UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EL ATLAS DE WIKIDATA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

BENJAMÍN OSVALDO DEL PINO BADILLA

PROFESOR GUÍA:
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CARDENAS
PABLO GONZÁLEZ JURE

SANTIAGO DE CHILE
2023

Resumen

En el presente trabajo se muestra el desarrollo de un sistema que permite buscar y geolocalizar instancias de tipos de entidades de Wikidata como por ejemplo: montañas, ríos, museos, obras de arte, centros clandestinos de detención y tortura; entre muchos otros.

En primer lugar, se hizo una revisión de distintos conceptos y herramientas para tener una idea general sobre como funciona el acceso y manejo de los datos del sistema. Entre ellos destacan: Wikidata, SPARQL, y Wikidata Query Service (WDQS). También, se vieron algunas fuentes de datos geográficos como OpenStreetMap y herramientas de visualización como Leaflet que permite la creación de un mapa mundial personalizable. Por último se presentan algunos sistemas geográficos relacionados con Wikidata y con el presente trabajo.

Luego, se presenta el preprocesamiento y análisis de los datos de Wikidata que van a ser utilizados en el sistema. Los resultados generan un JSON con todos los tipos de entidades georreferenciables de Wikidata, vital para la implementación del autocompletado del sistema. También se presenta el acceso a los datos de las instancias de los tipos encontrados utilizando WDQS para obtener sus coordenadas geográficas y más información relevante.

Posteriormente, se muestra la implementación de las distintas partes del sistema y la forma en como estas se comunican entre si generando una arquitectura cliente-servidor. Se verá la implementación de una API Flask que se comunica con los datos obtenidos del análisis y también a los datos que Wikidata provee al utilizar su API. Se explicarán las funcionalidades de autocompletado de búsqueda y los resultados que estas generan. Además, se presentarán las distintas componentes de la interfaz del sistema como la barra de búsqueda y selector de límite, el mapa mundial y el cuadro con resultados de las búsquedas.

Con el sistema funcionando, se realizaron distintos experimentos para medir el rendimiento del mismo. En primer lugar, se realizó una evaluación del preprocesamiento y análisis del *dump* de Wikidata. Luego, se evaluó el rendimiento tanto para el autocompletado como para la obtención de los resultados para los más de 21 mil tipos que se encontraron. Finalmente, se muestran los resultados de una encuesta completada por usuarios que tuvo el sistema para medir la usabilidad. Finalmente, se concluye sobre los objetivos inicialmente planteados y se exponen los aspectos positivos y negativos del sistema. Se cree que el objetivo de desarrollar un sistema con las características propuestas se cumplió de buena manera, sin embargo, este podría mejorar principalmente en la generación del mapa y su contenido; también se piensa en nuevas funcionalidades para que el sistema sea mucho más atractivo para los usuarios y así sacarle el máximo provecho a la obtención de datos de Wikidata.

Agradecimientos

Quiero aprovechar esta parte para agradecer a todas las personas que estuvieron conmigo en estos años de Universidad, que de manera directa o indirecta me han ayudado a estar donde estoy ahora. También agradecer por sobre todo a las personas que han estado conmigo durante toda mi vida, a mi familia.

Agradezco a Claudia y a Gonzalo, mis padres, quienes siempre se esforzaron para darme una vida plena a mi y a mi hermano, para que nunca nos falte nada. Agradezco el amor, los valores y las enseñanzas que me han dado durante mi vida y la incondicionalidad que han tenido conmigo. Por esto estaré eternamente agradecido, los amo.

A mi hermano Sebastián, por el apoyo incondicional que me ha dado durante toda mi vida y por ser un compañero fiel, del cual siempre he sentido confianza y hasta admiración. Han sido largos años de esfuerzo y sacrificio que se ven plasmados en nuestros trabajos de tesis y ya estamos finalizando, casi al mismo tiempo como si no pudiéramos ser más iguales. Te amo hermano mio, mi sangreee.

A mis abuelos. A Tava, Tavita, mi abuela querida, que al igual que mis padres siempre me ha dado su apoyo y amor incondicional. A Edgar, Osvaldo o careloco; como te gustaba decirnos a mi hermano y a mi. Lamentablemente ya no estás aquí conmigo pero, ¡acá estoy abuelo!, finalizando este proceso, estoy seguro que estarías muy orgulloso de mi. Un abrazo a donde sea que estés careloco.

A mis amigos, al grupo más futbolero de todos, a F666. Gracias por los momentos de ocio que siempre son necesarios. Ojalá nunca falten nuestras risas. Los quiero mucho.

Finalmente agradezco a Aidan, mi profesor guía. Gracias por el apoyo durante estos meses en los que desarrollamos este proyecto tan interesante, que desde el primer momento me llamó la atención. Gracias por darme tranquilidad y apoyarme en momentos donde veía que todo se me escapaba de las manos y se me volvía cuesta arriba. He conocido a un gran profesional, a un gran académico y a una gran persona.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

Actualmente vivimos en la llamada era de la información y la revolución digital. Es sabido que con las nuevas tecnologías en la actualidad se dispone de una cantidad excesiva de información que viene de muchas fuentes distintas de una manera constante y que día a día crece más con la ayuda de todos. Esto puede ser muy útil ya que hoy gracias a los avances de la tecnología y el desarrollo de la Internet se puede acceder a cualquier tipo de información de una manera muy rápida y sencilla.

A pesar de los múltiples beneficios que otorga la desmesurada cantidad de información disponible en la web, existen algunos problemas que pueden afectar negativamente la interacción y el acceso de los individuos a los datos que se les presentan. Uno de los problemas más comunes es la dificultad que se tiene para localizar o encontrar la información que es requerida. Esto ha generado una explosión de la información que provoca que cada vez sea más difícil realizar la labor de calificación de datos, además de que presenta un problema al discriminar cual es la información que es realmente importante de la que no. Un último problema es que este exceso de información en el individuo produce que no sea capaz de procesar y comprender la información obtenida, lo cual puede generar diversos problemas sobre la percepción de la realidad que lo rodea.

Es por esto que surge la necesidad de encontrar una fuente de datos confiable, que se acople a las formas en cómo hoy se genera información y que otorgue una solución a los problemas anteriormente expuestos. Es aquí en donde nace la idea de acceder y distribuir la base de datos de Wikidata.

Wikidata [1] es una base de **datos libre**, **colaborativa** y **multilingüe**, que sirve como una base de datos secundaria y que recopila datos estructurados para dar soporte a Wikipedia, Wikimedia Commons, así como a otras wikis del movimiento Wikimedia y a cualquier persona en el mundo. Wikidata contiene actualmente más de 100 millones de ítems, los cuales han sido editados más de 1.700 millones de veces por alrededor de 24.500 usuarios activos [1].

Estos ítems, también llamados entidades, son usados para representar todas las cosas posibles del conocimiento humano, es decir tópicos, conceptos u objetos. Por ejemplo “Uruguay v Brazil 1950”, “filosofía”, “pasión”, “Estadio Monumental”, “Ville Valo ” son todas entida-

[1] `https://www.wikidata.org/wiki/Wikidata:Statistics`

des de Wikidata. En Wikidata cada uno de estos ítems se identifica de forma única con una `Q` seguida de un número que representa su identificador [2].

En esta base de datos, se pueden buscar entidades por *keyword*, se pueden hacer consultas para encontrar entidades que satisfagan algunas condiciones, etc., pero es difícil navegar Wikidata geográficamente, por ejemplo, para ver las entidades locales en una ciudad. Es por esto que abordar este problema resulta muy interesante ya que permitiría explorar los dataset de una forma geográfica, lo cual ayudaría a entender, por ejemplo, distribuciones geográficas de distintos tipos de entidades como centrales nucleares, puertos navieros, estadios de fútbol, entre otros; puntos cercanos a la ubicación actual del usuario que son instancias de un tipo particular; y sesgos regionales, en términos de que tan completo es Wikidata en distintas partes del mundo.

1.1. Objetivos

1.1.1. Objetivo General

El objetivo de este trabajo es desarrollar una interfaz que permita navegar la base de datos de Wikidata mediante un mapa interactivo e informativo. En esta interfaz, se podrán realizar búsquedas seleccionando un tipo de entidad o instancia de Wikidata como por ejemplo un museo, templo, estadio, playa, entre otros; utilizando un buscador con auto completado que muestre los resultados con mejor ranking de la base de datos. Para las instancias o resultados de la búsqueda de ese tipo de entidad, se generarán en un mapa del mundo *clusters* de los resultados de la búsqueda georreferenciados (registrados en Wikidata con geocoordenadas). Por ejemplo si se decide hacer una búsqueda de "museo", la aplicación deberá generar en el mapa marcadores y/o *clusters* en donde hayan museos (los *clusters* se generan cuando hay muchas entidades cercanas en una región del mundo. También se podrá hacer zoom para ver las ubicaciones de los resultados individuales que en el ejemplo serían las ubicaciones de los museos con sus respectiva información obtenida de Wikidata.

1.1.2. Objetivos Específicos

A continuación se presentarán algunos de los hitos principales que se quieren lograr para cumplir con el objetivo general de este trabajo.

1. Obtener los datos relacionados a los tipos de entidades de Wikidata.
2. Obtener información relevante y actualizada de las entidades georreferenciables de Wikidata.
3. Geolocalizar los resultados de la búsqueda de las entidades.

[2] `https://www.wikidata.org/wiki/Wikidata:Identifiers`

4. Diseñar e implementar una interfaz de usuario intuitiva que permita realizar búsquedas y muestre de una manera atractiva y llamativa los resultados en un mapa mundial.
5. Obtener buenos resultados en términos de eficiencia cuando los resultados de entidades que se obtengan sean muchos.
6. Hacer validaciones de la interfaz y experiencia al usar la aplicación con usuarios reales. Esto se hará generando distintos casos de uso del sistema para luego recopilar una evaluación y apreciación general del mismo.

En el presente informe se presenta el trabajo para abordar los distintos problemas que se han expuesto hasta ahora y para abordar los distintos objetivos propuestos. En la siguiente lista se puede ver una descripción del contenido de cada capitulo:

- En el Capitulo 2 se presenta el estado del arte asociado al trabajo, en donde se profundizará en distintos conceptos necesarios para entender la solución que se propone y se explicarán algunas de las herramientas que se usaron para conseguir los objetivos.
- En el Capitulo 3 se va a hablar del preprocesamiento, análisis y obtención de datos que va a ser parte fundamental de la implementación del sistema
- En el Capitulo 4, se hablará acerca de la arquitectura de la solución, que abarca el *backend* y *frontend* de la aplicación.
- En el Capitulo 5 se mostrarán distintos experimentos con sus respectivos resultados para realizar la evaluación del sistema, así como también de la experiencia que tienen los usuarios de la aplicación.
- En el Capitulo 6 se encontrarán las conclusiones.

Capítulo 2

Estado del Arte

A continuación se presentarán distintos conceptos y estándares de la web, así como también algunas herramientas que se piensan son muy importantes para entender y dar contexto al trabajo que se quiere realizar.

2.1. La Web Semántica

La Web Semántica se trata de un conjunto de actividades desarrolladas en el núcleo de la *World Wide Web Consortium* (W3C) con tendencia a la creación de tecnologías para publicar datos legibles por aplicaciones informáticas (máquinas). Aporta estructura al contenido significativo de las páginas web, creando un entorno en el que los agentes de software que se desplazan de una página a otra puedan realizar fácilmente tareas sofisticadas para los usuarios. La Web Semántica no es una web separada de la ya conocida, sino más bien es una extensión de la misma, en la que se le da un significado bien definido a la información, lo cual permite que los computadores o máquinas y también las personas trabajen mejor en cooperación [2]. Esta extensión de la web se da a través de estándares que promueven formatos de datos comunes y protocolos de intercambio en la web. Para entender el trabajo realizado resulta necesario conocer cuatro de ellos, los cuales son RDF, RDFS, OWL y SPARQL, que serán explicados a continuación.

2.1.1. RDF

El RDF o *Resourse Description Framework* [1] es un estándar del *World Wide Web Consortium* (W3C) usado para, como su nombre lo indica, describir *resources*, que puede ser cualquier cosa con una identidad que uno podría considerar describir en datos. Pueden ser desde entidades virtuales como paginas web, sitios web, archivos de escritorio; entidades concretas como personas, monumentos, lugares, libros; entidades abstractas como especies de flora y fauna, definiciones matemáticas, acontecimientos históricos, entre otras [3]. Fue di-

[1] `https://www.w3.org/TR/rdf11-primer/`

señado originalmente como un modelo de datos para metadatos. Se usa como un método general para la descripción y el intercambio de datos de grafos y proporciona una variedad de notaciones de sintaxis y formatos de serialización de datos. En la figura 2.1 se puede apreciar un modelo simple de datos RDF con información sobre Irlanda y su capital (y condado) Dublín. Los nodos azules indican recursos identificados con IRIs (*Internationalized Resource Identifiers* [4]), que son identificadores que además permiten enlazar recursos en la Web, mientras que los nodos verdes indican *literales*, que representan valores como *strings*, booleanos, fechas, etc.

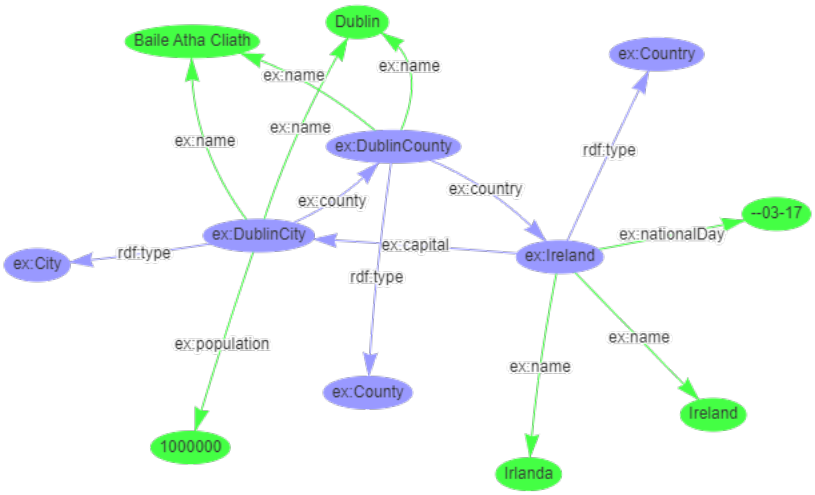

Figura 2.1: Modelo de datos RDF

Para la serialización de RDF, o en otras palabras, para escribir un grafo RDF en archivos para almacenamiento, análisis y procesamiento, es necesaria la existencia de una sintaxis para que una máquina sea capaz de parsear el contenido. Existen variados tipos de sintaxis para serializar grafos RDF, las cuales tienen sus ventajas y desventajas y se adecuan a distintos tipos de configuraciones. Dentro de las más conocidas se encuentran RDF/XML, N-Triples y Turtle. Para este trabajo la sintaxis más relevante es la de N-Triples.

