express* ofrece la capacidad de administrar las diferentes páginas de la aplicación usando un sistema de ruteo por URL, capacidad de realizar consultas SQL para registrar, consultar y eliminar usuarios, *middleware* para gestionar el acceso a ciertas funcionalidades de la Aplicación Web y peticiones HTTP para establecer comunicación con Apache Marmotta. Además, se usó la el *framework* bCrypt que permite cifrar las contraseñas usando el algoritmo MD5 con el fin de no guardar contraseñas en MySQL de forma insegura.

Para la visualización se consideraron varias opciones como Map4RDF, Triply o Sophox ya que están enfocados en visualización de datos geoespaciales tipo SPARQL. Sin embargo, ninguna de ellas se adaptaba a los requerimientos del sistema por lo que se tuvieron que considerar opciones distintas. Las 2 que se consideraron ya que cumplían con los requerimientos del sistema fueron ArcGis mediante su API de JavaScript y Leaflet que es una herramienta *OpenSource* y que también cumple con las especificaciones del sistema. Al final, se decidió usar Leaflet debido a que es software libre y ser también una dependencia que se emplea de manera exitosa en la herramienta de visualización Sophox.

***DIAGRAMOTE***

## Caso de uso

Los diagramas de caso de uso sirven para especificar el comportamiento esperado de un sistema sin ahondar cómo se llevará a cabo. El uso de este tipo de diagramas permite representar a nivel visual y textual el diseño del sistema desde la perspectiva del usuario final.

En los diagramas de caso de uso representa la relación existente entre sistemas, usuarios y actores, pero no muestra el orden de cómo es que será ejecutado cada caso de uso.

El siguiente es el único caso de uso, la cual es para la aplicación Web. Es el único ya que es la sección del proyecto que tiene una interacción directa con los usuarios finales. El caso de uso para la aplicación Web es la mostrada en la figura 17.

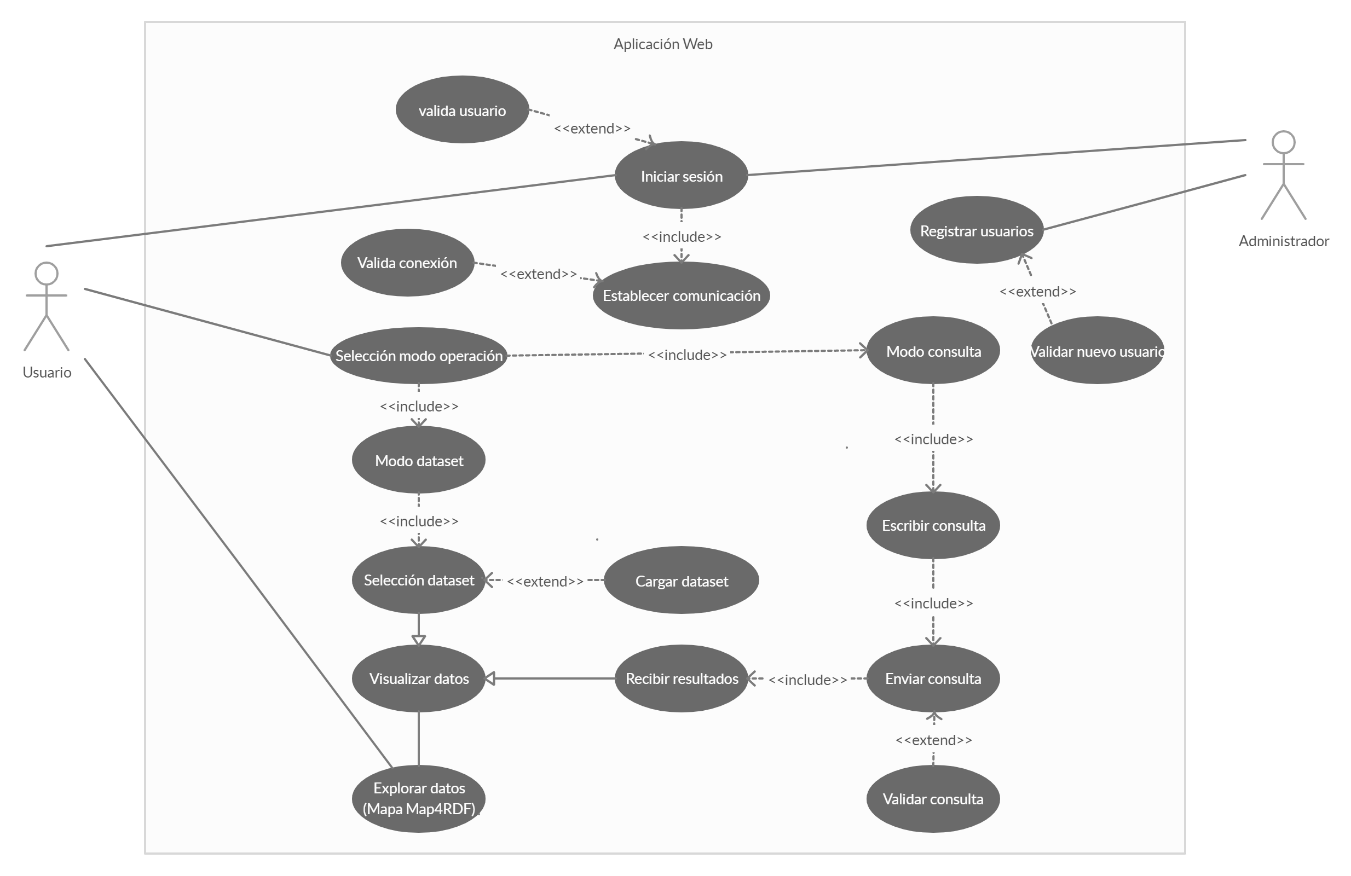


Figura 16 Caso de uso del diseño del sistema.

## Diagramas de clase

Los diagramas de clase son diagramas estáticos de estructura de un sistema que permite visualizar y construir sistemas orientados a objetos el cual muestra sus clases, métodos, atributos además de la relación que existen entre los objetos.

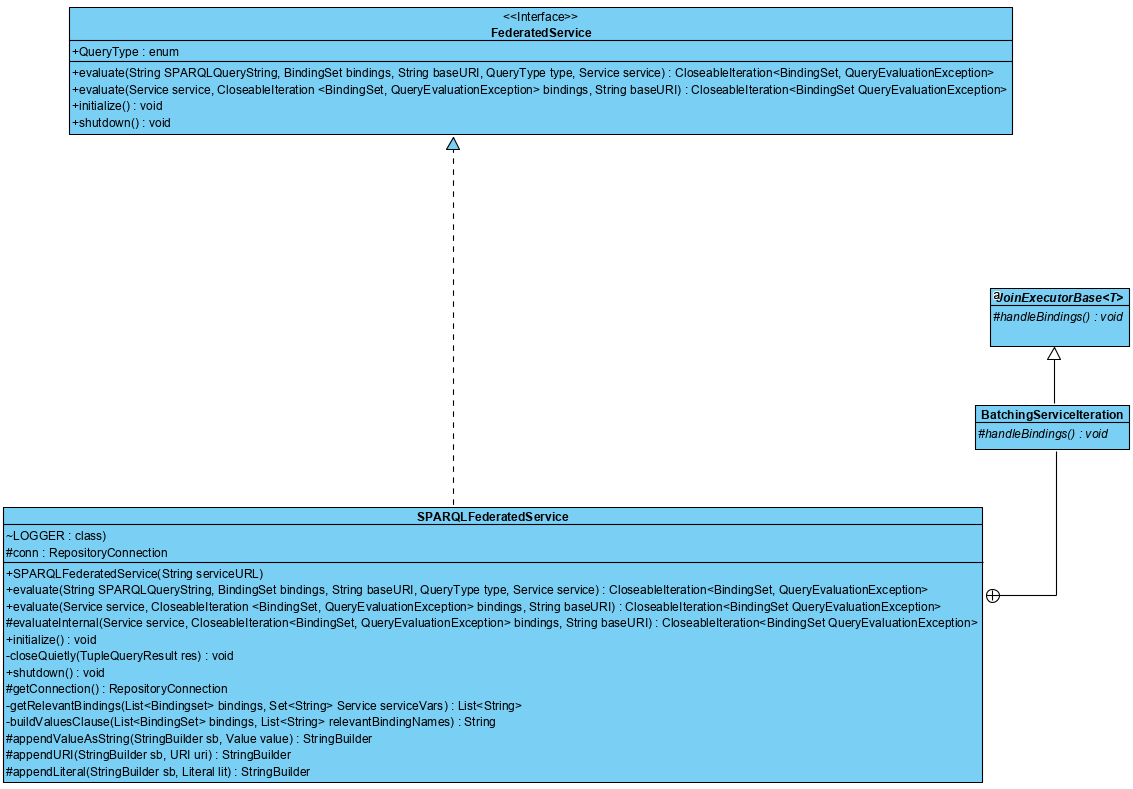
Los siguientes diagramas de clase corresponden a los diagramas de Apache Marmotta y de la aplicación Web.

### Apache Marmotta – SPARQLFederatedService

La siguiente figura muestra la interacción que tiene la clase SPARQLFederatedService.java con la interfaz FederatedService y la extensión de la misma clase con la clase BatchingServiceIteration.

Tal y como se describe en el marco teórico, la interfaz FederatedService determina la firma de los métodos y el argumento con el que se cuenta para aquella clase que la implementa, que este caso es SPARQLFederatedService y es esa la razón por la que se observan los mismos 4 métodos de la interfaz en la clase.

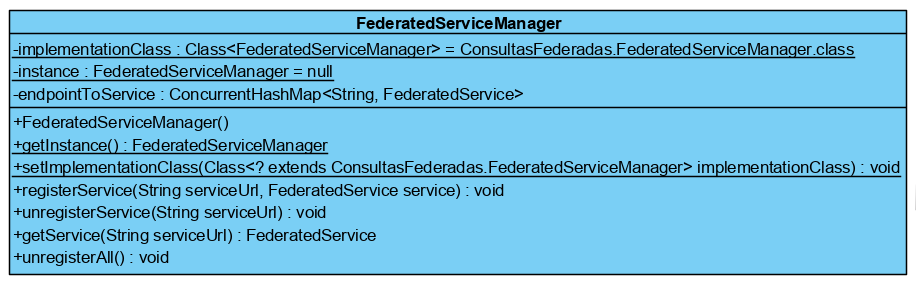
Esta clase posee 2 atributos y 13 métodos.



### Apache Marmotta – FederatedServiceManager

La figura XX representa a la clase encargada de administrar la consulta federada. La función de la presente clase es crear, registrar, desactivar y monitorear sesiones que asocien las URI de los *SPARQL endpoint* solicitados en la consulta federada y establecer las conexiones con Apache Marmotta.