N-Triples

La sintaxis más básica para representar grafos RDF es N-Triples [2]: un formato de texto basado en lineas fácil de analizar que es un subconjunto de Turtle. Su intención original era escribir casos de prueba, pero ha demostrado ser una sintaxis muy popular como formato de intercambio de datos RDF. Un documento en formato N-Triples se limita solo a declaraciones directas y veraces, por lo que no contienen metadatos como calificadores y referencias. Las líneas o triples de un archivo en formato N-Triples son una secuencia de términos RDF que representan el **sujeto**, el **predicado** y el **objeto** (SPO[3]), los cuales pueden estar separados

[2] `https://www.w3.org/TR/n-triples/`
[3] SPO (*Subject, Predicate, Object*) también conocido como triple semántico

por espacios en blanco o tabuladores y terminan con un punto y una nueva línea (opcional al final del documento).

En el Listing 1 se puede ver la representación de declaraciones del grafo RDF de la figura 2.1 en formato N-Triple:

Listing 1 Declaraciones RDF usando sintaxis N-Triple

```
1  <http://ex.org/Ireland>
   ↪ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <http://ex.org/Country> .
2  <http://ex.org/Ireland> <http://ex.org/capital>
   ↪ <http://ex.org/DublinCity> .
3  <http://ex.org/Ireland> <http://ex.org/name> "Ireland"@en .
4  <http://ex.org/Ireland> <http://ex.org/name> "Irlanda"@es .
5  <http://ex.org/Ireland> <http://ex.org/nationalDay>
   ↪ "--03-17"^^<http://www.w3.org/2001/XMLSchema#gMonthDay> .
6
7  <http://ex.org/DublinCity>
   ↪ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <http://ex.org/City> .
8  <http://ex.org/DublinCity> <http://ex.org/name> "Dublin"@en .
9  <http://ex.org/DublinCity> <http://ex.org/name> "Baile Atha
   ↪ Cliath"@ga .
10 <http://ex.org/DublinCity> <http://ex.org/county>
   ↪ <http://ex.org/DublinCounty> .
11 <http://ex.org/DublinCity> <http://ex.org/population> 1000000 .
12
13 <http://ex.org/DublinCounty>
   ↪ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <http://ex.org/County> .
14 <http://ex.org/DublinCounty> <http://ex.org/name> "Dublin"@en .
15 <http://ex.org/DublinCounty> <http://ex.org/name> "Baile Atha
   ↪ Cliath"@ga .
16 <http://ex.org/DublinCounty> <http://ex.org/country>
   ↪ <http://ex.org/Ireland> .
```

2.1.2. RDFS

RDFS o RDF Schema [4] es una extensión **semántica** de RDF, la cual proporciona un vocabulario de modelado de datos para grafos RDF. Además, proporciona mecanismos para describir grupos de recursos relacionados y las relaciones entre estos recursos, los cuales se

[4] https://www.w3.org/TR/rdf-schema/

utilizan para determinar las características de otros recursos, como los dominios y rangos de propiedades.

El sistema de clases y propiedades de RDFS es similar a los sistemas de tipos de los lenguajes de programación orientado a objetos como Java, C++ o Python. Se diferencia de estos sistemas en que, en lugar de definir una clase en términos de las propiedades que pueden tener sus instancias, RDFS describe las propiedades en términos de las clases de recursos a los que se aplican. Una de las ventajas del enfoque centrado en las propiedades de RDF es que permite a cualquiera ampliar la descripción de los recursos existentes, uno de los principios arquitectónicos de la web.

Algunas de las clases de RDFS más básicas e importantes para el entendimiento de este trabajo son:

- **rdfs:Class** que permite declarar recursos como clases para otros recursos
- **rdfs:Resource** la cual es la clase a la que pertenecen todos los recursos
- **rdf:Property** que es la clase que abarca las propiedades

Dentro de RDFS también existen las propiedades, de las cuales a continuación se presentan las más relevantes:

- **rdfs:domain** de una **rdf:Property** declara la clase del sujeto en un triple cuyo predicado es esa propiedad.
- **rdfs:range** de una **rdf:Property** declara la clase o tipo de datos del objeto en un triple cuyo predicado es esa propiedad.
- **rdfs:subClassOf** que es una instancia de **rdf:Property** que permite definir jerarquias. Relaciona una clase con sus subclases.
- **rdfs:subPropertyOf** que es una instancia de **rdf:Property** que permite definir jerarquias de propiedades.

2.1.3. OWL

Al igual que RDFS, OWL o *Web Ontology Language* [5] es un estándar para definir semánticas sobre RDF. Está diseñado para que sea utilizado por aplicaciones que necesiten procesar el contenido de la información en vez de solo presentar información a los humanos. Facilita una mayor capacidad de interpretación del contenido web por parte de la máquina que la soportada por XML, RDF y RDFS al proporcionar un vocabulario adicional junto con una semántica formal. Se puede utilizar para representar explícitamente el significado de los términos en los vocabularios y las relaciones entre esos términos (ontología).

[5] `https://www.w3.org/TR/owl-features/`

2.1.4. SPARQL

SPARQL [6] es un acrónimo recursivo del inglés *SPARQL Protocol and RDF Query Language* y es usado para expresar consultas de grafos RDF. Es una tecnología clave en el desarrollo de la Web Semántica ya que tiene la capacidad para consultar modelos de grafos requeridos y opcionales. También admite agregación, subconsultas, negación, creación de valores por expresiones, prueba de valor extensible y restricción de consultas por grafos RDF de origen. Los resultados de las consultas SPARQL pueden ser conjuntos de resultados u otros grafos RDF.

Ejemplos

A continuación se presentarán algunos ejemplos de consultas en SPARQL sobre el grafo RDF que se presentó anteriormente en la figura 2.1:

Listing 2 Consulta SPARQL para obtener el nombre de la ciudad de Dublín en distintos idiomas

```
PREFIX ex: <http://ex.org/>
SELECT ?name WHERE {
  ex:DublinCity ex:name ?name .
}
# Los resultados de esta consulta son "Dublin"@en y "Baile Atha
↪  Cliath"@ga
```

Listing 3 Consulta SPARQL para obtener la población de la capital de Irlanda

```
PREFIX ex: <http://ex.org/>
SELECT ?population WHERE {
  ex:Ireland ex:capital ?capital .
  ?capital ex:population ?population .
}
# El resultado para esta consulta es 1000000
```

2.1.5. Wikidata

Wikidata [1] es una base de datos libre, colaborativa y multilingüe, que sirve como una base de datos secundaria y que recopila datos estructurados para dar soporte a Wikipedia, Wikimedia Commons, así como a otras wikis del movimiento Wikimedia y a cualquier persona en el mundo.

[6] https://www.w3.org/TR/sparql11-query/

Actualmente Wikidata se basa en el software Wikibase [7], que consiste en un set de extensiones del software MediaWiki [5] para almacenar y administrar datos. MediaWiki por otra parte es el software para wikis más popular, es fácil de usar, tiene poderosos *features*, es altamente configurable y puede escalar a millones de usuarios. Más de 2 mil wikis utilizan MediaWiki como motor para su funcionamiento, incluida la famosa Wikipedia.

Wikidata proporciona distintas formas para acceder a la información sobre las entidades que conforman su base de datos, también permite hacer búsquedas por palabras clave. Por ejemplo si buscamos a la entidad "Alan Turing", algunos de los resultados que Wikidata acerca de este ítem nos entrega son una descripción, su nacionalidad, causa de muerte, ocupación, influencias, número de Erdös, entre otras muchas propiedades (ver figura 2.2). En particular en este trabajo van a ser de mucha importancia las propiedades *coordinate location* (P625) e *instance of* (P31), para la georreferenciaciación de tipos de entidades.

Acceso a los datos de Wikidata

Actualmente y gracias al trabajo que realiza día a día la gente y las máquinas, Wikidata contiene más de 100 millones de artículos y más de 650.000 lexemas [8]. Para acceder a estos datos existen múltiples métodos disponibles, los cuales se presentaran a continuación:

- Mediante búsquedas tradicionales utilizando la interfaz de la página principal de Wikidata que utiliza Elasticsearch [6] para obtener los resultados
- *Linked Data Interface* (URI[9]) para acceder a la información completa de entidades individuales via URI: `http://www.wikidata.org/entity/Q???`[10]
- *Wikidata Query Service* (WDQS) [11] que devuelve los resultados de las consultas realizadas en el lenguaje SPARQL utilizando el *endpoint* `https://query.wikidata.org`
- MediaWiki Action [12], que es la API propia de Wikidata ampliada para incluir algunas acciones específicas de Wikibase, cuyo *endpoint* es `https://wikidata.org/w/api.php`, usada cuando el trabajo que se quiere realizar involucra la edición de entidades de Wikidata, obtención de datos sobre las entidades como historial de revisión y para obtener todos los datos de una entidad en formato JSON.
- Por último existen los *dumps* RDF de Wikidata que contienen los datos completos de todas las entidades de Wikidata los cuales se actualizan semanalmente y son usados cuando el conjunto de resultados esperados es significativamente grande o cuando se desea configurar un servicio de consultas propios. Este formato se representa principalmente mediante dos declaraciones: *truthy* y *full statements*, en donde la primera representa declaraciones que tienen el mejor *ranking* no obsoleto para una propiedad dada y la segunda representa todos los datos sobre las declaraciones en el sistema.

[7] `https://wikiba.se/`

[8] `https://www.wikidata.org/wiki/Wikidata:Data_access`

[9] *Uniform Resource Identifier*, cadena de caracteres que identifica un recurso al que se puede acceder a través de Internet.

[10] `Q???` corresponde al identificador único (UID) utilizado en Wikidata para la representación de entidades

[11] `https://query.wikidata.org/`

[12] `https://www.mediawiki.org/wiki/MediaWiki`

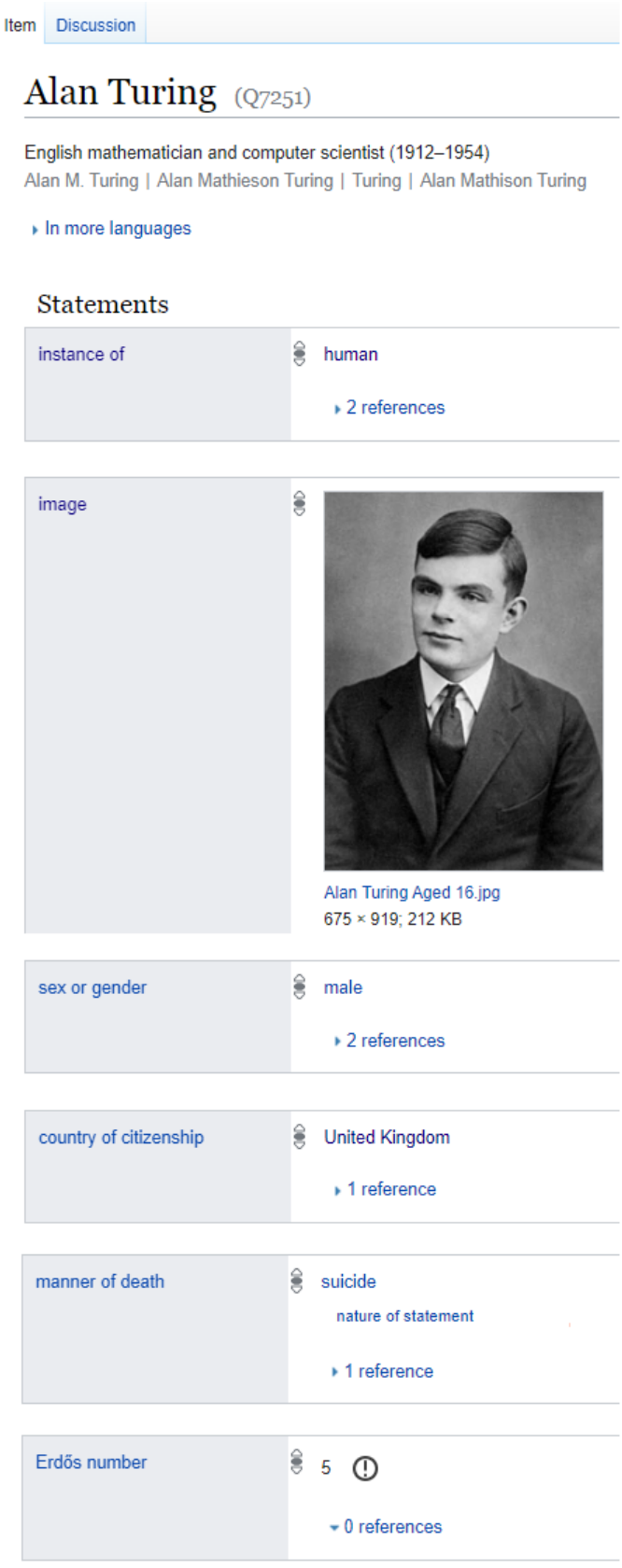

Figura 2.2: Perfil de Wikidata de la entidad Alan Turing con algunas de sus propiedades (Fuente: `https://www.wikidata.org/wiki/Q7251`).

2.2. Sistemas Geográficos

2.2.1. Fuentes de datos geográficos

OpenStreetMap

OpenStreetMap [13], también conocido por sus sigla OSM, es un proyecto colaborativo para crear mapas editables y libres. Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles, ortofotografías y otras fuentes libres. Su base de datos se distribuye bajo la Licencia Abierta de Bases de Datos. Según las estadísticas este proyecto cuenta con más de 7,5 millones de usuarios registrados.

LinkedGeoData

LinkedGeoData [7] se trata de una fuente de datos geográficos que tiene por objetivo agregar una dimensión espacial a la Web de Datos (Web Semántica). Utiliza la información recopilada por el proyecto OpenStreetMap para crear una gran base de conocimientos espaciales y la pone a disposición como una base de conocimiento RDF que consta de más de 3 mil millones de nodos y los cuales comprenden aproximadamente 20 mill millones de triples de acuerdo con los principios de *Linked Data* [14]. Interconecta estos datos con otras bases de conocimiento en la iniciativa Linking Open Data. Los datos están disponibles de acuerdo con los principios de datos vinculados e interrelacionados con DBpedia y Geo Names.

GeoNames

La base de datos geográfica de GeoNames[15] contiene más de 27 millones de nombres geográficos y consta de más de 12 millones de características únicas de las cuales 4,8 millones corresponden a lugares poblados y 15 millones de nombres alternativos. Todas las funciones se clasifican en una de las nueve clases de funciones y se subcategorizan en uno de los 645 códigos de funciones. Se puede acceder a los datos de forma gratuita a través de una serie de servicios web y una exportación diaria de la base de datos. GeoNames ya está atendiendo más de 150 millones de solicitudes de servicios web por día. GeoNames está integrando datos geográficos como nombres de lugares en varios idiomas, elevación, población y otros de varias fuentes. Todas las coordenadas de latitud/longitud están en WGS84 (World Geodetic System 1984)[16]. Los usuarios pueden editar, corregir y agregar nuevos nombres manualmente utilizando una interfaz wiki fácil de usar. GeoNames tiene embajadores en muchos países que ayudan con su ayuda y experiencia.

[13] `https://www.openstreetmap.org/`

[14] `https://www.w3.org/wiki/LinkedData`

[15] `https://www.geonames.org/`

[16] El WGS84 es un sistema geodésico de coordenadas geográficas usado mundialmente, que permite localizar cualquier punto de la Tierra por medio de tres unidades dadas.

2.2.2. Herramientas de visualización geográfica

Hoy en día existe un sin fin de librerías y herramientas para poder abordar el problema de geolocalización. A continuación se presentarán algunas de ellas:

- Leaflet [17]: es una librería escrita en el lenguaje de programación Javascript, que facilita la creación de mapas interactivos de una manera muy simple y con un buen rendimiento. Funciona de manera eficiente en todas las principales plataformas móviles y de escritorio y se puede ampliar con muchos *plugins*. Dentro de sus características más importantes se encuentran que puede ubicar marcadores dentro de los mapas, dibujar formas, agrupar conjuntos y creación de *pop-ups*. En la figura 2.3 se puede ver un ejemplo de visualización con algunas de las funcionalidades comentadas.

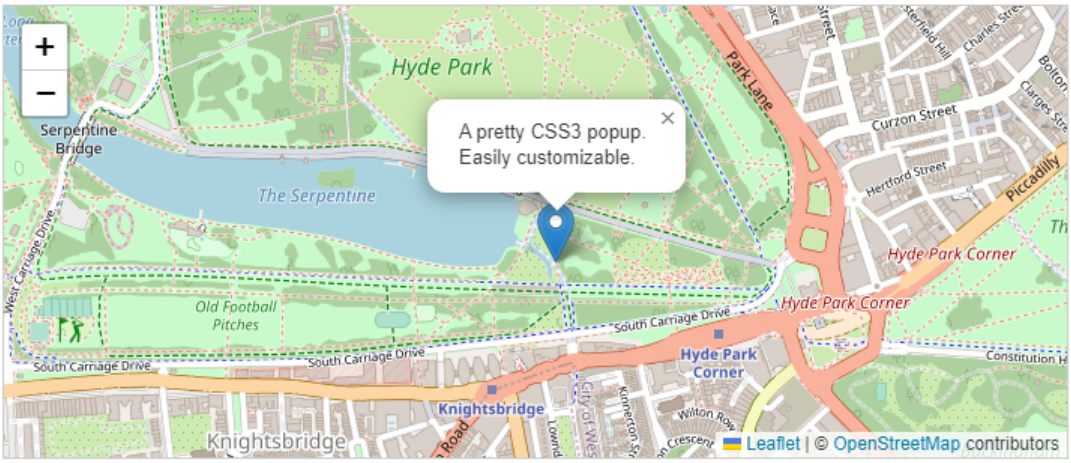

Figura 2.3: Mapa generado con Leaflet

- Folium [18]: es una potente librería de visualización de datos en Python que fue elaborada principalmente para ayudar a la gente a visualizar datos geoespaciales. Folium es un *wrapper* de la librería de Javascript Leaflet.

- D3.js [19]: es una librería de Javascript que permite manipular documentos con bases en datos. Su nombre viene de *Data Driven Document* que tiene que ver con su representación de datos en documentos orientados a objetos. Utiliza HTML, SVG y CSS para hacer sus visualizaciones y aprovecha las ventajas que tienen los buscadores modernos como lo son el poder de visualización y el enfoque basado en datos para la manipulación del DOM.

[17] `https://leafletjs.com/`
[18] `https://python-visualization.github.io/folium/`
[19] `https://d3js.org/`

2.2.3. Sistemas geográficos relacionados con Wikidata

Wikidata:*Map data*

Map data [20] se trata de información geográfica almacenada como puntos, líneas o polígonos en un mapa que describe un área física. Puede almacenar información sobre límites políticos actuales e históricos, límites geológicos, mapas de distribución de especies, etc. y la Tierra, otros planetas e incluso lugares ficticios. Los datos de mapas se pueden utilizar para mostrar información en consultas de Wikidata y mapas de Kartographer [21], que es una extensión de Wikidata que permite crear mapas interactivos y estáticos en las páginas wiki de Wikimedia. Los datos del mapa se almacenan en Wikimedia Commons y se pueden vincular a elementos de Wikidata usando la propiedad *geoshape* (P3896) de datos geográficos de Wikimedia Commons.

Los datos de mapas permiten a los usuarios almacenar datos de GeoJSON[22] en wiki, de forma similar a las imágenes. Otros wikis pueden usar estos datos para dibujar encima de los mapas, junto con otras personalizaciones de mapas. En la figura 2.4 se puede ver un ejemplo de esto que muestra la distribución de la actividad electoral de las elecciones municipales de Finlandia en 2012.

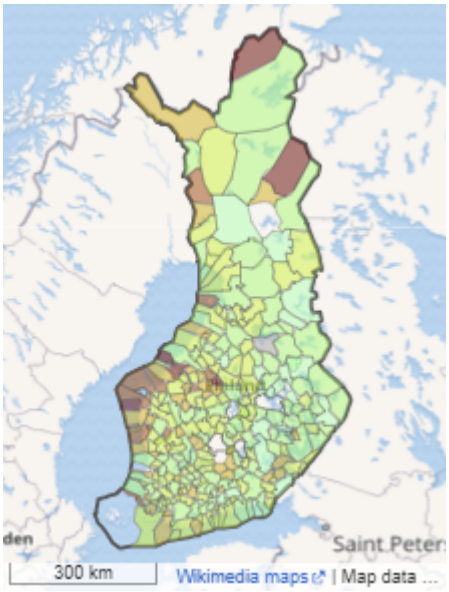

Figura 2.4: Mapa electoral de municipios de Finlandia del año 2012 generado por Kartographer (Fuente: `https://www.wikidata.org/wiki/Template:Election_map_example`)

[20] `https://www.wikidata.org/wiki/Wikidata:Map_data`
[21] `https://www.mediawiki.org/wiki/Help:Extension:Kartographer`
[22] `https://en.wikipedia.org/wiki/GeoJSON`

iD

iD[23], se trata de una herramienta para mapeadores y desarrolladores que permite realizar contribuciones increíbles a mapas sin tener que pasar por una curva de aprendizaje pronunciada. Cuando selecciona una función del mapa, como una escuela, una estatua, un límite o una carretera, iD sugiere los artículos de Wikipedia apropiados para etiquetar la función. Luego, tan pronto como seleccione un artículo, iD también etiqueta la función con el elemento de Wikidata asociado. Una vez que una función de OpenStreetMap se vincula a un elemento de Wikidata, las aplicaciones basadas en OpenStreetMap pueden obtener acceso a una gran cantidad de información. En la figura 2.5 se puede ver la interfaz del sistema.

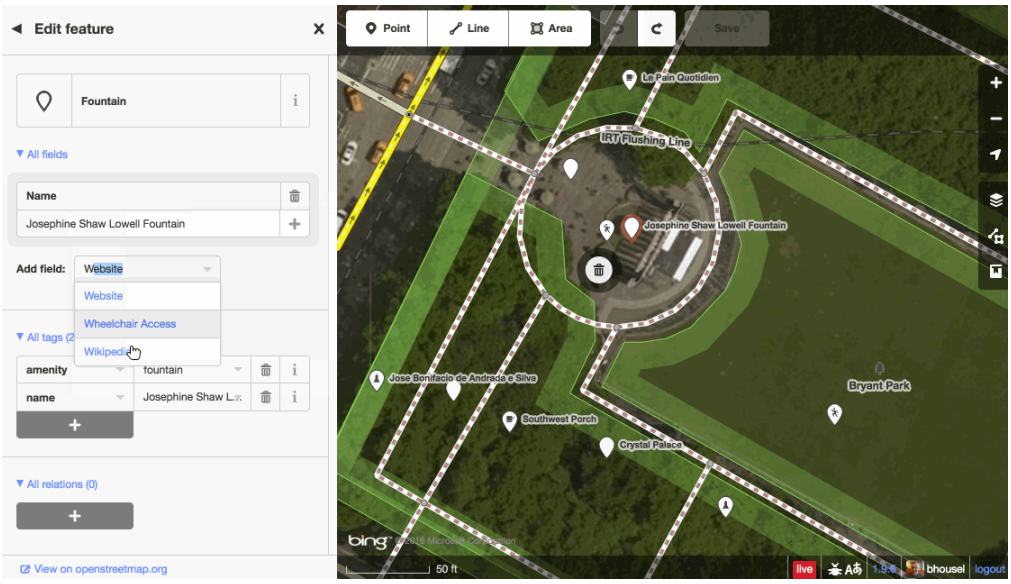

Figura 2.5: Interfaz de iD en donde se está añadiendo un nuevo campo a una fuente (función de mapa).

Sistema para geolocalizar investigación

Se trata de un sistema que permite geolocalizar artículos de ciencias de la computación [8], el cual se pensó desde la ausencia de sistemas en la web que tengan la característica de agrupar publicaciones científicas con respecto al lugar de origen o trabajo de sus autores. Tiene como principales objetivos el fomentar la colaboración entre expertos de un área, mejorar la organización de conferencias y ayudar a investigadores en la elección de lugares para estudiar o trabajar. Utiliza distintas fuentes de datos como DBLP[24], Open Academic Graph[25] y Semantic Scholar[26]. Para geolocalizar la información utiliza Wikidata y herramientas de *entity linking* que es la tarea de asignar una identidad única a entidades mencionadas en un texto.

[23] https://blog.mapbox.com/connecting-openstreetmap-and-wikidata-232f0c412926
[24] https://dblp.org/db/conf/dbpl/index.html
[25] https://www.microsoft.com/en-us/research/project/open-academic-graph/publications/
[26] https://www.semanticscholar.org/about

Wiki Atlas

Wiki Atlas[27] (no confundir con Wikidata Atlas, nombre de este proyecto) es un proyecto que aspira a convertirse en un medio alternativo para explorar la información y el conocimiento disponible en Wikipedia más allá de las interfaces existentes. Permite que una amplia gama de actores descubran, se conecten y compartan conocimientos a través de los límites de la comunidad. Puede ser utilizado como una herramienta con el potencial de mejorar y facilitar la educación y el aprendizaje. Como caso de uso, los educadores pueden crear una colección de artículos sobre un tema de interés (por ejemplo, Historia de la Segunda Guerra Mundial) y compartirlo con los estudiantes en el contexto de clases de historia y geografía. En la figura 2.6 se puede ver su interfaz.

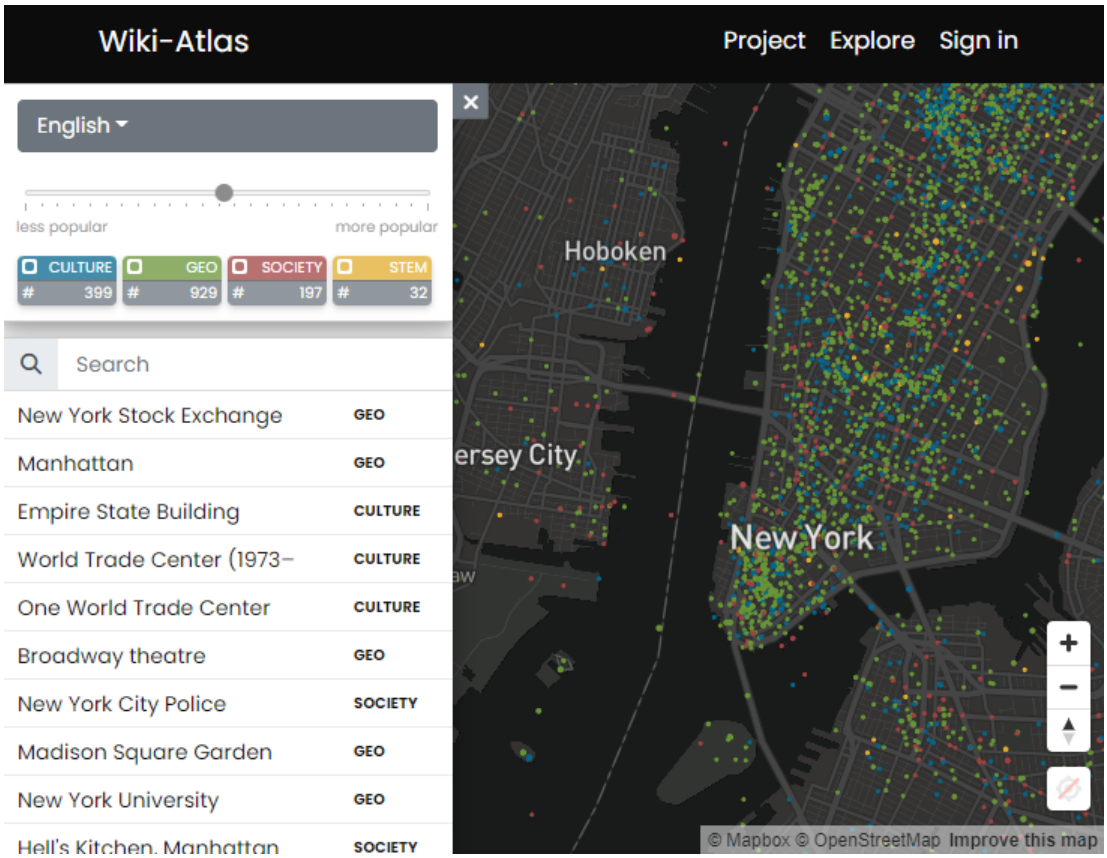

Figura 2.6: Interfaz de Wiki-Atlas

[27] http://www.wiki-atlas.org/

Capítulo 3

Datos

En el presente capitulo se describirá todo el trabajo relacionado al tratamiento de los datos del proyecto los cuales fueron obtenidos principalmente y únicamente a través de Wikidata. Se partirá por explicar de forma muy superficial qué son las propiedades de Wikidata y cuáles de estas son las más relevantes para el contexto del proyecto, a continuación se explicará la forma en cómo son obtenidos los datos desde Wikidata, se nombrarán algunos métodos, sus propósitos y casos de uso. Luego se hablará del manejo que se hizo con estos datos para obtener en primer lugar las entidades de Wikidata correspondientes a las propiedades de entidades georreferenciables como por ejemplo un museo, montaña, universidad, templo, entre otros; y en segundo lugar, de las entidades que son instancias de las propiedades mencionadas anteriormente, por ejemplo en el caso de los museos deberán aparecer entidades como el Museo de Bellas Artes de Santiago, el Museo del Louvre de París o el Rijksmuseum de Amsterdam.