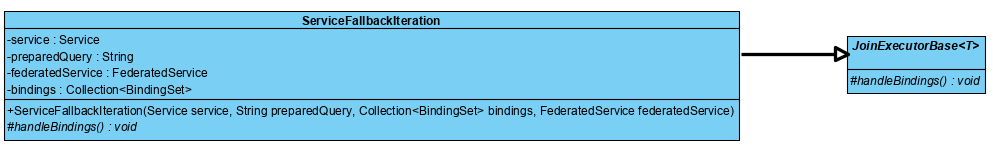
Esta clase posee 3 atributos y 7 métodos.



### Apache Marmotta – ServiceFallbackIteration

En la siguiente figura se muestra la clase ServiceFallbackIteration y es la encargada de ejecutar la consulta federada.

La clase ServiceFallbackIteration posee 4 atributos y 2 métodos.



### Aplicación Web

A continuación, en la figura 21, se muestra el diagrama de clases para la aplicación Web. En ella se maneja la herencia de una clase, la cual es “Usuario” que generaliza a dos tipos de usuario: normal y administrador. Cada uno de ellos tiene sus métodos propios; el usuario “Estudiante” posee el atributo *nivel\_usuario* que le permite al sistema reconocer si el usuario que esté haciendo uso del sistema es un usuario experto o no. La escala va del 0 al 5 donde el cero indica que el usuario no tiene conocimiento de cómo construir una consulta federada geoespacial y cinco, que indica que el usuario está completamente familiarizado con el desarrollo la consulta. Sin embargo, ambos comparten los atributos de correo y contraseña, y el método iniciar sesión que sirven para acceder a los elementos de la aplicación Web.

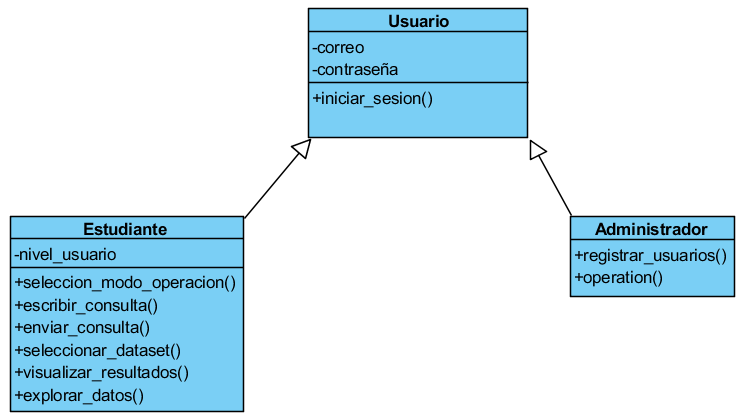


Figura 17 Diagrama de clases.

## Diagramas de estado

Los diagramas de estado son un tipo de diagrama que permite representar los diversos estados de una entidad no solo como una consecuencia de las entradas sino también de sus estados previos. De igual forma, muestra como un sistema responde a diferentes eventos dependiendo del cambio de un estado a otro.

A diferencia de los diagramas pasados, en esta sección se muestran los diagramas de estado que existen en la aplicación Web y en Apache Marmotta con el módulo de consultas federadas geoespaciales que se implementó.

### Diagrama de estado para la aplicación web – Usuario

En la figura 19 se muestra el diagrama de estados presente en la aplicación Web. Con este diagrama se puede ver cómo el usuario común puede avanzar en la aplicación considerando el inicio de sesión, el selección de modo de operación y la visualización de datos.

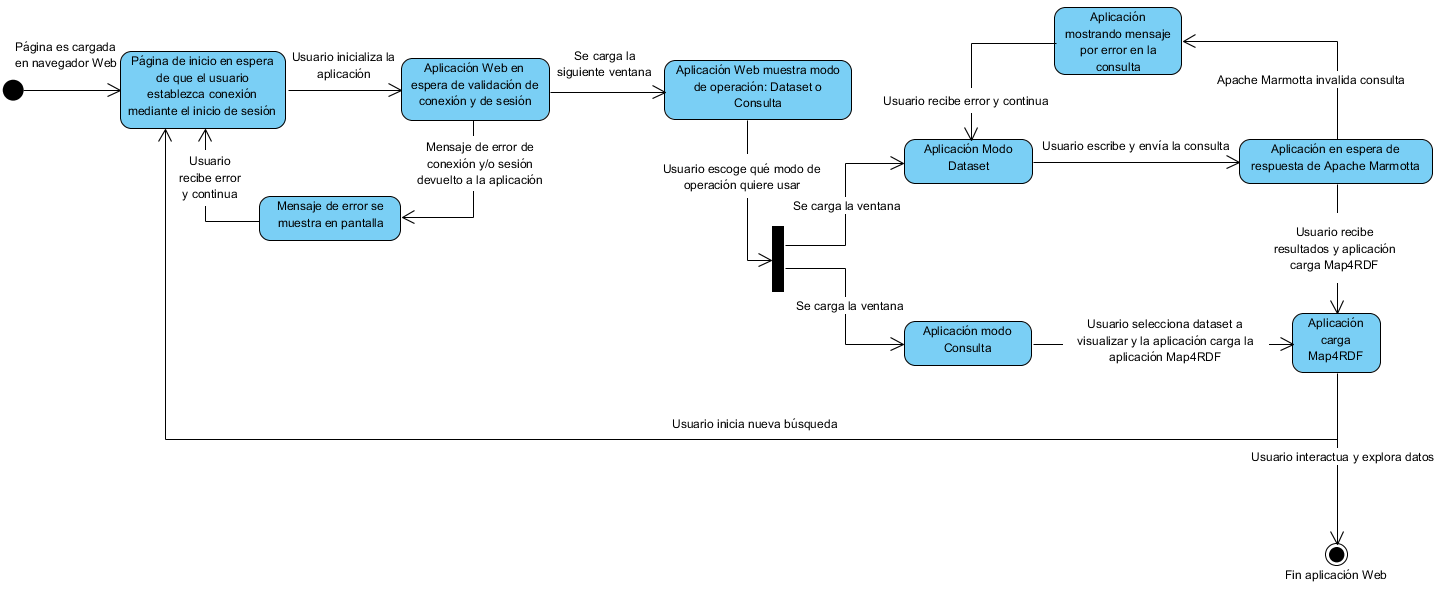


Figura 18 Diagrama de estados - aplicación Web – usuario.

### Diagrama de estado para la aplicación web – Administrador

El diagrama de estados para el usuario administrador, la cual se muestra en la figura 20, detalla los estados de la aplicación Web cuando el usuario administrador está usándola.

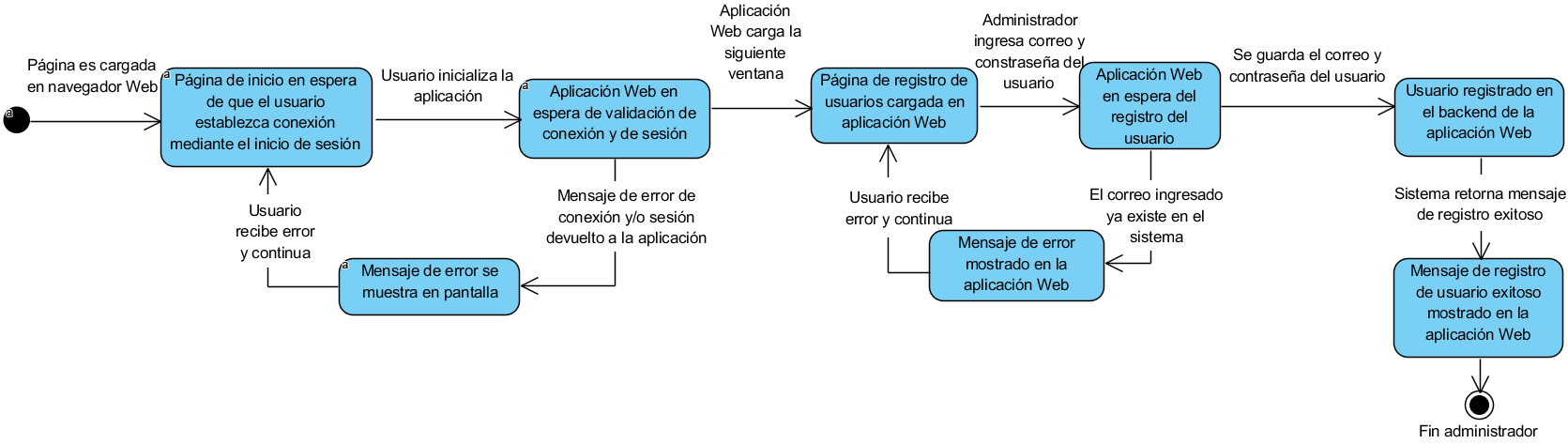


Figura 19 Diagrama de estados - Aplicación Web Administrador.

### Diagrama de estado para módulo de consultas en Apache Marmotta

Para el módulo de consultas federadas geoespaciales que se implementó en Apache Marmotta, se diseñó el siguiente diagrama de estados que propone cómo es que el módulo reacciona a las entradas y a los estados previos. Si bien el bloque azul es el módulo que se desarrolló, los demás objetos también son elementos por considerar ya que hay que modificar el respectivo elemento de Apache Marmotta que procesa la consulta para que sea capaz de identificar si es una consulta federada o no (Figura 21).