3.1. Propiedades de Wikidata

Hasta la fecha, en Wikidata, existen un total de 10.562 propiedades [1], las cuales describen el valor de datos de una declaración y se pueden considerar como una categoría de datos en donde a cada declaración en una página de ítem se vincula a una propiedad y le asigna un valor. Las propiedades, cuando se combinan con valores, forman una declaración en Wikidata, se utilizan en los clasificadores, tienen sus propias páginas en Wikidata; y están conectadas a elementos o entidades, lo que da como resultado una estructura de datos vinculados, por ejemplo, "continente", "capital" o "expectativa de vida" para el ítem con etiqueta "Chile" son algunas de sus propiedades.

Dentro de las más de 10 mil propiedades existentes en la Wikidata existen dos en particular a las que se le hará enfoque para fines de este trabajo. Una corresponde a la propiedad P31 (*instance of*) que tiene que ver con la clase de la cual una entidad es un ejemplo particular y miembro, en otras palabras al tipo de una entidad, la cual será de gran importancia para la implementación del buscador de la aplicación. Por otra parte, la segunda propiedad de Wikidata que resulta necesaria mencionar es la P625 (*coordinate location*), que indica las

[1] `https://www.wikidata.org/wiki/Wikidata:List_of_properties`

geocoordenadas de una entidad y va a ser importante ya que el principal objetivo del proyecto es justamente la geolocalización de entidades y la visualización de las mismas en un mapa mundial. En la figura 3.1 se puede ver la interfaz de Wikidata con las propiedades antes mencionadas para la entidad Rijksmuseum.

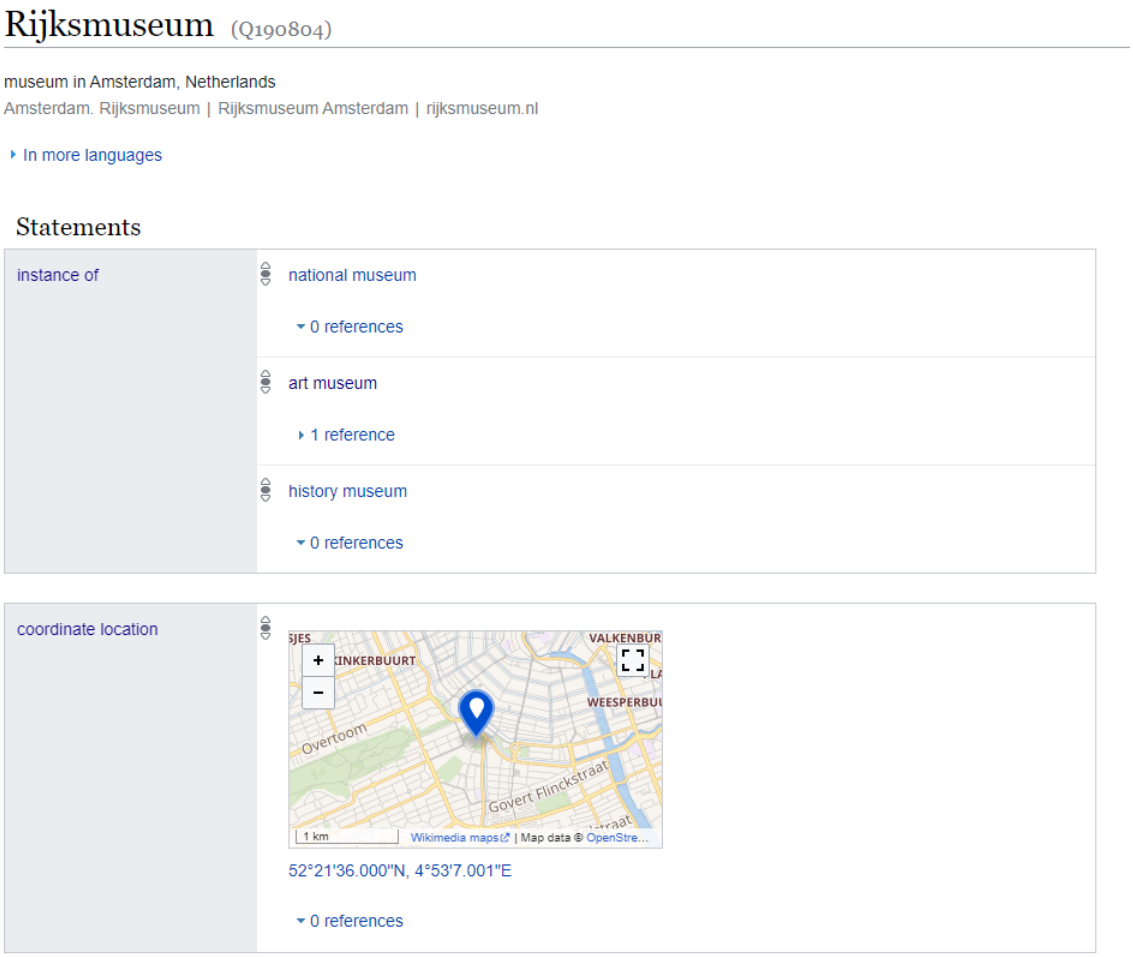

Figura 3.1: Parte del perfil de Wikidata de la entidad Rijksmuseum en donde se aprecian las propiedades *instance of* (P31) y *coordinate location* (P625) (Fuente: `https://www.wikidata.org/wiki/Q190804`)

Como se comentó anteriormente en la sección 2.1.5, existen muchos métodos disponibles para poder acceder a la información de Wikidata, los cuales se adaptan para distintas necesidades y casos de uso. En el caso de este trabajo se va a acceder a la base de datos de Wikidata principalmente mediante los RDF *dumps* y el servicio de consultas WDQS para obtener todas las entidades y la información relevante que cada parte del sistema requiere. El primer método se va a utilizar para hacer un preprocesamiento y posterior análisis de datos del *dump* RDF de declaraciones de tipo *truthy* en formato NTriple (nt) para la obtención de las propiedades P31 de Wikidata, mientras tanto el segundo método se va a utilizar para obtener distinta información relevante y actualizada de las entidades de Wikidata que contengan la propiedad P625 y además sean instancias de alguna propiedad P31, es decir que sean georreferenciables y sean de algún tipo (pertenezcan a alguna clase).

3.2. Obtención de propiedades P31

El sistema en desarrollo debería visualizar en un mapa todas las ubicaciones de las instancias de un tipo como museo, universidad, etc. Para seleccionar el tipo en la interfaz, el usuario debería empezar a teclear su nombre (en inglés por el momento) y seleccionar el tipo que quiere de una lista de sugerencias autocompletadas. Wikidata ofrece una API con una función de autocompletado, pero es para cualquier entidad, y así genera muchas sugerencias irrelevantes. Al teclear "`mus`", sugiere *Musca* (una constellación), *Mus* (una género de roedores), etc., que son irrelevantes en el contexto del propósito de este sistema. Por eso, se decidió generar un caché local solo de entidades relevantes, y un sistema de autocompletado local sobres esa entidades, referentes a tipos de entidades (como museos, universidades, etc.) cuyas instancias (como Rijksmuseum, Universidad de Chile, etc.) tienen coordenadas geográficas.

Para la obtención de los tipos de entidades georreferenciables (propiedades P31) se pensó en varias soluciones que fueron evaluadas en función del rendimiento, escalabilidad y robustez. Las dos soluciones principales consideradas para identificar los tipos de entidades georeferenciables fueron las de usar el servicio de consulta de Wikidata, y usar el *dump* de Wikidata, para extraerlos. De estas soluciones la que más beneficios otorgaba en términos de escalabilidad y robustez, pero no de rendimiento, fue la de hacer primero un preprocesamiento del *dump* RDF de Wikidata [2] en formato NTriple (nt) para obtener todos los triples que tuvieran como predicado el IRI[3] de las páginas de Wikidata de las propiedades P31 y P625 para generar un nuevo *dump* de menor tamaño y con triples relevantes. Esta evaluación se presentará más adelante en el capitulo 5 de Evaluación.

En el Listing 4 se pueden ver dos triples del *dump* de Wikidata que tienen como sujeto el IRI correspondiente a la entidad de Chile (Q298). En el primer triple se puede ver que el predicado corresponde al IRI de la propiedad P31 y el objeto corresponde al IRI que representa una clase de la cual Chile es miembro, que es la entidad *democratic republic* (Q5255892), es decir Chile es una república democrática o en palabras más técnicas, es una instancia de la clase república democrática. En el segundo triple en cambio se aprecia que el sujeto corresponde al IRI de la misma entidad, el predicado al IRI de la propiedad P625 y el objeto corresponde a las coordenadas geográficas de la entidad Chile.

Listing 4 Triples que tienen como sujeto IRIs de las propiedades P31 y P625

```
1 <http://www.wikidata.org/entity/Q298>
  ↪ <http://www.wikidata.org/prop/direct/P31>
  ↪ <http://www.wikidata.org/entity/Q5255892> .
2 <http://www.wikidata.org/entity/Q298>
  ↪ <http://www.wikidata.org/prop/direct/P625> "Point(-71
  ↪ -33)"^^<http://www.opengis.net/ont/geosparql#wktLiteral> .
```

[2] `https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format`

[3] Un IRI (*Internationalized Resource Identifier*) dentro de un grafo RDF es una cadena UNICODE que se ajusta a la sintaxis definida en RFC 3987 [4]

3.2.1. Preprocesamiento de datos

Como se dijo anteriormente, el acceso a la base de datos de Wikidata mediante un *dump* RDF obtiene prácticamente **toda** la información de la totalidad de las entidades de Wikidata. En este sentido si se quiere hacer un análisis de los datos mediante este método resulta imperante hacer un preprocesamiento de los datos para que se obtengan resultados de una manera eficiente. Es por esto que se implementó el siguiente algoritmo para la creación de un nuevo *dump* que va a servir como *input* para el posterior análisis que se hará:

Algoritmo para la creación de un nuevo *dump*

1. Se descarga el archivo del *dump* desde Wikidata[4]
2. Se crea un nuevo archivo en formato Ntriples que contendrá el nuevo *dump*
3. Se leen los triples del *dump* descargado y por cada uno:
   (a) Se parsea el triple para obtener su **predicado**
   (b) Si el predicado corresponde al IRI de la propiedad P31 o P625, entonces se agrega al nuevo *dump*, caso contrario se ignora.

3.2.2. Análisis de datos

Una vez obtenido el nuevo *dump* es posible realizar análisis de los datos obtenidos de una manera muchísimo más eficiente que haciendo el análisis del *dump* completo, ya que como se verá a continuación la solución planteada para la obtención de las propiedades P31 relevantes necesitaba hacer mínimo dos veces un recorrido completo del *dump* y en este sentido tener que hacer más de una vez el recorrido del *dump* completo era algo muy negativo en términos de tiempo de ejecución.

Para obtener las propiedades P31 de entidades georreferenciables resulta necesario saber cuáles son las entidades del nuevo *dump* que cumplen con tener la propiedad P625. Para esto lo que se hizo primero fue implementar un algoritmo muy similar al anterior usado para la creación del nuevo *dump*. Este algoritmo crea y retorna un diccionario en memoria principal con todas las entidades que son georreferenciables, el cual usa como llave el identificador **único** (UID) para poder acceder en tiempo $O(1)$ a las entidades cuando se requiera:

Algoritmo para la obtención de diccionario de entidades georreferenciables

1. Se crea un diccionario vacío que contendrá a las entidades
2. Se lee el nuevo *dump*
3. Por cada triple:

[4]El *dump* de Wikidata se descarga desde https://dumps.wikimedia.org/wikidatawiki/entities/

(a) Se parsea el triple para obtener el **sujeto** y el **predicado**

(b) Si el predicado corresponde al IRI de la propiedad P625, entonces se guarda la entidad en el diccionario utilizando como llave el identificador único de la entidad (QID) que se obtiene del IRI del sujeto del triple. Caso contrario, se ignora.

Ahora como se tienen todas las entidades georreferenciables de Wikidata se puede hacer un análisis del nuevo *dump* para saber cuál es la distribución de los tipos de estas entidades. Además se podrá hacer un análisis de la distribución para ver cuántas entidades por cada tipo son o no georreferenciables, ya que, puede darse el caso que existan entidades que correspondan a tipos georreferenciables que sin embargo no tengan la propiedad P625, por lo que no aparecerían en el diccionario de entidades que se encontró en el paso anterior. Este análisis será de gran utilidad para la implementación posterior del buscador de entidades, una de las partes fundamentales del proyecto. En el pseudocódigo del Listing 5 se muestra el algoritmo que se utilizó para llevar a cabo la obtención de diccionarios que como llave tienen el UID de la propiedad P31 y como valor tienen un contador de las entidades que son instancias de la propiedad que tengan o no la propiedad P625, es decir que sean o no georreferenciables:

Listing 5 Pseudocódigo para obtener diccionarios de propiedades P31. El valor de cada elemento del diccionario corresponde a un contador de entidades que pueden ser georreferenciables o no

```
tipos_con_coords = {}
tipos_sin_coords = {}
tipos_set = set() # set objeto para evitar duplicados
nuevo_dump = open('nuevo_dump.nt') # archivo del nuevo dump
entidades_dict = get_entitites_with_coords_dict()
    for ntriple in nuevo_dump:
        suj, pred, obj = ntriple.split(' ') # (Sujeto, Predicado,
        ↪  Objeto)
        id_entidad = pred.getId() # ID de la entidad del predicado
        id_tipo = subj.getId() # ID de la entidad del sujeto
        if '/P31>' in pred:
            if id_entidad in entidades_dict:
                tipos_set.add(id_tipo)
                if id_tipo in tipos_con_coords:
                    tipos_con_coords[id_tipo] +=1
                else:
                    tipos_con_coords[id_tipo] = 1 # Caso base
            else:
                if id_tipo in tipos_sin_coords:
                    tipos_sin_coords[id_tipo] += 1
                else:
                    tipos_sin_coords[id_tipo] =  1 # Caso base
```

Con estos diccionarios obtenidos se puede hacer un último análisis para poder obtener el porcentaje que tiene cada propiedad P31 de entidades que son georreferenciables; esto se logra haciendo una simple división entre el número de entidades georreferenciables de la propiedad y el total de entidades, es decir la suma de las entidades que no son georreferenciables con las que si lo son. Se usará estos datos para generar un ranking que, de considerar el número de entidades georreferenciables, además considere el porcentaje, lo cual será importante al implementar el autocompletado del sistema.

Junto con el porcentaje, en esta parte de la obtención de datos y análisis también se va a obtener información extra de las propiedades P31 de Wikidata utilizando la API de Wikidata, MediaWiki, realizando la acción `wbgetentities`[5] a la cual se le van a solicitar propiedades extras como la etiqueta y la descripción de la entidad, para finalmente generar un archivo en formato JSON que contendrá toda información procesada y analizada de las propiedades de Wikidata relevantes para este proyecto. En el Listing 6 se puede ver el resultado de la API de Wikidata para la entidad *museum* (Q33506):

Listing 6 Ejemplo de respuesta que entrega la API de Wikidata para obtener información sobre entidades

```
{
"entities": {
    "Q33506": {
        "type": "item",
        "id": "Q33506",
        "labels": {
            "en": {
                "language": "en",
                "value": "museum"
            }
        },
        "descriptions": {
            "en": {
                "language": "en",
                "value": "institution that holds artifacts and other
                ↪  objects of scientific, artistic, cultural,
                ↪  historical, or other importance"
            }
        }
    }
}, "success": 1
}
```

Utilizando la información que entrega esta respuesta que da la API de Wikidata para cada propiedad P31 y junto con las estadísticas que se encontraron en pasos anteriores se

[5] `https://www.wikidata.org/w/api.php?action=help&modules=wbgetentities`

puede generar el archivo JSON final que contendrá la información de todas las propiedades P31 relevantes para este proyecto. En el Listing 7 se puede ver un extracto del archivo JSON final de las propiedades P31 de Wikidata, en concreto se presenta la información relacionada a la entidad **museo** (Q33506):

Listing 7 Ejemplo de objeto que representa la entidad museo (Q33506) junto con su información

```
"Q33506":
    {
        "label": "museum",
        "description": "institution that holds artifacts and other
        ↪  objects of scientific, artistic, cultural, historical, or
        ↪  other importance",
        "entitiesWithCoords": 33289,
        "entitiesWithoutCoords": 8722,
        "total": 42011,
        "percentage": 0.792387707981243,
    }
```

Del codigo JSON de arriba se puede desprender que el objeto representado hace referencia a la entidad cuyo UID es Q33506, su etiqueta es *museum*, su descripción "*institution that holds artifacts and other objects of scientific, artistic, cultural, historical, or other importance*", el número de entidades con y sin propiedad P625 es 33.289 y 8.722 respectivamente, el total de entidades que son instancia de este tipo es igual a 42.011 y finalmente que el porcentaje de entidades georreferenciables es 79,2 %.

3.2.3. Resultados

Los resultados finales para el preprocesamiento del *dump* de Wikidata y el posterior análisis arrojaron un total de 21.263 tipos de entidades georreferenciables para la propiedad P31, las cuales tienen un porcentaje de entidades con propiedad P625 mayor o igual a 1 %; este filtro se aplicó con el fin de quitar ruido generado por tipos con pocas entidades georreferenciables. Cabe mencionar que este análisis se hizo con el *dump* de Wikidata actualizado hasta el 20 de octubre de 2022.

3.3. Entidades georreferenciables

Para la obtención de las entidades que van a ser visualizadas en el mapa mundial se accedió a los datos utilizando *Wikidata Query Service* (WDQS). Para esto se creó la consulta detallada en el Listing 8 la cual fue ejecutada en el servicio web de WDQS para ver los resultados que entregaba.

Listing 8 Consulta SPARQL para obtener información relevante sobre entidades georreferenciables

```
SELECT DISTINCT
  ?item ?label ?description ?thumb ?image ?lat ?lon ?countryCode
  WHERE {
    # Se obtitnen todas las instancias de un ítem dado
    ?item wdt:P31/wdt:P279* wd:{UID} .
    # Se obtienen las etiquetas de los ítems
    ?item rdfs:label ?label .
    FILTER (LANG(?label)="en")
    # Para cada ítem se obtienen sus coordenadas geográficas
    ?item wdt:P625 ?coord .
    # Filtro para obtener solo ítems ubicados en el planeta Tierra
    ?item p:P625 [ psv:P625[ wikibase:geoGlobe ?globe ; ] ] .
    FILTER ( ?globe = wd:Q2 )
    # Descripción del ítem (opcional)
    OPTIONAL {?item schema:description ?description .
            FILTER (LANG(?description)="en")}
    # Imagen del ítem (opcional)
    OPTIONAL {?item wdt:P18 ?image .}
    # País del ítem (opcional)
    OPTIONAL {?item wdt:P17 ?country .
            ?country wdt:P297 ?code . }
    BIND(LCASE(?code) as ?countryCode)

    # Se separan las coordenadas en longitud y latitud
    BIND(geof:longitude(?coord) AS ?lon)
    BIND(geof:latitude(?coord)  AS ?lat)

    # Si se obtiene una imagen, se genera un thumbnail de la siguiente
     ↪ forma:
    BIND(REPLACE(wikibase:decodeUri(STR(?image)),
     ↪ "http://commons.wikimedia.org/wiki/Special:FilePath/", "") as
     ↪ ?fileName) .
    BIND(REPLACE(?fileName, " ", "_") as ?safeFileName)
    BIND(MD5(?safeFileName) as ?fileNameMD5) .
    BIND(CONCAT("https://upload.wikimedia.org/wikipedia/commons/thumb/",
                SUBSTR(?fileNameMD5, 1, 1), "/",
                SUBSTR(?fileNameMD5, 1, 2), "/", ?safeFileName,
                 ↪ "/350px-", ?safeFileName) as ?thumb)
  } LIMIT {LIM} # límite de resultados, útil cuando un tipo tiene
   ↪ muchas instancias
```

Esta consulta entrega como resultado las entidades que son instancias directas o **subclases** (propiedad P279[6]) de una propiedad P31. Por ejemplo si se desea buscar la entidad *museum*, la consulta devolverá todas las entidades que sean instancias directas de *museum* pero también devolverá aquellas entidades que sean instancias de las subclases de la misma como *archaeological museum*, *art museum*, *history museum*, entre otras. Esta propiedad viene dada por el objeto del triple, el cual es representado por su UID (ver línea 3 del Listing 8), dado como parámetro a la consulta.

Para entender cómo funciona la consulta, resulta necesario entender que son los prefijos en Wikidata y SPARQL. Los prefijos permiten escribir un URI largo en una forma mucho más corta. Algunos son internos a Wikidata como **wd, wdt, p, ps, bd** y muchos otros son prefijos externos de uso común, como **rdf, skos, owl, schema**. En relación a la consulta que se mostró anteriormente los prefijos más importantes que se van a utilizar son **wd, wdt, rdfs, schema, geof y wikibase** que serán detallados en la siguiente tabla:

Prefijo	Full URL
wd:	`http://www.wikidata.org/entity/`
wdt:	`http://www.wikidata.org/prop/direct/`
rdfs:	`http://www.w3.org/2000/01/rdf-schema#`
schema:	`http://schema.org/`
geof:	`http://www.opengis.net/def/function/geosparql/`
wikibase:	`http://wikiba.se/ontology#`

Estos prefijos serán claves para poder referenciar a la entidad de la propiedad P31 en el caso del prefijo **wd:**. También serán vitales para obtener las propiedades más relevantes de cada entidad del resultado de la ejecución de la consulta SPARQL. Algunas de estas propiedades son detalladas en la siguiente tabla:

Propiedad	ID de Wikidata Descripción
P625	Coordenadas geográficas de la entidad
P18	Imagen/es de la entidad
P17	País de la entidad
P297	Código identificador de un país en formato de dos letras según ISO 3166-1 [7]

Estas propiedades serán obtenidas usando el prefijo **wdt:** para acceder a ellas mediante el uso de los triples semánticos, ya que este prefijo vincula la entidad con el valor directamente. Un ejemplo de esto sería la línea 10 del Listing 8 de la consulta en donde se puede ver cómo se vincula directamente el valor que tiene la propiedad P625 de un ítem con la variable `?coord` de la siguiente forma: `?item wdt:P625 ?coord`. De esta manera se obtienen además de las coordenadas geográficas, la imagen, el país de origen y el código identificador del país de cada entidad.

Para obtener las propiedades restantes, es decir, la etiqueta, descripción, latitud y longitud, y *thumbnail* de las entidades se usaron otros métodos. En el caso de la etiqueta y

[6]Propiedad P279 *subclass of*, indica que un ítem es una subclase (subconjunto) de otro ítem.

descripción se utilizaron los prefijos **rdfs:label** y **schema:description** respectivamente para vincular las variables `?label` y `?description` con sus valores. Por otra parte para obtener las coordenadas de latitud y longitud de la entidad se utilizó GeoSPARQL, estándar para la representación y consulta de datos vinculados geoespaciales para la Web Semántica del Open Geospatial Consortium (OGC) [8]. Este estándar provee el prefijo **geof:** que a su vez ofrece las funciones **geof:latitude(coord)** y **geof:longitude(coord)** que reciben como argumento una coordenada de tipo `Point` y extrae los valores de latitud y longitud respectivamente. En relación a esto hay que tener en cuenta que como Wikidata tiene prácticamente toda la información del universo existen entidades que pueden ser instancias de tipos georreferenciables que no necesariamente estén en la Tierra, tal es el caso de Mons Agnes[9] una montaña ubicada en la Luna o Alba Mons[10], un volcán ubicado en Marte. Es por esto que en las líneas 12 y 13 del Listing 8 se filtran todas las entidades que **no pertenezcan** al planeta Tierra[11]. Para conseguir el *thumbnail* de las entidades se implementó una solución algo primitiva pero funcional. Es primitiva ya que lo que se hizo básicamente fue construir a mano la URL de las imágenes de las entidades para que esta no fuera del mismo tamaño que la imagen original, esto para que al momento de cargar la imagen de la entidad en la visualización del mapa no ocurrieran problemas de eficiencia, tomando en cuenta que muchas de las imágenes son de alta resolución, y pueden pesar varios megabytes. Las imágenes de Wikidata tienen como origen Wikimedia Commons [12], que es una colección de más de 88 millones de archivos multimedia como imágenes, sonidos y vídeos de libre uso a la cual cualquier persona puede contribuir. En el transcurso del proyecto se analizaron muchas imágenes de Wikimedia Commons y se logró identificar que las URLs de estas siguen el siguiente patrón sintáctico:

Dada la siguiente URL:

`https://upload.wikimedia.org/wikipedia/commons/thumb/a/a8/Tour_Eiffel_Wikimedia_Commons.jpg/200px-Tour_Eiffel_Wikimedia_Commons.jpg`

- La primera parte siempre es la misma:
  `https://upload.wikimedia.org/wikipedia/commons/thumb`
- La segunda parte corresponde al primer caracter del hash MD5[13] del nombre del archivo. En este caso, el hash MD5 de `Tour_Eiffel_Wikimedia_Commons.jpg` es
  `a85d416ee427dfaee44b9248229a9cdd`.
- La tercera parte resulta ser los primeros dos caracteres del hash MD5 anterior, es decir:
  `/a8`
- La cuarta parte corresponde al nombre del archivo, es decir:
  `/Tour_Eiffel_Wikimedia_Commons.jpg`

[8] `https://en.wikipedia.org/wiki/OGC_GeoSPARQL`
[9] `https://www.wikidata.org/wiki/Q498923`
[10] `https://www.wikidata.org/wiki/Q608793`
[11] `https://www.wikidata.org/wiki/Q2`
[12] `https://commons.wikimedia.org/wiki/Main_Page`
[13] MD5 o *Message-Digest Algorithm 5* es un algoritmo de reducción criptográfico de 128 bits usado para comprobar que un archivo no haya sido modificado

- Por último, la parte final de la URL corresponde al tamaño en píxeles de la imagen deseada, seguido del nombre del archivo de la forma:

  `/200px-Tour_Eiffel_Wikimedia_Commons.jpg`

Este mismo patrón sintáctico fue el que se implementó para conseguir el *thumbnail* de las entidades, el cual se puede ver entre las líneas 29 y 34 del Listing 8.

Por último, cabe destacar que como las entidades de Wikidata son editables por cualquier persona o máquina, existen muchas de estas que tienen campos indefinidos o que pueden volverse indefinidos en algún momento. Para el propósito de este proyecto hay campos que no pueden estar indefinidos como lo es la etiqueta de la entidad y las coordenadas geográficas, ya que forman parte esencial del autocompletado y la visualización de las entidades en el mapa respectivamente. Sin embargo hay algunas propiedades interesantes que pueden estar indefinidas y no habría mayor problema como lo son la descripción, la imagen y el país en donde está ubicada la entidad. Para esto la palabra clave `OPTIONAL` es la encargada de permitir esta ausencia de campos ya que actúa como un *left join* para devolver los resultados, y de no estar disponibles simplemente no devuelve nada para ese campo pero si lo hace para los otros. En el Listing 9 se puede ver un ejemplo de respuesta que entrega WDQS al ejecutar la consulta del Listing 8.