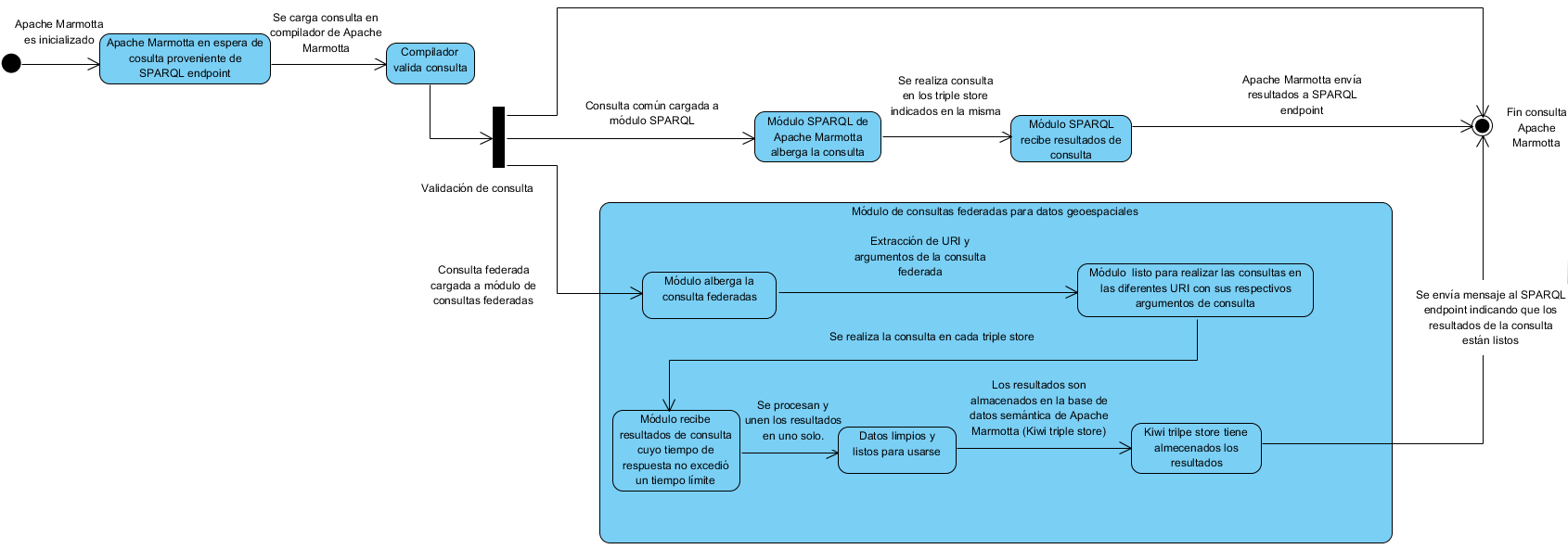


Figura 20 Diagrama de estados - Módulo de consultas federadas.

## Diagramas de secuencia

Los diagramas de secuencia son un tipo de diagramas de interacción que especifica como los objetos interactúan entre sí. En estos diagramas se observan las interacciones que existen entre los objetos involucrados en el sistema mediante mensajes, que van de un objeto a otro o a sí mismos, y columnas que representan el tiempo que los objetos están presentes en el sistema. Cabe decir que a pesar de que estos diagramas involucran el tiempo, no implica la especificación de cuánto tiempo debe de existir entre mensajes.

### Diagrama de secuencia para aplicación web (Usuario).

En el siguiente diagrama de secuencia, figura 22, muestra la interacción que existe entre el usuario común y la aplicación Web. Con la ayuda del diagrama de casos de uso, se visualiza como es que el usuario puede navegar en la aplicación Web desde el inicio de sesión hasta la visualización de datos.

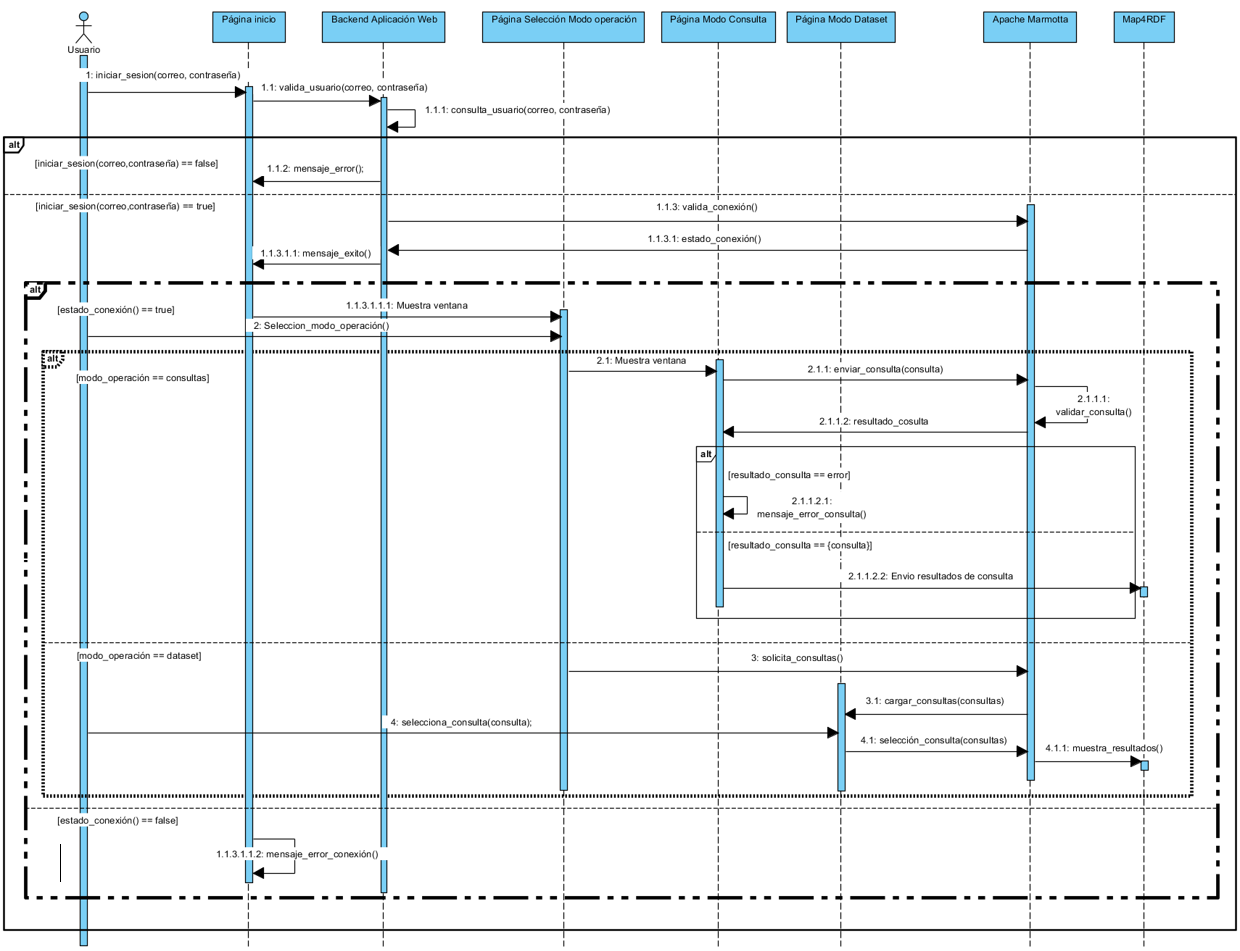


Figura 21 Diagrama de secuencia - aplicación Web – usuario común.

Con el fin de mostrar a mayor detalle el diagrama de secuencia mostrado en la figura 23, en las siguientes figuras se muestra el mismo diagrama de secuencia, pero de manera separada mostrando la referencia al siguiente diagrama.

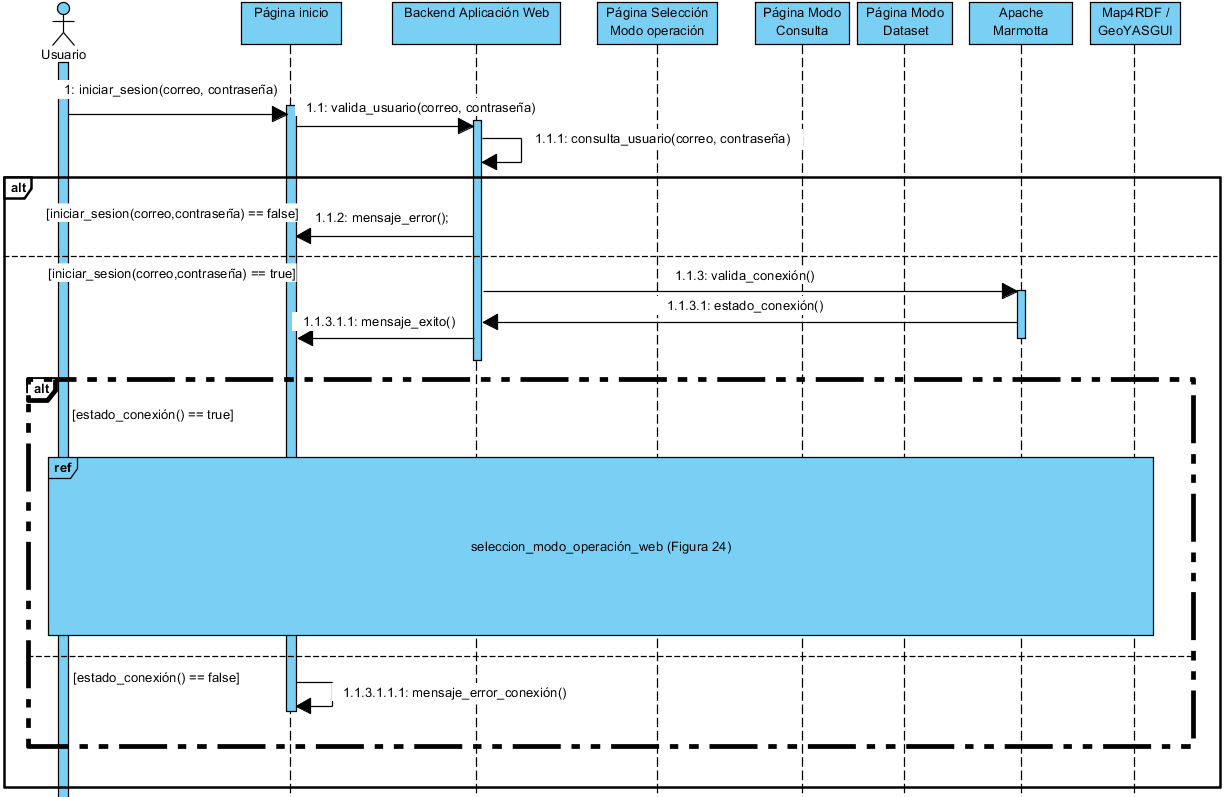


Figura 22 Diagrama de secuencia - Aplicación Web - usuario común.

En la figura 24 se encuentra el diagrama de secuencia que corresponde a la interacción del usuario con la aplicación Web al seleccionar un modo de operación.

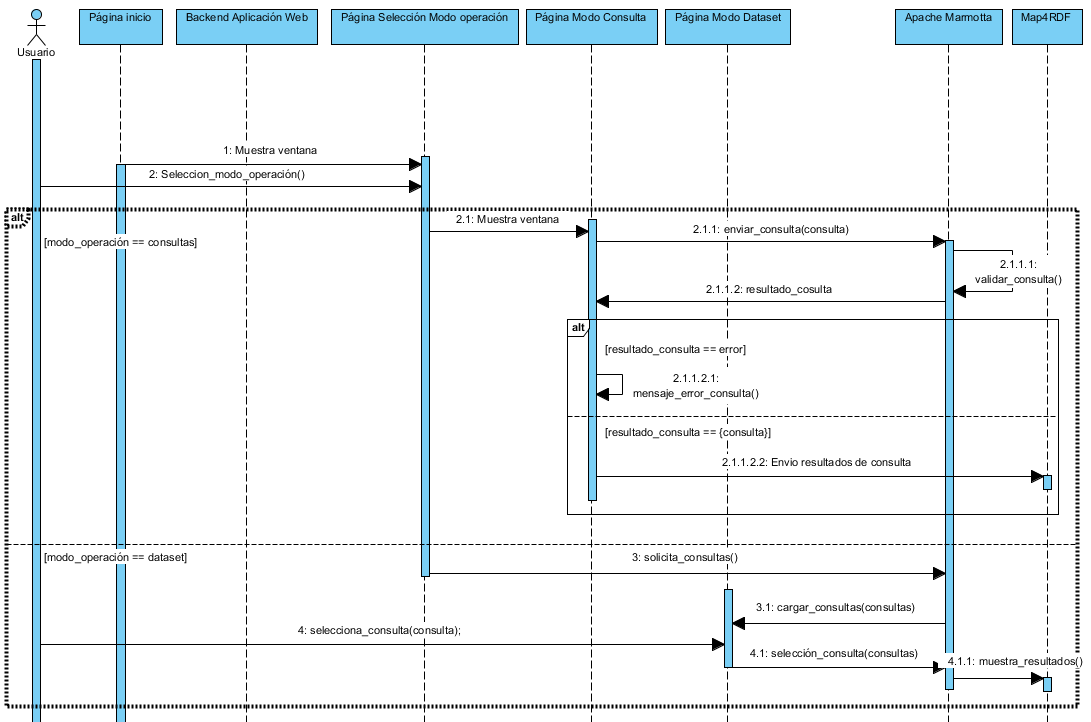


Figura 23 Diagrama de secuencia - Aplicación Web - usuario común, continuación.

### Diagrama de secuencia para aplicación Web (Administrador)

Para el administrador de la aplicación Web se tiene el diagrama de secuencia en la figura 25. El diagrama muestra el proceso del administrador para iniciar sesión y para registrar a un nuevo estudiante.

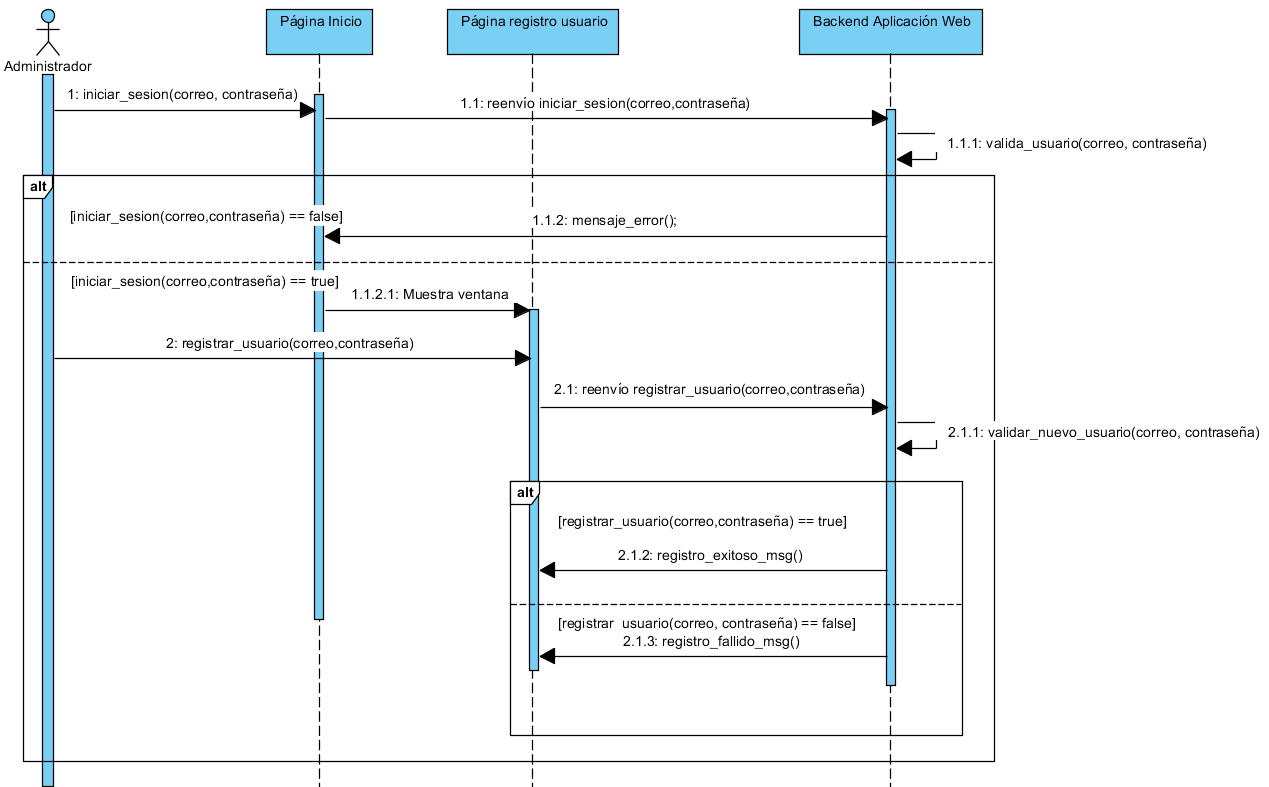


Figura 24 Diagrama secuencia aplicación Web - Administrador.

### Diagrama de secuencia para el módulo.

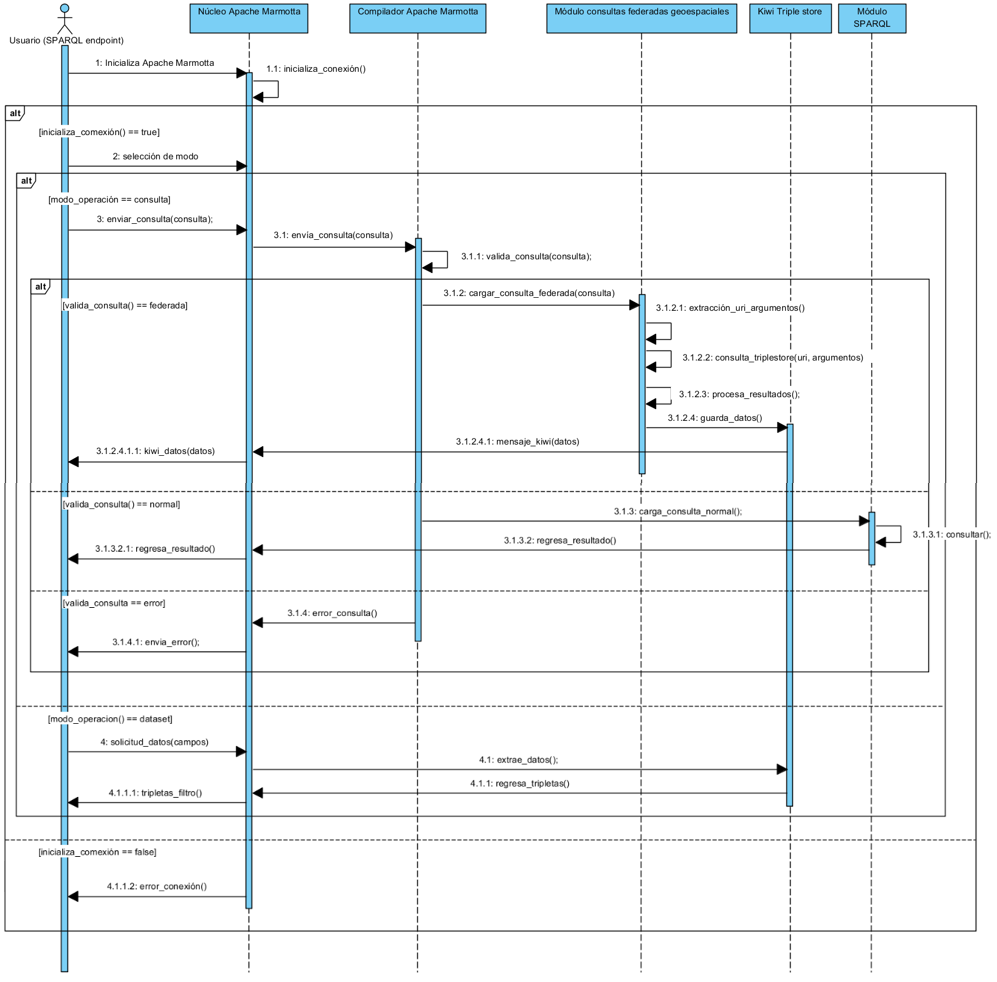
La figura 26 muestra el diagrama de secuencia que el módulo de consultas federadas posee para llevar a cabo una consulta. Auxiliándose de su respectivo diagrama de estados, este diagrama detalla cómo es que se lleva a cabo el proceso.

Figura 25 Diagrama de secuencia para módulo de consultas federadas geoespaciales

Con el objetivo de mostrar a detalle la figura 27, se muestran a continuación tres figuras que particularizan el diagrama de secuencias por partes.

La figura 23 muestra el diagrama de secuencia para establecer conexión con Apache Marmotta.



Figura 26 Diagrama secuencia inicio sesión.

La figura 28 muestra el diagrama de secuencia correspondiente a la acción de selección de modo en proveniente de la aplicación Web.

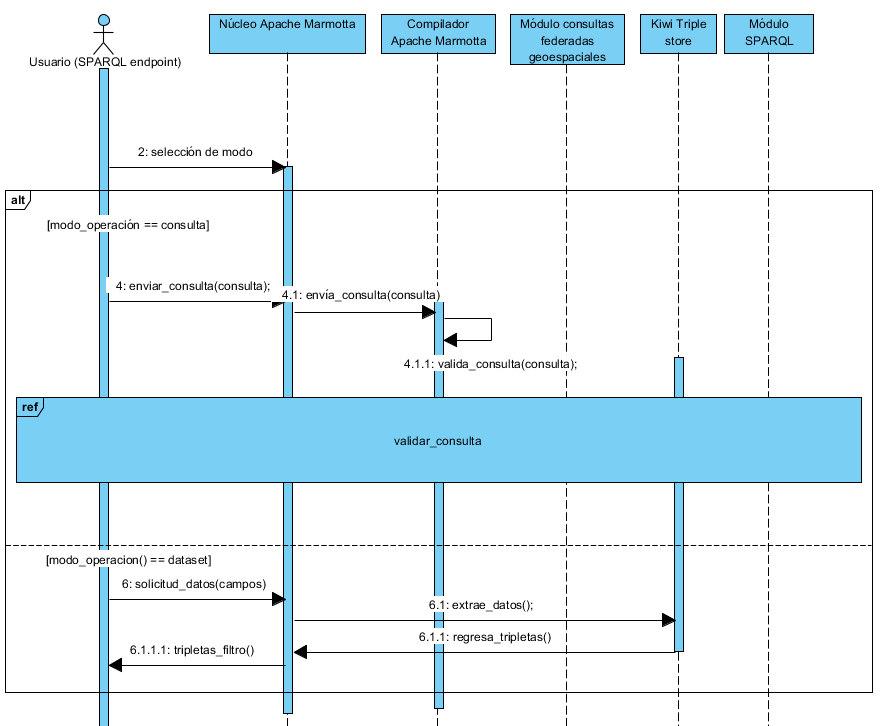


Figura 27 Diagrama de secuencia - selección de modo.