Listing 9 Ejemplo de respuesta para una entidad de tipo *museum* que entrega WDQS al ejecutar la consulta del Listing 8)


```
{
    "item":{
        "type":"uri",
        "value":"http://www.wikidata.org/entity/Q4890"
    },
    "label":{
        "xml:lang":"en",
        "type":"literal",
        "value":"Gemäldegalerie Alte Meister"
    },
    "description":{
        "xml:lang":"en",
        "type":"literal",
        "value":"art gallery in Dresden, Germany"
    },
    "lon":{
        "datatype":"http://www.w3.org/2001/XMLSchema#double",
        "type":"literal",
        "value":"13.734707"
    },
    "lat":{
        "datatype":"http://www.w3.org/2001/XMLSchema#double",
        "type":"literal",
        "value":"51.053388"
    },
    "image":{
        "type":"uri",
        "value":"http://commons.wikimedia.org/wiki/Special:FilePath/Dres
        ↪ den-Zwinger-Courtyard.11.JPG"
    },
    "countryCode":{
        "type":"literal",
        "value":"de"
    },
    "thumb":{
        "type":"literal",
        "value":"https://upload.wikimedia.org/wikipedia/commons/thumb/2/
        ↪ 24/Dresden-Zwinger-Courtyard.11.JPG/350px-Dresden-Zwinger-Co
        ↪ urtyard.11.JPG"
    }
}
```


Con esto termina el presente capitulo de Datos, en el cual se pudo ver todo lo relacionado

al preprocesamiento, tratamiento, análisis y obtención de datos del proyecto. Se pudo ver desde la obtención de las propiedades P31 mediante el *dump* de Wikidata hasta la obtención de las entidades georreferenciables y sus propiedades más relevantes utilizando principalmente el servicio de consultas WDQS. Esto será de gran ayuda para entender la implementación del buscador con auto-completado y la visualización de las entidades en el mapa mundial del sistema, funcionalidades y características esenciales que serán explicadas en el siguiente capítulo.

Capítulo 4

Sistema

En los siguientes párrafos se explicará la arquitectura del sistema, corresponde a la clásica arquitectura de cliente-servidor. Se comenzará por presentar el *backend* del sistema en donde ocurren los procesos de preprocesamiento y análisis de datos, y donde además se implementa una API para realizar la comunicación entre el cliente y el servidor. Luego se explicará la parte de la arquitectura del cliente, la cual comprende al *frontend* de la aplicación. Finalmente se mostrará un diagrama correspondiente al resumen de la arquitectura del sistema.

4.1. *Backend* / API

Como se comentó anteriormente en el capitulo de Datos, para la obtención de los datos se utilizó principalmente el *dump truthy* de Wikidata para en primer lugar realizar un preprocesamiento de datos para obtener un nuevo *dump* reducido con el cual, después de realizar análisis sobre el mismo, se obtiene el JSON con todos los tipos de entidades georreferenciables de Wikidata, junto con la ayuda de la MediaWiki API. Este proceso ejecutado en el servidor es parte esencial del sistema ya que el JSON generado servirá como fuente de datos para la generación del autocompletado de la barra de búsqueda que será explicada más adelante. También como se comentó anteriormente en la sección 3.3, se implementó una consulta SPARQL (ver Listing 8) que al ser ejecutada en WDQS entrega como resultado las entidades georreferenciables.

Para poder realizar la conexión entre el cliente y el servidor lo que se hizo fue implementar una API utilizando el *framework* Flask[1] de Python dado a la familiaridad del lenguaje y que se trata de un *framework* muy minimalista y fácil de usar. Una de las fuentes de datos de esta API corresponde al archivo JSON de los tipos de entidades georreferenciables que se obtuvo luego de realizar el análisis del *dump* de Wikidata (ver Sección 3.2.2). Por otra parte para la obtención de los datos de las entidades georreferenciables (ver Sección 3.3) se utilizó el *endpoint* de WDQS `https://query.wikidata.org/sparql` a la cual se le pasa como parámetro la consulta SPARQL que se vió con anterioridad en el Listing 8.

[1] `https://flask.palletsprojects.com/en/2.2.x/`

4.1.1. API *endpoints*

Tipos de entidades georreferenciables

Este *endpoint* se encarga de devolver el contenido del archivo JSON de los tipos de entidades georreferenciables. Su sintaxis es la siguiente.

- `GET $API_SERVER$/types`

La respuesta que entrega este *endpoint* se puede ver en el Listing 10, en el cual se puede apreciar que se retorna la cantidad total de tipos de entidades encontrados en el preprocesamiento y análisis, y un diccionario de objetos similares al mostrado en el Listing 7, en el cual la llave corresponde al identificador único de la entidad y los valores a la información relevante. Cabe mencionar que se decidió por dejar a los tipos en un diccionario de objetos para poder acceder en tiempo $O(1)$ a la entidad cuando se le requiera.

Listing 10 Respuesta que entrega el endpoint `/types`

```
{
    "count": COUNT_OF_P31_TYPES,
    "types": TYPES_DICT
}
```

Información sobre tipo de entidad georreferenciable

Este se trata de otro *endpoint*, el cual se encarga de devolver información sobre un tipo de entidad georreferenciable. Su sintaxis es la siguiente:

- `GET $API_SERVER$/type/<id>`

En donde el parámetro `<id>` se trata de un *string* que corresponde al identificador único de la entidad. En el Listing 11 se puede ver la respuesta luego de ejecutar el *endpoint* con parámetro Q928830 cuya etiqueta es *metro station*.

Listing 11 Objeto que contiene la información del tipo de entidad cuyo identificador es Q928830 (*metro station*)


```
{
  "description": "railway station of a rapid transit system",
  "entitiesWithCoords": 14865,
  "entitiesWithoutCoords": 530,
  "label": "metro station",
  "percentage": 0.9655732380643066,
  "total": 15395
}
```


Autocompletado

Este endpoint se trata de uno de los más importantes ya que entrega como resultados las sugerencias de autocompletado de la barra de búsqueda de la aplicación. Su sintaxis es la siguiente:

- `GET $API_SERVER$/autocomplete/<search>`

En donde el parámetro `<search>` corresponde a un *string* que representa la búsqueda de la etiqueta de una entidad en la barra de búsqueda del sistema.

El algoritmo que se utilizó para conseguir los resultados fue el siguiente:

1. La primera vez que se llama al *endpoint* se abre el archivo JSON de tipos de entidades almacenando los datos en una variable global en memoria. De esta forma se evita tener que abrir el archivo cada vez que se llama al *endpoint*, lo cual supondría problemas de rendimiento.
2. Se itera sobre los ítem del diccionario de tipos del JSON y por cada ítem:
   (a) Se obtiene su etiqueta y se comprueba si su comienzo coincide con la búsqueda o si dentro de la etiqueta se encuentra la misma. Si alguna de estas condiciones se cumple entonces se guarda el objeto en un diccionario, en donde la llave corresponde al identificador único de la entidad y los valores a su información. Caso contrario, si no se cumple alguna de las condiciones, se ignora.
3. Una vez que se obtiene el diccionario de objetos se procede a ordenarlos para generar un ***ranking*** de tipos. Este ranking es múltiple ya que ordena a las entidades tomando en cuenta primero la cantidad de instancias georreferenciables que tiene y luego el porcentaje de entidades georreferenciables del total.

En el Listing 12 se puede ver la respuesta del endpoint de la API luego de realizar la búsqueda ***museum***, en la cual se obtuvieron 211 resultados y en donde se puede comprobar

evidentemente lo que se dijo anteriormente, ya que el primer resultado tiene como etiqueta *museum* y el segundo *art museum*, además se puede ver que el orden se realiza primero considerando la cantidad de instancias georreferenciables y luego el porcentaje del total que lo son.

Listing 12 Extracto de respuesta de *endpoint* de autocompletado al buscar la etiqueta *museum*

```
{
  "count": 211,
  "search": "museum",
  "types": [
    [
      "Q33506",
      {
        "description": "institution that holds artifacts and other
        ↪  objects of scientific, artistic, cultural, historical, or
        ↪  other importance",
        "entitiesWithCoords": 33289,
        "entitiesWithoutCoords": 8722,
        "label": "museum",
        "percentage": 0.792387707981243,
        "total": 42011
      }
    ],
    [
      "Q207694",
      {
        "description": "building or space for the exhibition of art (for
        ↪  institution, use Q3196771)",
        "entitiesWithCoords": 5463,
        "entitiesWithoutCoords": 822,
        "label": "art museum",
        "percentage": 0.8692124105011934,
        "total": 6285
      }
    ], ...
  ]
}
```

Entidades georreferenciables

El otro *endpoint* clave de la API corresponde al que genera los resultados luego de ejecutar la consulta SPARQL del Listing 8 en WDQS. Su sintaxis es la siguiente:

- `GET $API_SERVER$/data/<search>/<limit>`

En donde `<search>` es un *string* que representa la búsqueda de la etiqueta del tipo de entidad que se desea buscar (ej: *museum*, *beach*, *street*) y `<limit>` que es un *int* que corresponde al límite de resultados que se espera de la consulta[2], el cual es pasado como parámetro a la consulta del Listing 8 (ver última línea de la consulta). Este parámetro puede tomar valores de 0 a 5, donde si es 0 significa que la consulta no tiene límite (valor por defecto), y si tiene entonces la consulta va a buscar 10^n resultados, donde $n \in \{1, 2, 3, 4, 5\}$.

Para que la consulta no tenga errores de sintaxis por el tipo de parámetro de tipo *string* que recibe, lo que se debe hacer antes de ejecutar la consulta y enviar la petición a WDQS; es convertir la etiqueta del tipo que se desea buscar en su identificador único. Para esto se implementó una función que, dado un *string* retorna el identificador único de la entidad. Esta función se implementó utilizando la MediaWiki API, en concreto realizando la acción `wbsearchentities`, la cual recibe como parámetro un *string* correspondiente a la búsqueda y devuelve la información de las propiedades relacionadas a la entidad, una de ellas su identificador. Una vez se tiene el identificador único de la entidad, se envía una petición HTTP GET al endpoint de WDQS, que recibe como parámetro la consulta principal de SPARQL para la obtención de entidades (ver Listing 8) con sus respectivos parámetros que se acaban de comentar.

Un ejemplo de respuesta para una entidad que entrega esta consulta se puede ver en el Listing 9, que como se puede apreciar, tiene información detallada que no es del todo relevante. Es por esto que como último paso, para la obtención de los datos de las entidades, se realizó un parseo y formateo de la respuesta que entrega el *endpoint* de WDQS para que sea mucho más fácil acceder a los campos de los resultados y ahorrar espacio en memoria. En el Listing 13 se puede ver la misma respuesta que se presentó anteriormente, pero formateada.

Listing 13 Respuesta para un tipo de entidad de WDQS formateada que contiene solo información relevante (mismos datos que Listing 9).

```
{
    "item":"http://www.wikidata.org/entity/Q4890",
    "label":"Gemäldegalerie Alte Meister",
    "description":"art gallery in Dresden, Germany",
    "thumb":"https://upload.wikimedia.org/wikipedia/commons/thumb/2/24/D
    ↪ resden-Zwinger-Courtyard.11.JPG/350px-Dresden-Zwinger-Courtyard.
    ↪ 11.JPG",
    "image":"http://commons.wikimedia.org/wiki/Special:FilePath/Dresden-
    ↪ Zwinger-Courtyard.11.JPG",
    "lat":"51.053388",
    "lon":"13.734707",
    "countryCode":"de"
}
```

Una vez explicado esto, en el Listing 14 se puede ver cómo queda finalmente la respuesta que retorna el endpoint. La cual contiene la cantidad de entidades que se encontraron y

[2] `https://www.w3.org/TR/sparql11-query/#modResultLimit`

una lista de objetos que representan a las entidades, siguiendo el formato presentado en el Listing 13.

Listing 14 Respuesta del *endpoint* valida, cuando se explicita un límite, en este caso 100.

```
{
  "count": 100,
  "results": [
    entityObject1, entityObject2, ... , entityObject100
  ]
}
```

Si se llegase a presentar algún error al momento de recibir y ejecutar las peticiones a la API, la respuesta que entrega el *endpoint* cambia totalmente y queda como se presenta en el Listing 15.

Listing 15 Respuesta cuando un error externo ocurre.

```
{
  "count": n,
  "search" search,
  "status": 500
}
```

Estos errores en la API se pueden presentar debido a una desactualización de las etiquetas de las entidades de Wikidata, que al momento de buscarlas, harán *match* con ninguna y no se encontrará el identificador de entidad del tipo; también se puede deber a una eliminación de la entidad de la base de datos de Wikidata debido a que el nuevo *dump* no estaría actualizado y por ende el archivo JSON de los tipos tampoco; en estos casos n sera igual a -2. Por otra parte cuando al querer obtener las instancias de algún tipo georreferenciable estas sean muchas WDQS entrega un error interno, típicamente un error de *Timeout* luego de 60 segundos de búsqueda, caso donde n será -1. El valor *search* corresponde a la etiqueta de la búsqueda que se quiso hacer. Por último el `"status": 500` corresponde al *status* de error del servidor cuando se presentan estos errores. De esta manera el *front* de la aplicación podrá manejar estos errores y sabrá qué hacer para cada caso.

4.2. *Frontend*

Para el front de la aplicación se decidió por usar el *framework* de Javascript, React.js[3] ya que se contaba con experiencia trabajando con este y cuenta con mucha documentación en Internet, lo cual es muy relevante cuando se encuentran trabas en el desarrollo. Para

[3] `https://reactjs.org/`

implementar algunas funciones y algunos procesos del sistema se utilizó *vanilla* Javascript. Para el aspecto visual del sistema se utilizó el *framework* Bootstrap[4], el cual es muy fácil de usar y permite un desarrollo veloz de una interfaz simple, intuitiva y que además está diseñado para facilitar el proceso de desarrollo para sitios web responsivos y orientados a los dispositivos móviles, y al igual que React.js tiene una extensa documentación.

4.2.1. Barra de navegación

El sistema cuenta con una barra de navegación (ver figura 4.1) en la parte superior de todas las páginas en donde se puede acceder a la página de inicio (*Home*) del sistema y al *About* del proyecto sitio en donde se encuentran información general sobre el sistema, la forma en cómo se usa la aplicación con un ejemplo concreto y algunos de los conceptos claves para entender el desarrollo del sistema como lo son la Web Semántica, Wikidata, Wikidata Query Service y MediaWiki API.

Figura 4.1: Barra de navegación del sistema

4.2.2. Página principal

La página principal del sistema (ver figura 4.2) cuenta con una barra de búsqueda donde el usuario debe ingresar la etiqueta del tipo de entidad que se quiera buscar, un selector para poner un límite de resultados a la consulta SPARQL que se ejecuta y un mapa mundial creado gracias a la librería Leaflet[5] en donde se van a visualizar las entidades.

[4] `https://getbootstrap.com/`

[5] En concreto se usó el paquete `react-leaflet` para el uso de componentes

Figura 4.2: Página principal del sistema

Si se quiere realizar una búsqueda, cada vez que el usuario escriba algo en la barra de búsqueda, la aplicación hará una petición HTTP GET al endpoint de la API que devuelve las sugerencias de autocompletado. Su respuesta ayuda a generar una lista de opciones para que el usuario seleccione una de las disponibles en la base de datos de los tipos. En la figura 4.3 se puede ver la lista que se despliega si el usuario ingresa la etiqueta *stadium*.

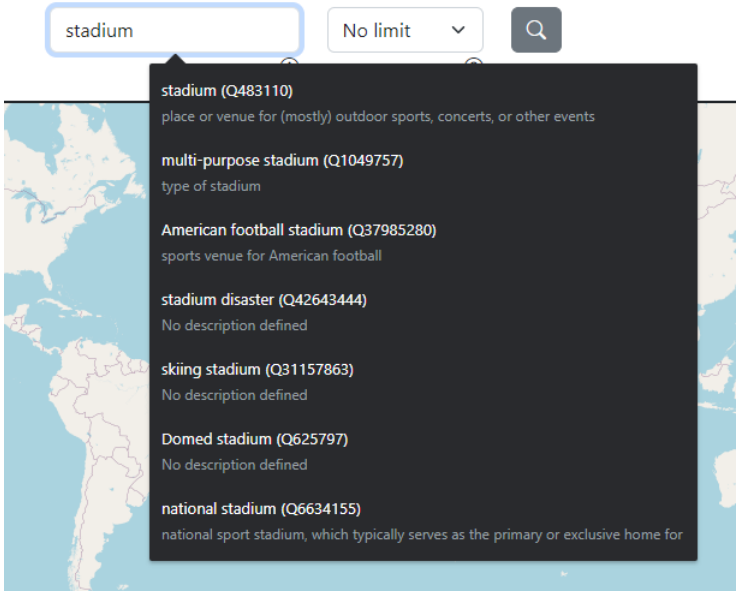

Figura 4.3: Autocompletado de búsqueda

Se puede apreciar también que al lado derecho de la barra de búsqueda se encuentra un selector (ver figura 4.4) que puede ser usado para definir un límite de entidades que se desea obtener. Este puede ser 10, 100, 1.000, 10.000, 100.000, o sin límite (valor por defecto).

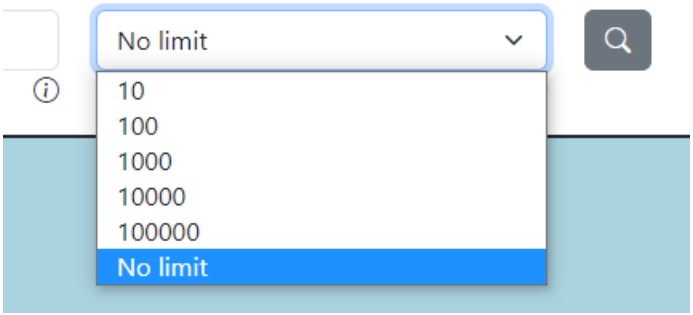

Figura 4.4: Selector de límite de respuestas

4.2.3. Visualización de entidades

Cuando el usuario ejecute la consulta (haga *click* en el boton con lupa, para buscar), el *front* del sistema hará una petición HTTP GET al *endpoint* de la API que devuelve las instancias de un tipo georreferenciable a la cual se le pasa como parámetro la etiqueta de la búsqueda y el límite si es que se especificó uno.

Para la georreferenciación de las entidades, se leen los campos de *latitude* y *longitude* que entrega la respuesta de la API para cada entidad y se ubican en un mapa generado gracias a la herramienta Leaflet, en concreto gracias al paquete **React Leaflet**[6]. En la figura 4.5

[6] `https://react-leaflet.js.org/`

se pueden ver las entidades georreferenciadas luego de realizar la búsqueda de instancias de tipo *stadium*.

Figura 4.5: Visualización de entidades de tipo *stadium* georreferenciadas en el mapa mundial

Se puede ver que ahora en el mapa aparecen algunos marcadores que representan a entidades georreferenciadas y además aparecen *clusters* que corresponden a conjuntos de entidades. Si se presiona sobre un *cluster* el mapa automáticamente va a hacer *zoom in* para ver con más detalles la distribución geoespacial de las entidades correspondiente al *cluster*. Por otra parte si se presiona un marcador individual, se mostrará un *pop-up* que contendrá información sobre la entidad como su etiqueta, su identificador único y perfil de Wikidata, su descripción, país al que pertenece, coordenadas geográficas y una imagen si es que tiene, esto se puede ver en la figura 4.6.

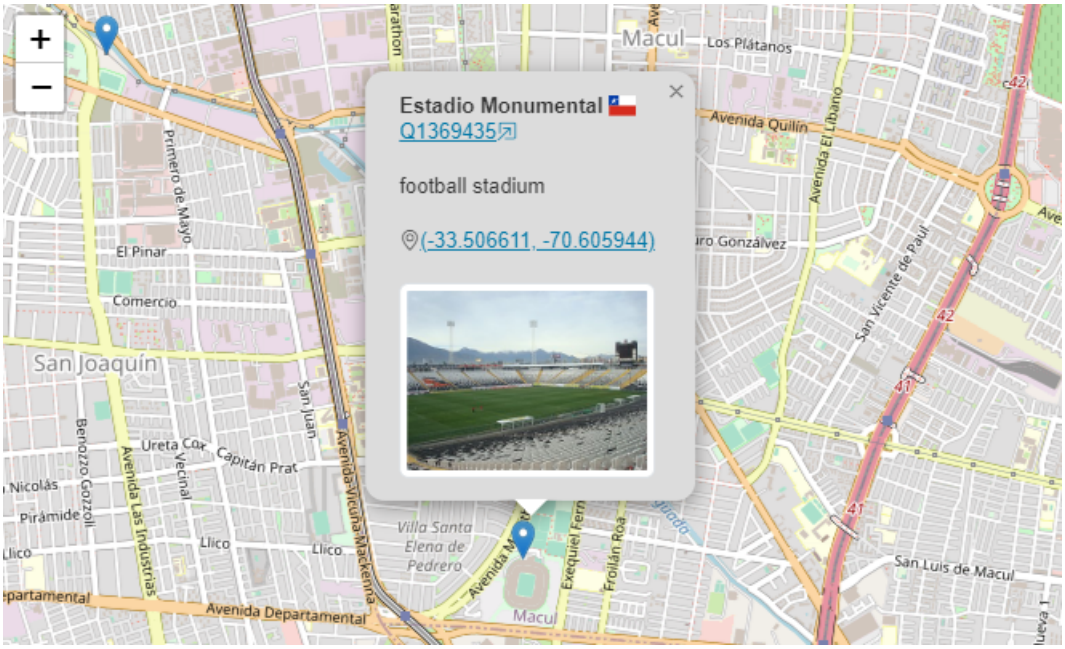

Figura 4.6: *Popup* que se genera cuando se hace *click* en un marcador

Si se hace click en el identificador de la entidad, se abre en una nueva pestaña con el perfil de la entidad en Wikidata; por otra parte, si se hace click en las coordenadas, se abre una nueva pestaña con la visualización de la misma ubicación geográfica pero utilizando la herramienta de Google Maps. Por último, si se hace click en la imagen se abrirá una nueva pestaña en el navegador con la visualización de la misma imagen en su máxima resolución posible.

4.2.4. Resultados de búsqueda

Finalmente, una vez que se realiza la búsqueda, abajo del mapa se genera un cuadro con información básica del tipo como su etiqueta, perfil de Wikidata y descripción; junto con el número de instancias que se encontraron (ver figura 4.7).

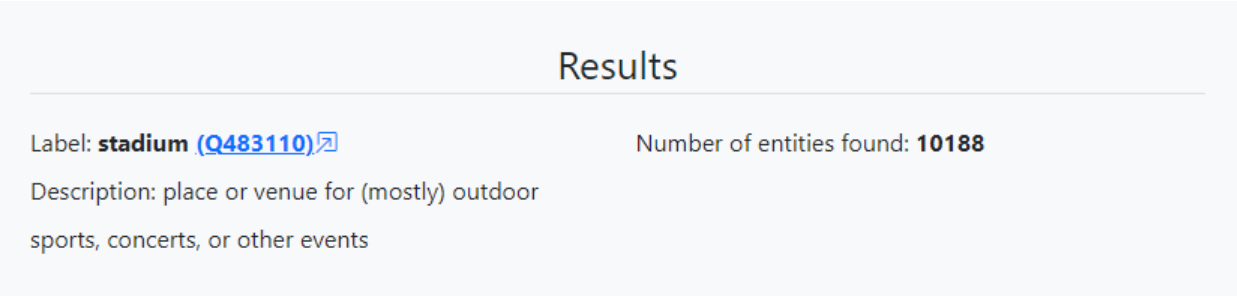

Figura 4.7: Cuadro de resultados de la búsqueda

4.3. Resumen de la arquitectura

En la figura 4.8 se puede ver un resumen de la arquitectura de cliente-servidor que se implementó para el sistema Wikidata Atlas. En este resumen aparecen los distintos procesos de preprocesamiento, análisis, el flujo de datos; también aparecen las distintas herramientas que se utilizaron para desarrollar cada parte del proyecto y la forma en como se comunica el **cliente** con el **servidor**.

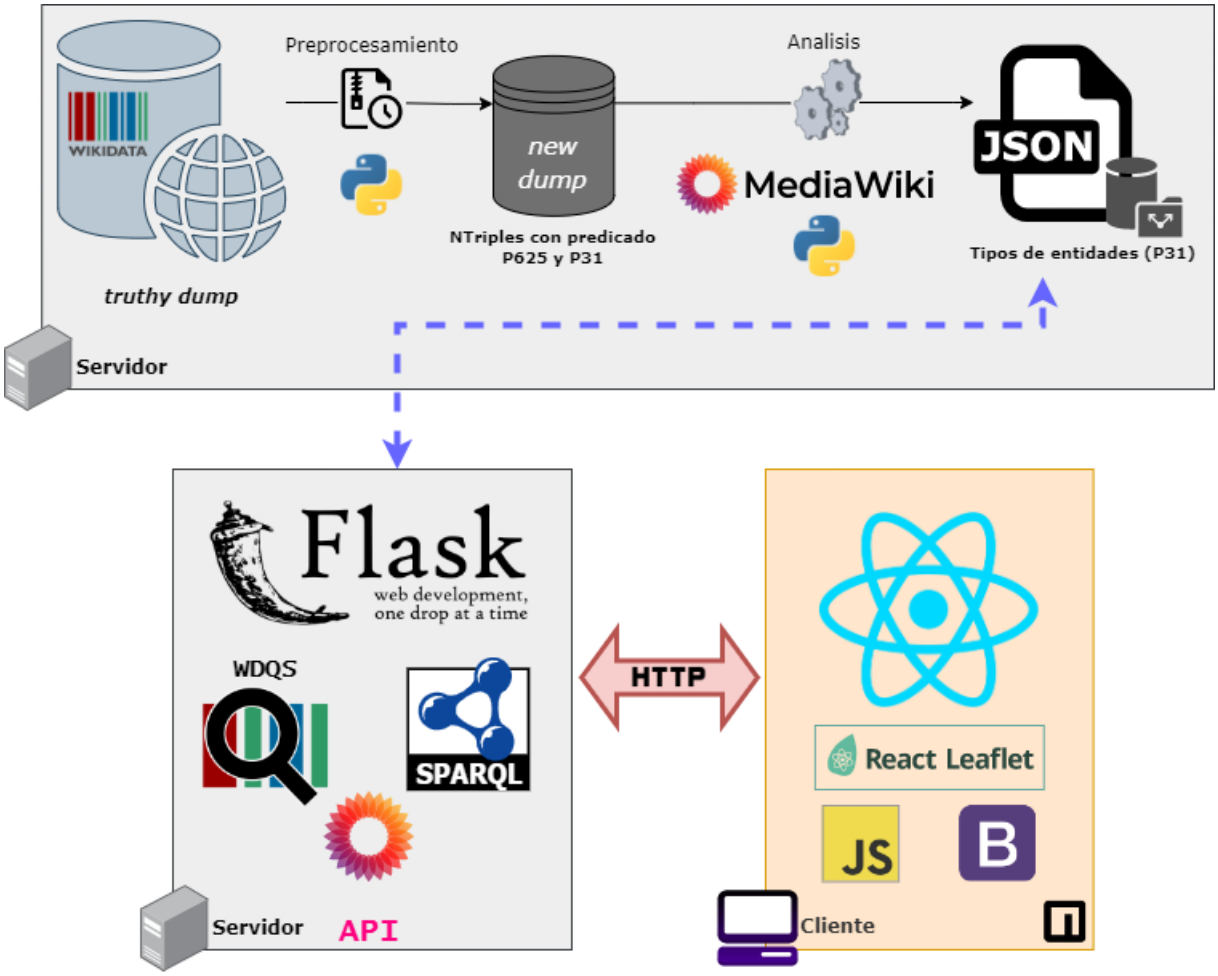

Figura 4.8: Arquitectura del sistema

Capítulo 5

Evaluación

En el actual capítulo se presentará el trabajo relacionado a la evaluación del sistema. Gracias a a este capítulo se podrán responder preguntas como: ¿cuánto demora el preprocesamiento del *dump* de Wikidata?, ¿existen diferencias de contenido entre *dumps* de distintas fechas?, ¿las funciones de la API tienen buen rendimiento?, ¿cuál es la percepción de los usuarios sobre el sistema?, ¿es esta aplicación fácil de usar?.

En primer lugar se verán experimentos relacionados al rendimiento que tiene el preprocesamiento y análisis de la base de datos del sistema, en particular, el *dump* de Wikidata para dos formatos de compresión (gzip y bzip2). Por otra parte también se vera el rendimiento que tienen las distintas funciones implementadas en la API del sistema, como el autocompletado y la obtención de las entidades georreferenciables para todos los tipos disponibles. Por último se va a presentar la percepción que tuvieron usuarios del sistema y resultados de una encuesta aplicada a los mismos para ver el nivel de usabilidad de la interfaz de la aplicación desarrollada.

5.1. Wikidata *truthy dumps*

5.1.1. Experimento de rendimiento

Como se comentó anteriormente en el capítulo de Datos, existen múltiples formas de acceder a la base de datos de Wikidata; una de estas es mediante *truthy dumps* que contienen la totalidad de la base de datos de Wikidata y están disponibles en formato gzip y bzip2. Como estos datos se actualizan semanalmente sería muy costoso en términos de tiempo tener siempre una versión actualizada de la base de datos por lo que se decidió por usar los *dumps* en ambos formatos disponibles correspondientes al mes de octubre de 2022 para hacer el análisis.