La figura 29 muestra el diagrama de secuencia correspondiente al flujo de la aplicación después de que Apache Marmotta reciba y valide la consulta ingresada por el usuario.

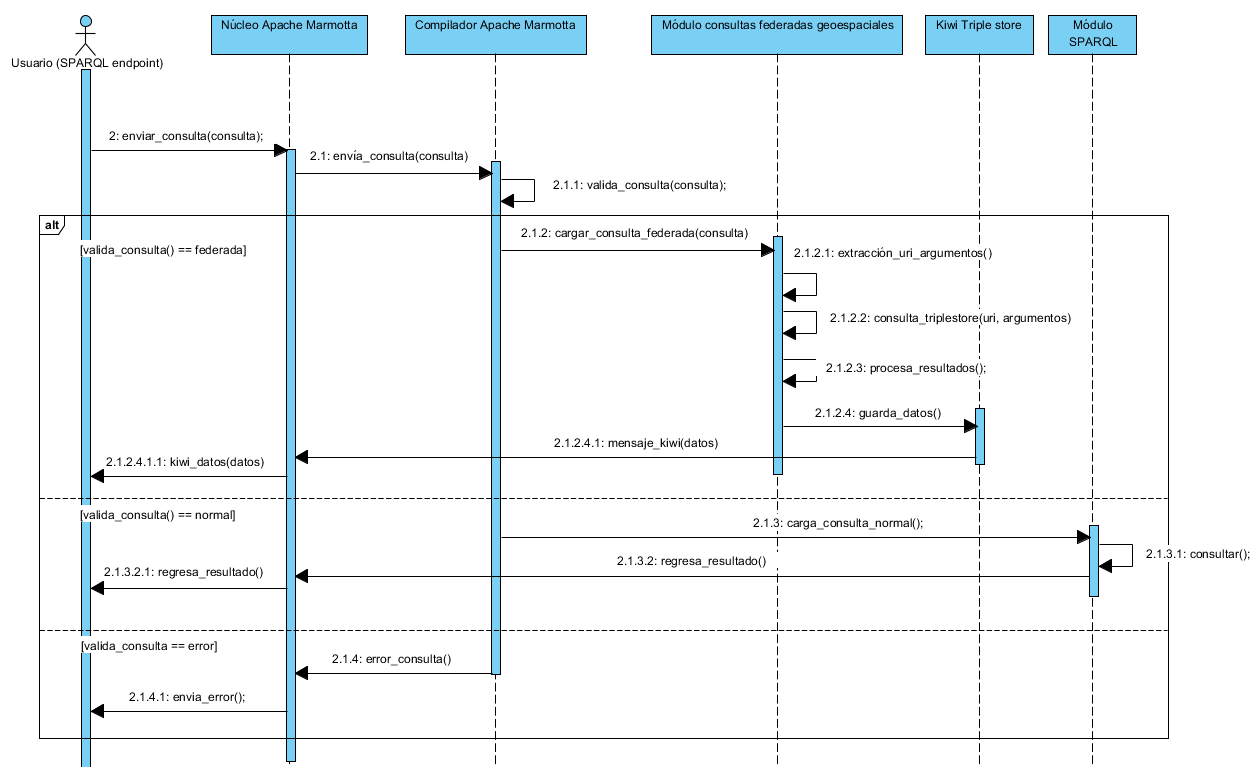
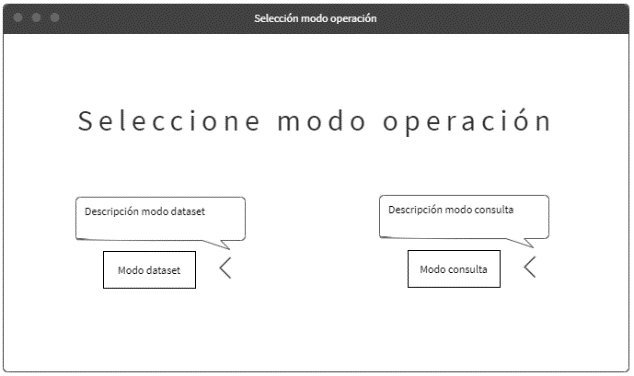
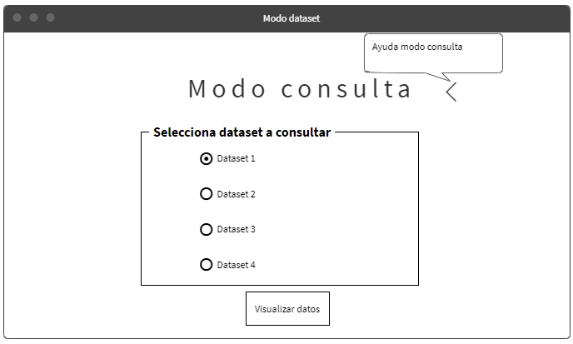
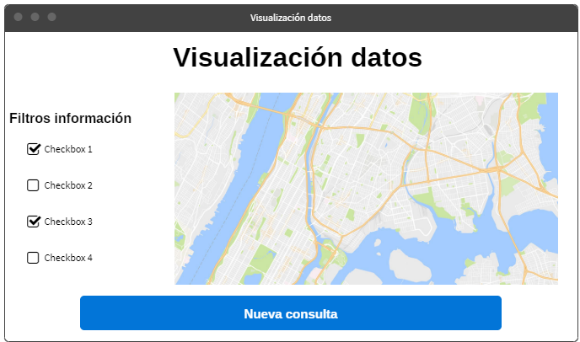
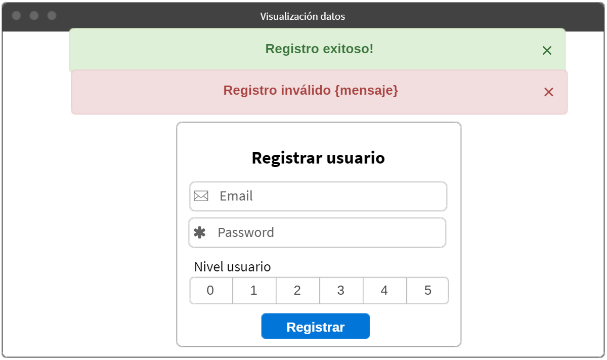
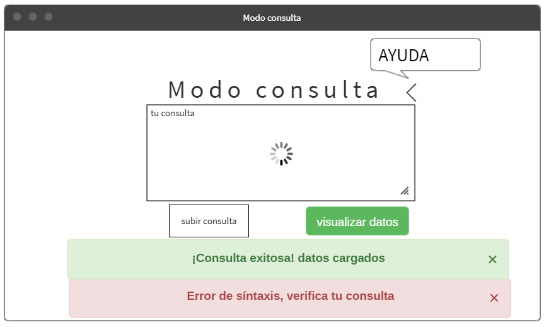


Figura 28 Diagrama de secuencia - validación consulta.

## Secuencia de interfaces

A continuación, se muestra la secuencia de interfaces, *mockups*, que tiene la aplicación Web tanto para el usuario común como para el administrador. Ambos usuarios comparten una pantalla y es la de inicio de sesión, pero para el usuario administrador solo hay una pantalla y es la de registro de usuarios mientras que para el usuario común hay cuatro páginas por la que es usuario puede navegar.





Usuario: Administrador

Usuario: Común

# Implementación

En la presente sección se detalla cómo es que se implementó y desarrolló cada tecnología tanto en la implementación del módulo de consultas federadas en Apache Marmotta y la metodología a seguir para hacer una consulta geoespacial usando datos extraídos de *SPARQL endpoints* remotos, así como el desarrollo de la aplicación Web y el uso de esta.

Se presentan fragmentos de código fundamentales para ambos casos por lo que, si se requiere ahondar a detalle el código y su comportamiento, puede irse a la sección de anexos y ahí podrá encontrar el código completo y los diagramas de flujo de cada método empleado.

## Preparación del proyecto

El producto final del presente proyecto terminal constó de ***N fases***:

* Clonar repositorio de AM cuya rama es la que posee la característica GeoSPARQL
* Compilar el código usando Maven y todas las demás dependencias (PostgreSQL y extensión PostGis)
* Metodología e implementación de código de OpenRDF en Marmotta
* Metodología para llevar a cabo una consulta federada geoespacial
* Desarrollar aplicación Web para visualizar los datos en 3 formatos JSON, Tabla y en un Mapa usando AM como *SPARQL endpoint.*
* Desarrollar pruebas a los *triple store* similares.

A continuación, se describe cada una de las fases

Para clonar e instalar el código de Apache Marmotta en Ubuntu se ejecutan los siguientes comandos

Mkdir ProyectoTerminal2

Cd ProyectoTerminal2

git clone <https://github.com/apache/marmotta.git>

cd marmotta/

git checkout MARMOTTA-584

Después, se debe de instalar Maven ya que será el encargado de construir el código de Apache Marmotta. Se exporta la ruta de Maven como una variable de entorno y luego se procede a compilar el código.

Sudo install Maven

export PATH=/usr/share/maven/bin:$PATH

mvn -DskipTests=true clean install

Una vez ejecutados los comando anteriores, se proceden a instalar las demás dependencias. Se empieza por instalar PostgreSQL y luego se instala la extensión PostGIS. El usuario y contraseña usados en este proyecto son:

* Usuario: marmotta
* Password: ProyectoTerminal2

Si se requiere saber más sobre los detalles de instalación de PostgreSQL y de PostGIS, diríjase al anexo 1 del presente documento.

Con relación a los pasos mencionados anteriormente, lo que sigue es definir la metodología e implementación del código extraído de OpenRDF en Apache Marmotta. Las clases e interfaz implementadas en Apache Marmotta son:

* Interfaz FederatedService.
* Clase SPARQLFederatedService.
* Clase FederatedServiceManager.
* Clase ServiceFallbackIteration.

Tal como se explicó en la sección de diseño, estas clases e interfaz se implementaron basándose en la estructura de un compilador, específicamente en la sección de análisis de un compilador. Si requiere mayor detalle, se recomienda leer la sección de diseño del presente documento.

Una vez implementadas las clases, se procedió a construir nuevamente el proyecto por completo con el mismo comando usado previamente para compilar a Apache Marmotta:

mvn -DskipTests=true clean install

[***IMAGEN DE SISTEMA COMPILADO***]

Una vez implementada la característica se procedió a desarrollar la metodología para realizar una consulta federada a repositorios con datos geográficos.

Una de las limitantes que se tuvieron para obtener datos geográficos es que en varios SPARQL endpoints [***REFERENCIAR LA LSITA DE SPARQL ENDPOINTS CON CARACTERISTICAS***] cuentan con información geográfica en el formato WGS854 el cual está basado en longitud y latitud. Esto se considera como limitante ya que el estándar y lenguaje *GeoSPARQL* requiere datos en tipo Texto Bien Conocido, WKT por sus siglas en inglés, así que buscar *SPARQL endpoints* que cumplieran con esta característica fue otro reto a vencer.

Existe una lista publicada por Wikidata [REFERENCIA] que muestra los *SPARQL endpoints* disponibles y que están en funcionamiento actualizado a la fecha de 20 de abril del 2020. A partir se empezó una nueva búsqueda para encontrar los *SPARQL endpoints* con información geoespacial del tipo WKT.

Uno de los SPARQL endpoints con los que se trabajaron fue LinkedGeoData. Usando este endpoint, se procedió a desarrollar la metodología para llevar a cabo una consulta federada geoespacial y es la siguiente.

1. Consultar los SPARQL endpoints y determinar cuáles son las tripletas que son necesarias.

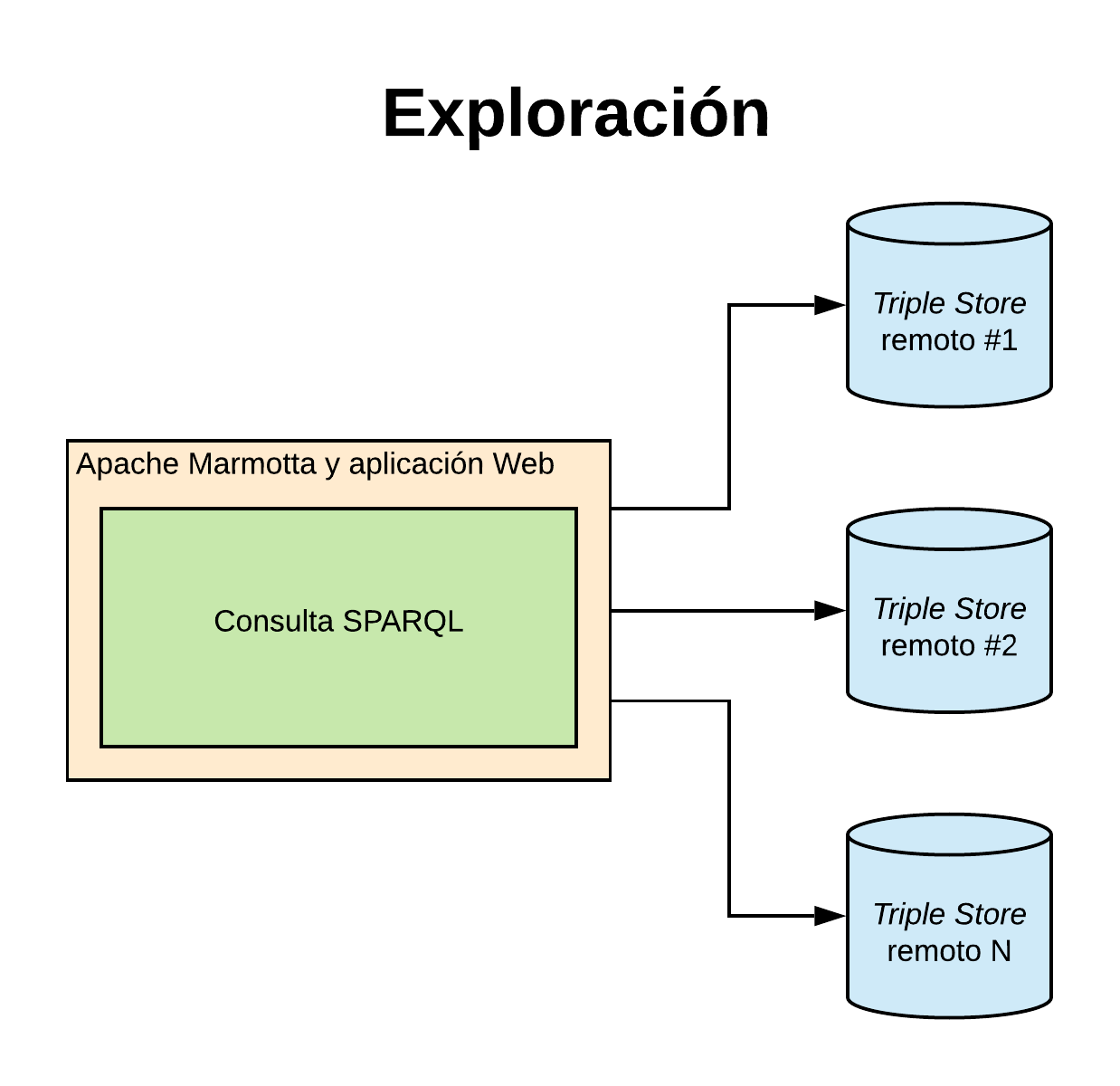


Figura 29 Exploración de Triple Stores

1. Guardar los datos remotos en nuevos grafos. La manera de guardar información de una SPARQL endpoint remoto es mediante la siguientes instrucciones:

Insert data { graph <NombreGrafo> {?subj ?pred ?obj} where { service <URI> {consulta\_federada}}}

1. Repetir el paso 1 y 2 ‘N’ veces por cada *triple store* remoto. Cabe decir que el nombre de cada grafo asociado a cada SPARQL endpoint remoto debe de ser distinto ya que con el nombre se identificará a cada endpoint.

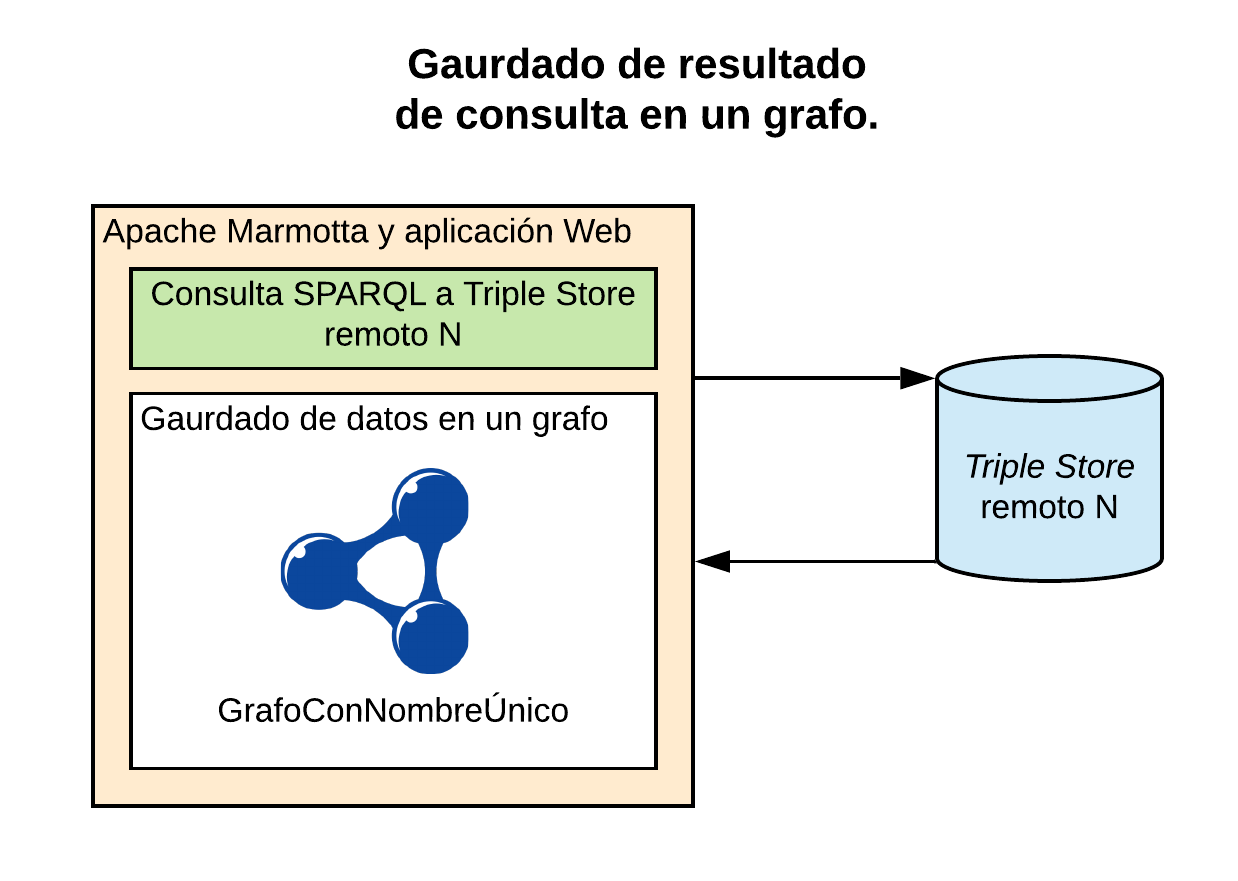


Figura 30 Se guardan los resultados en un grafo nuevo por cada Triple Store remoto distinto.

1. Una vez concluida la recopilación de datos, se procede a hacer las consultas necesarias a cada grafo para que luego los datos de cada grafo sean procesados mediante operaciones geoespaciales disponibles en Apache Marmotta. [***REFERENCIA AL ANEXO DE LAS OPERACIONES DISPONIBLES***].
2. Esperar a que los resultados de la consulta sean retornados.

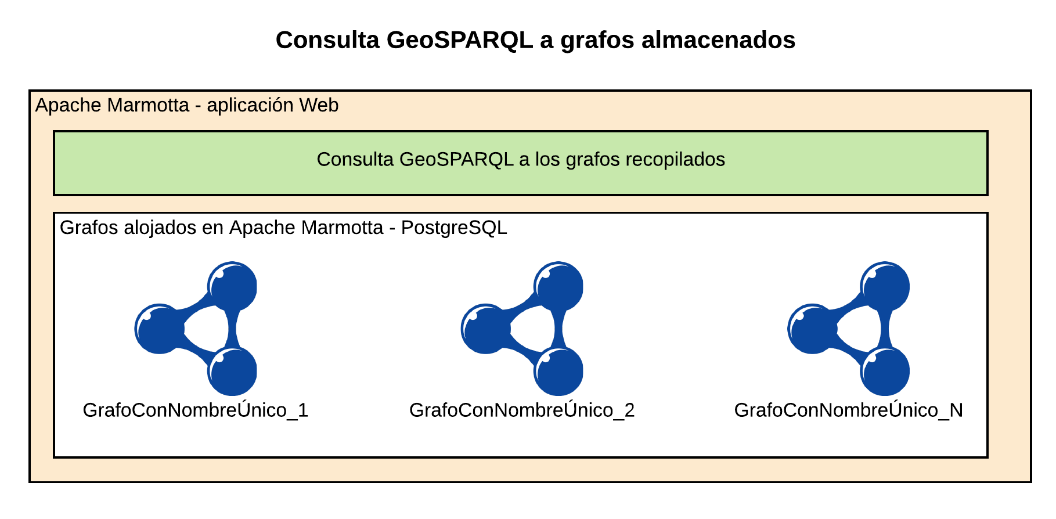


Figura 31 La consulta GeoSPARQL se lleva a cabo de manera Interna

1. En caso de que la consulta geoespacial sea llevada a cabo en la aplicación Web, los datos podrán ser visualizados en formato JSON, tabla, JSON y tabla o en un mapa el cual está basado sobre la herramienta LeafLet [***REFERENCIA A LEAFLET***].

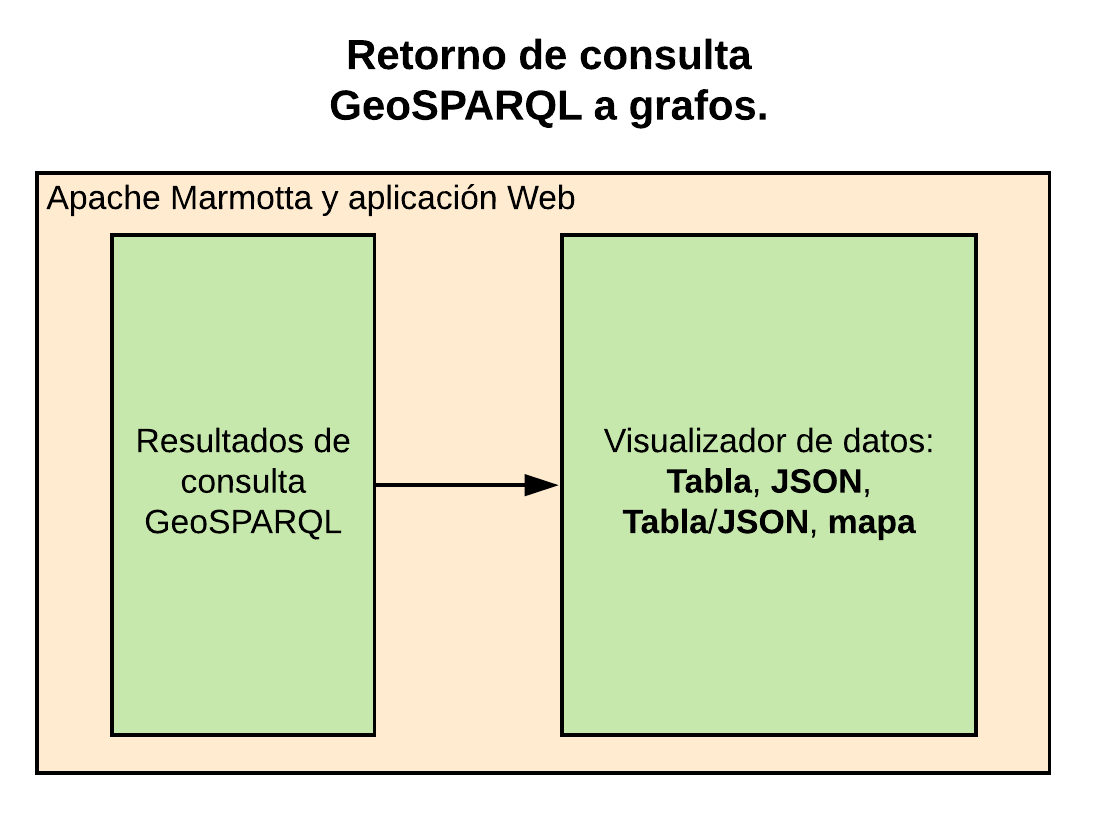


Figura 32 Los resultados de la consulta GeoSPARQL pueden ser visualizados en 4 opciones.

La forma que se puede llevar a cabo consultas federadas y el procesamiento de datos geográficos con *GeoSPARQL* consta de una serie de pasos y son:

## Implementación de código de Apache Marmotta

# Referencias

|  |  |
| --- | --- |
| [1] | T. Berners Lee, J. Hendler y O. Lassila, «The Semantic Web,» *Scientific American,* vol. 284, nº 5, pp. 34-43, 2001. |
| [2] | C. Bizer, T. Heath y T. Berners-Lee, «Linked Data - The Story So Far,» de *Semantic services, interoperability and web applications: emerging concepts*, USA, Information Science Reference, 2009, pp. 205-227. |
| [3] | W3C, «SPARQL 1.1 Overview,» 21 Marzo 2013. [En línea]. Available: https://www.w3.org/TR/sparql11-overview/. [Último acceso: 18 Marzo 2019]. |
| [4] | OGC, «GeoSPARQL - A Geographic Query Language for RDF Data,» 10 Septiembre 2012. [En línea]. Available: https://portal.opengeospatial.org/files/?artifact\_id=47664. [Último acceso: 24 Abril 2019]. |
| [5] | R. Battle y D. Kolas, «Linking Geospatial Data With GeoSPARQL,» de *Semant Web J Interoperability, Usability, Appl. Accessed*, vol. 24, Arlington, 2011. |
| [6] | L. Lupercio, F. Baculima, M. Espinoza y V. Saquicela, «Explotación de información en el dominio geo-hídrico ecuatoriano utilizando tecnología semántica,» *Maskana,* vol. 6, pp. 69-77, 2015. |
| [7] | L. M. Vilches Blázquez y J. Saavedra, «A framework for connecting two interoperability universes: OGC Web Feature Services and Linked Data,» *Transactions in GIS,* vol. 23, nº 1, pp. 22-47, 2018. |
| [8] | R. Klischewski, «Semantic Web for e-Government,» de *International Conference on Electronic Government*, Heidelberg, 2003. |
| [9] | Apache, «Apache Marmotta,» 15 Febrero 2019. [En línea]. Available: http://marmotta.apache.org/. [Último acceso: 20 Febrero 2019]. |
| [10] | O. E. ingGroup, «MAP4RDF,» 2015. [En línea]. Available: http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/downloads/172-map4rdf/index.html. [Último acceso: 24 Abril 2019]. |
| [11] | W. Beek, W. Folmer, L. Rietveld y J. Walker, «Geoyasgui: The GeoSPARQL query editor and result visualizer,» *The internationa Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. 42, p. 39, 2017. |
| [12] | A. Marmotta, «Apache Marmotta Platform: SPARQL,» 30 Abril 2014. [En línea]. Available: http://marmotta.apache.org/platform/sparql-module.html. [Último acceso: 24 Abril 2019]. |
| [13] | W3C, «Category: Triple Store,» 5 Noviembre 2013. [En línea]. Available: https://www.w3.org/2001/sw/wiki/Category:Triple\_Store. [Último acceso: 20 Febrero 2019]. |
| [14] | O. G. Consortium, «GeoSPARQL - A Geographic Query Language for RDF Data,» 7 Julio 2001. [En línea]. Available: https://www.opengeospatial.org/standards/geosparql. [Último acceso: 18 Marzo 2019]. |
| [15] | M. Shmidt, O. Görlitz, P. Haase, G. Ladwig, A. Shwarte y T. Tran, «FedBench: A Benchmark Suite for Federated Semantic Data Query Processing,» de *International Semantic Web Conference*, Berlín, 2011. |
| [16] | M. Schmidt, T. Hornung, G. Lausen y C. Pinkel, «SP^2 Bench: a SPARQL performance benchmark,» de *2009 IEEE 25th International Conference on Data Engineering*, Freiburg, Alemania, 2009. |
| [17] | C. Bizer y A. Schultz, «The berlin sparql benchmark,» *International Journal on Semantic Web and Information Systems,* vol. 5, nº 2, pp. 1-24, 2009. |
| [18] | A. Fuggetta, «Open source software––an evaluation,» *Journal of Systems and Software,* vol. 66, nº 1, pp. 77-90, 2003. |
| [19] | N. Wiegand, R. Grove, J. Wilson y D. Kolas, «Querying Geospatial Data over the Web: a GeoSPARQL Interface,» de *Proceedings of Workshop on Managing and Mining Enriched Geo-Spatial Data*, Virginia, 2014. |
| [20] | R. Battle y D. Kolas, «Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL,» *Semantic Web,* vol. 3, nº 4, pp. 355-370, 2012. |
| [21] | C. Buil-Aranda, A. Polleres y J. Umbrich, «Strategies for Executing Federated Queries in SPARQL1.1,» de *International Semantic Web Conference*, Chile, 2014. |
| [22] | J. Sheridan y J. Tenninson, «Linking UK Government Data,» de *Ldow*, Reino Unido, 2010. |
| [23] | T. Zhao, C. Zhang, L. Anselin, W. Li y K. Chen, «A parallel approach for improving Geo-SPARQL query performance,» *International Journal of Digital Earth,* vol. 8, nº 5, pp. 383 - 402, 2015. |
| [24] | M. Morsey, J. Lehmann, S. Auer y A. C. Ngonga Ngomo, «DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data,» de *International semantic web conference*, Berlin, Heidelberg, 2011. |
| [25] | N. Charlampos, D. Kallirroi, K. Kostis y K. Manolis, «Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data,» de *Extended Semantic Web Conference*, Grecia, 2013. |
| [26] | K. Bereta, G. Xiao y M. Koubarakis, «ANSWERING GEOSPARQL QUERIES OVER RELATIONAL DATA,» *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences,* vol. 42, 2017. |
| [27] | G. A. Atemezing y F. Amardeilh , «Benchmarking Commercial RDF stores with Publications Office Dataset,» de *European Semantic Web Conference*, 2018. |
| [28] | G. Garbis, K. Kyzirakos y M. Koubarakis, «Geographica: A Benchmark for Geospatial RDF Stores,» de *International Semantic Web Conference*, Berlin, 2013. |
| [29] | E. Ávila Barrientos, «Linked Open Data en la Biblioteca Digital,» de *Biblioteca Digital Académica en Bibliotecología y Estudios de la Información*, México, 2013, pp. 137-152. |
| [30] | A. Becerril García, R. Lozano Espinosa y J. M. Molina Espinosa, «Enfoque semántico para el descubrimiento de recursos sensible al contexto sobre contenidos académicos estructurados con OAI-PMH,» *Computación y sistemas,* vol. 20, nº 1, pp. 127-142, 2016. |
| [31] | R. Zárate Escobedo, Facilitador de contenido móvil para el viajero basado en servicios de localización y web semántica, México, 2018. |
| [32] | D. F. Rojas Carrasco y R. Torres Covarrubias, Recuperación de información geográfica utilizando similitud semántica, México, 2009. |
| [33] | W3C, «Semantic Web,» 9 Marzo 2019. [En línea]. Available: https://www.w3.org/standards/semanticweb/. [Último acceso: 18 Marzo 2019]. |
| [34] | W3C, «LinkedData,» 1 Agosto 2016. [En línea]. Available: https://www.w3.org/wiki/LinkedData. [Último acceso: 18 Marzo 2019]. |
| [35] | W3C, «RDF,» 15 Marzo 2014. [En línea]. Available: https://www.w3.org/RDF/. [Último acceso: 18 Marzo 2019]. |
| [36] | W3C, «URI,» 1 Febrero 2005. [En línea]. Available: https://www.w3.org/wiki/URI. [Último acceso: 18 Marzo 2019]. |
| [37] | Ontotext, «What is RDF Triple Store?,» 21 Noviembre 2016. [En línea]. Available: https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/. [Último acceso: 20 Febrero 2019]. |
| [38] | DBpedia, «Virtuoso SPARQL Query Editor,» 17 Abril 2018. [En línea]. Available: https://dbpedia.org/sparql?help=intro. [Último acceso: 20 Febrero 2019]. |
| [39] | DBpedia, «Learn about DBpedia,» DBpedia, 24 Octubre 2019. [En línea]. Available: https://wiki.dbpedia.org/about. [Último acceso: 10 Noviembre 2019]. |
| [40] | D. K. Becker, Information Quality & Service Oriented Architecture, Massachusetts: MIT, 2007. |
| [41] | T. Berners Lee, R. Fielding, J. Gettys, J. Mogul, H. Frystyk y L. Masinter, Hypertext Transfer Protocol -- HTTP/1.1, W3C/MIT, 1999. |
| [42] | M. Develpers, «MDN web docs,» 15 07 2019. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Methods. [Último acceso: 29 09 2019]. |
| [43] | Codecademy, «Codecademy,» 12 08 2019. [En línea]. Available: https://www.codecademy.com/articles/what-is-rest. [Último acceso: 29 09 2019]. |
| [44] | I. E. T. F. (IETF), The JavaScript Object Notation (JSON) Data Interchange Format, T. Bray, 2017. |
| [45] | J. H. Alejandro, Apuntes de compiladores, Ciudad de México: UNAM, 1980. |
| [46] | IEEE, Especificación de Requisitos de Software IEEE 830-1998, Standards IEEE, 1998. |

# Anexos

## Anexo 1: Instalación PostgreSQL para el backend de Apache Marmotta

Se actualizan los repositorios y luego se intala PostgreSQL.

sudo apt update

sudo apt install postgresql postgresql-contrib

Se crea un nuevo rol y se ingresa la información solicitada

sudo -i -u postgres

create user –interactive

Enter name of rol to add: marmotta

Shall the new role be a superuser? Y

Se crea una nueva base de datos con el mismo nombre

createdb marmotta

Puesto que lo anterior solo respecta a la creación de la bd y del usuario dentro del entorno de postgresql, falta agregarlo al usuario principal para que éste se pueda conectar a postgres

sudo adduser marmotta

password: ProyectoTerminal2

Se ingresa a PostgreSQL

psql

\* se le asigna una contraseña al usuario marmotta\_user

alter user marmotta\_user with PASSWORD ‘marmotta\_user’;

Se anexa la extensión PostGIS a PostgreSQL

CREATE EXTENSION postgis;

CREATE EXTENSION postgis\_topology;

Se procede a salir de la consola de PostgreSQL y luego se edita el archivo de configuración de PostgreSQL para que acepten las conexiones pertinentes.

Primero con postgresql.conf

listen\_addresses = '\*'

Luego en pg\_hba.conf se agrega la siguiente línea

host all all 0.0.0.0/0 md5

Se reinicia PostgreSQL

service postgresql restart

Se agrega al firewall el permiso de tráfico para el puerto de PostgreSQL [5432]:

sudo ufw allow 5432

\* se reinicia el Firewall de Ubuntu

sudo ufw reset

Una vez concluido el proceso, está lista la base de datos Postgres para ser usada en Apache Marmotta.

## Anexo 2: Instalación MySQL para la aplicación Web

Se instala PostgreSQL 9.X

Sudo install postgres 9.X

Se crea un usuario que cree la base de datos

crate user ‘user\_pt2’@’localhost’

rate user ‘user\_pt2’@’%’

Luego, se le asigna una contraseña y se le otorgan los privilegios necesarios.

set PASSWORD for 'user\_pt2'@'localhost' = PASSWORD('ProyectoTerminal2');

set PASSWORD for 'user\_pt2'@'%' = PASSWORD('ProyectoTerminal2');

grant all privileges on \*.\* to 'user\_pt2'@'%';show grants for user\_pt2@'localhost';

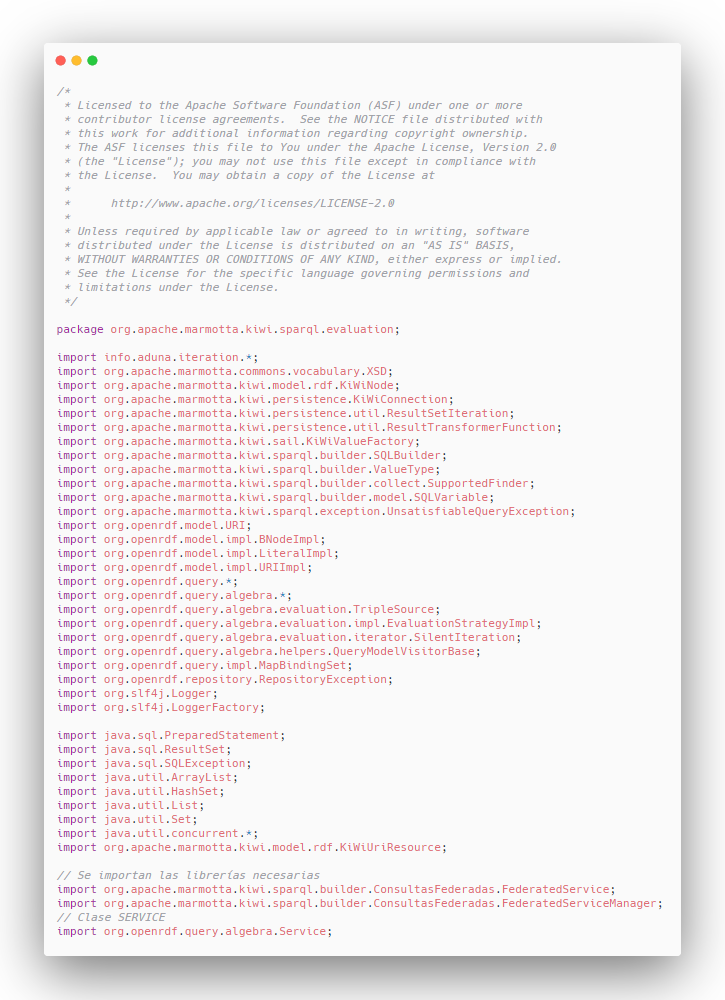
grant all privileges on \*.\* to 'user\_pt2'@'%';show grants for user\_pt2@'%';from

Se crea una base de datos

Create database pt2

## Anexo 3: Código de la clase KiWiEvaluationStrategy.java

Si bien hay más método cuyo nombre es *evaluate,* se documenta el código empleado en el proyecto.







## Anexo 4: Diagrama de flujo para los métodos implementados en KiWiEvaluationStrategy.

En este anexo se muestra el diagrama de flujo del método *evaluate*, *getBoundVariables* que es una función auxiliar invocada en *evaluate*.

### Método evaluate.

Imagen que contiene texto

Descripción generada automáticamente

Figura 33 Diagrma de flujo evaluate - Primera parte.

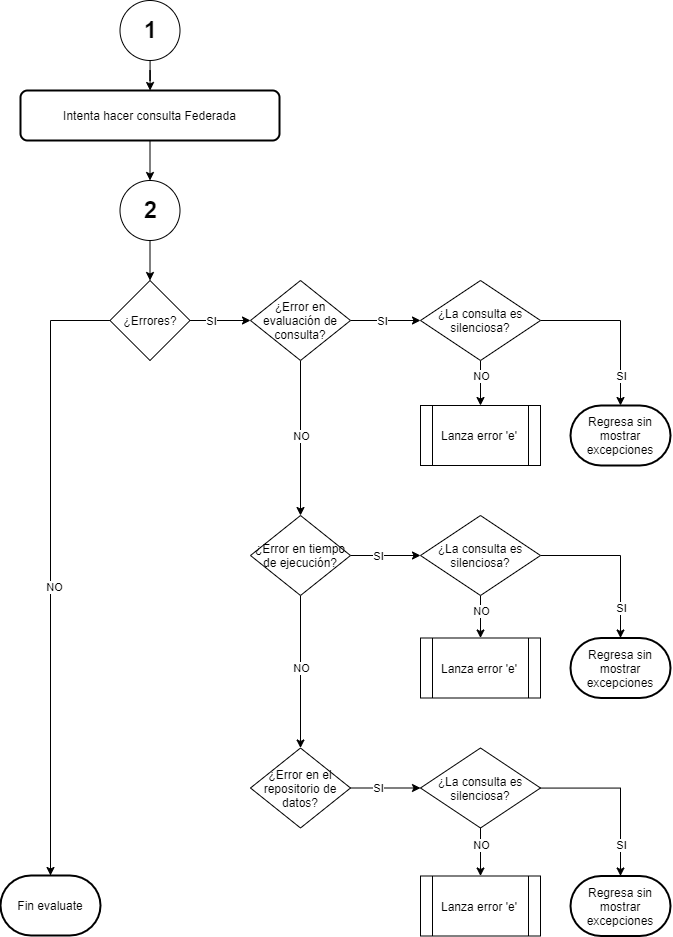


Figura 34 Diagrama de flujo evaluate - Parte 2.

Figura 2

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 35 Diagrama de flujo evaluate desglosado - Parte 1.

Captura de pantalla de un celular con letras

Descripción generada automáticamente

Figura 36 Diagrama de flujo desglosado - parte 2.

### Diagrama de flujo de método getBoundVariables.

Captura de pantalla de un celular con letras

Descripción generada automáticamente

Figura 37 Diagrama de flujo para método getBoundVariables.

1. <http://marmotta.apache.org/platform/sparql-module.html> [↑](#footnote-ref-1)