Al descargar ambos archivos se pudo dar cuenta de que uno era de mayor tamaño que otro, esto se debe a que la compresión del tipo bzip es mucho mayor a la de gzip[1]. Esto

[1] A Quick Benchmark: Gzip vs. Bzip2 vs. LZMA

ocasiona que dependiendo del ancho de banda de dónde se descargan estos archivos sea el de formato bzip el que menos se demore en descarga. Esto puede ser muy ventajoso ya que ambos archivos tienen la misma información y ahorrar tiempo de descarga siempre es algo positivo ¿Pero qué pasa cuando se quiere probar el rendimiento cuando se quiere hacer un preprocesamiento y análisis de datos?

Para este experimento lo que se hizo fue calcular el tiempo que tardan los algoritmos de preprocesamiento y análisis de datos explicados en el capitulo de Datos en ser ejecutados al utilizar los *dumps* en formato bzip y gzip. Los resultados se pueden apreciar en la gráfica de la Figura 5.1.

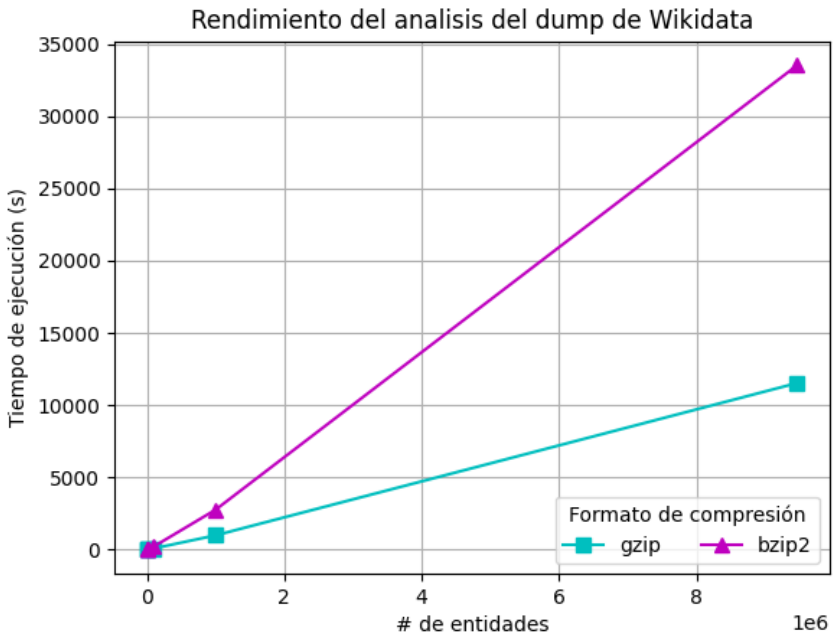

Figura 5.1: Rendimiento de los *dump* al ejecutar tareas de preprocesamiento y análisis

Como se puede ver, a medida que aumenta la cantidad de entidades que se quieren obtener del dump, el tiempo de ejecución aumenta de manera lineal para ambos formatos. Al comienzo no se aprecia una diferencia de tiempo significativa pero se puede ver que cuando se quieren obtener todas la entidades posibles la diferencia si es significativa, y es el tipo de formato gzip el que entrega mejores resultados realizando las tareas en un poco más de un tercio del tiempo que demora bzip. Es por esto que en este *trade-off* existente entre el tiempo de descarga y el tiempo de ejecución de tareas es preferible usar el *dump* en formato gzip.

5.1.2. Experimento de contenido

Otro experimento interesante que se puede hacer es analizar el contenido que tienen dos *dumps* de distintas fechas, ya que independiente del tipo de compresión, los *dumps* que genera Wikidata en una misma semana tienen exactamente el mismo contenido, no así aquellos que

sean de distintas semanas. Este experimento nos permitirá entender qué tan rápidamente se desactualizaron los datos cacheados localmente para facilitar el autocompletado.

Gracias a un trabajo de preprocesamiento y análisis que se realizó con anterioridad se podrá hacer la comparación del contenido de un *dump* de Wikidata del mes de mayo de 2022 con el dump en formato gzip de octubre de 2022.

Después de analizar el contenido se obtuvo que el *dump* más antiguo correspondiente al mes de mayo del presente año tiene un total de 20.473 tipos de entidades que tienen instancias georreferenciables, mientras tanto el más actualizado correspondiente a octubre tiene un total de 21.287 entidades, esto supone un aumento de casi un 4 % en el número de entidades georreferenciables.

Por otra parte se pudo calcular el número de entidades que formaban parte del *dump* más antiguo que después no aparecían. Esta cantidad corresponde a 580 entidades que desaparecieron del JSON de tipos georreferenciables.

Los siguientes experimentos fueron hechos utilizando el *dump* más actualizado con el que se contaba, es decir el de octubre de 2022.

5.2. Autocompletado de tipos

Para la evaluación del autocompletado del sistema lo que se hizo fue realizar la búsqueda para todas las letras del alfabeto, luego para más generalidad se realizó el mismo procedimiento pero para todas las combinaciones posibles de 2 letras del alfabeto. Considerando que el alfabeto tiene 26 letras, se realizó la búsqueda primero para las 26 letras y luego para las 26^2 combinaciones posibles.

Esta evaluación se hizo utilizando el *endpoint* de la API para obtener los resultados de autocompletado para una búsqueda cualquiera de largo variado. Los resultados de este experimento se pueden ver en la figura 5.2, en donde se presenta un *scatter plot* con los tiempos que demoran los resultados en ser obtenidos junto con la cantidad de entidades, esto caracterizado para cada largo de búsqueda, que puede ser 1 o 2.

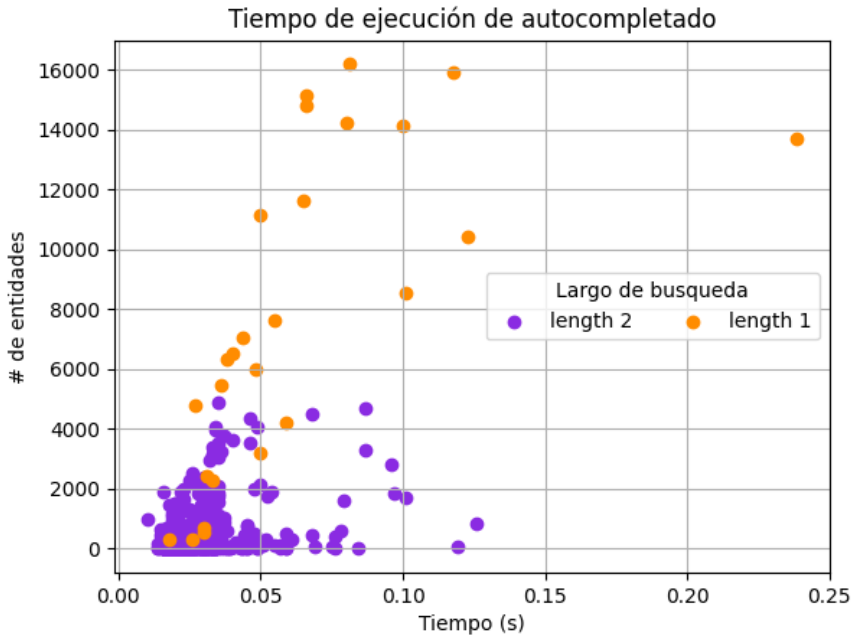

Figura 5.2: *Scatter plot* del tiempo y número resultados obtenidos para búsquedas de largo variado para el autocompletado

De la visualización se puede desprender que en general los tiempos de respuesta son muy bajos, es decir apenas el usuario empieza a escribir una búsqueda, el autocompletado muestra sugerencias de entidades. Por otra parte también se puede ver que existen muchas búsquedas que no entregan resultados, esto debido a que la combinación de dos letras para las 26 disponibles del abecedario genera muchas palabras sin sentido como *kj*, *pk*, *bt*, entre otras; que no entregan resultados validos.

Por otra parte para ver la distribución de los tiempos se generó un *boxplot* de los tiempos que demora en entregar las respuestas el autocompletado del sistema. Este gráfico se puede ver en la figura 5.3, del cual se desprende que existe una concentración de los datos desplazada hacia tiempos muy escasos de procesamiento. Lo cual es algo muy positivo para el autocompletado y lo será para el usuario, ya que las respuestas cada vez que escriba algo se van a actualizar de una manera muy rápida.

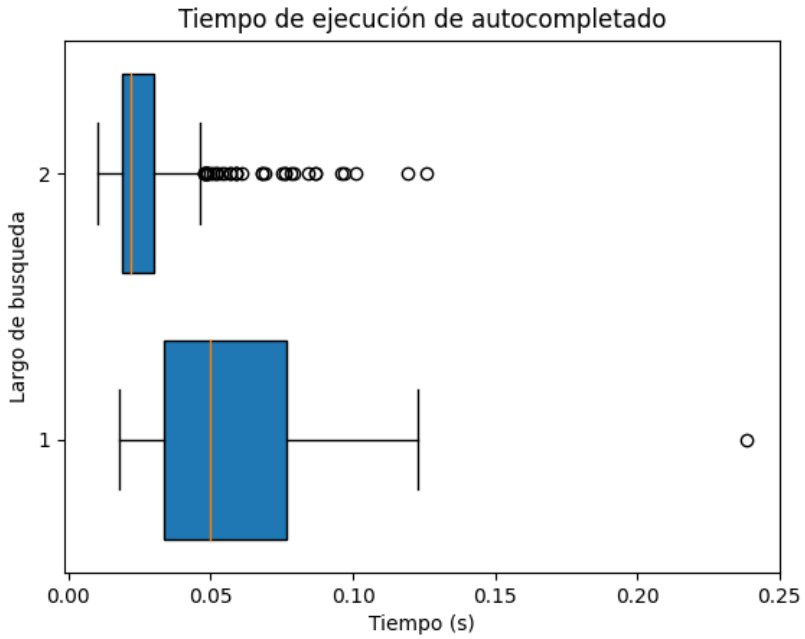

Figura 5.3: *Boxplot* de los tiempos de respuesta del autocompletado para búsquedas de largo variado (1 y 2)

5.3. Búsqueda de instancias de tipos

Para realizar la evaluación de la búsqueda de las entidades georreferenciables, lo que se hizo fue, para cada tipo presente en el JSON que se obtuvo del análisis del *dump* de Wikidata, realizar la búsqueda de su etiqueta utilizando el *endpoint* (sin límite) de la API, que obtiene los resultados de entidades georreferenciables dado un tipo. Los resultados fueron graficados en un *scatter plot* que se puede ver en la figura 5.4, donde el valor x indica el tiempo de respuesta, y valor y indica el número de entidades devueltas por la consulta.

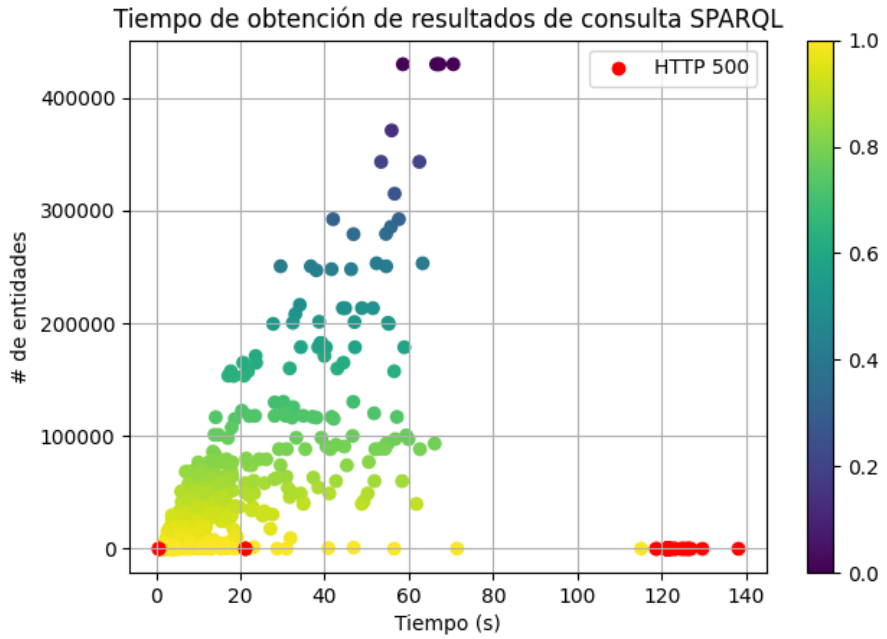

Figura 5.4: *Scatter plot* del tiempo de ejecución de la consulta para todos los tipos georreferenciables obtenidos del preprocesamiento de los datos de Wikidata

Los puntos rojos en el gráfico corresponden a errores en el servidor de WDQS (HTTP 500). Uno de ellos ocurría cuando una etiqueta de entidad P31 no se correspondía a ningún identificador único, esto se puede deber a una eliminación de la entidad de la base de datos de Wikidata después de la descarga del *dump*. Esto no debería ser algo tan grave ya que es muy difícil que tipos con muchas instancias georreferenciables sea eliminado de la base de datos, de hecho lo más seguro es que aquellos tipos eliminados o editados no tengan muchas instancias georreferenciables. El otro error ocurría cuando un tipo de entidad tenía muchas instancias, lo cual hacía que la consulta demorara mucho en ser procesada. Esto ocurría principalmente por ejemplo cuando las búsquedas se trataban sobre tipos geográficos como montañas, ríos, lagos; lo mismo pasaba con tipos de edificios como hospitales, escuelas o iglesias. Cuando ocurría este problema, el servidor de WDQS después de 60 segundos entregaba un error de *TimeoutException* que se propagaba a la API del proyecto, por esto fue que se implementó un selector de límite (ver Figura 4.4), para no obtener un error del servidor y para que el mapa pudiera generar los marcadores de las instancias de estos tipos que generaban problemas.

De esta visualización se puede desprender que existe una correlación entre la cantidad de entidades que la consulta devuelve y el tiempo que se demora en procesarla, lo cual tiene mucho sentido.

También se puede observar en los *boxplot* de la Figura 5.5 que existe una concentración sesgada hacia tiempos despreciables para la mayoría de las búsquedas y obtención de las entidades que son tipos georreferenciables. Además se puede observar que existen muchos *outliers*, una parte de estos, cuyos tiempos de ejecución de búsquedas fluctúan entre 110 y 140 segundos aproximadamente corresponden a aquellas entidades que provocan el error de *Timeout* en WDQS que se comentó anteriormente. La tabla 5.1 muestra los resultados

obtenidos de los tiempos de obtención de las instancias de los tipos para la totalidad de los datos, los que entregaron respuesta positiva del servidor y aquellos que entregaron errores (HTTP *Status* 500)

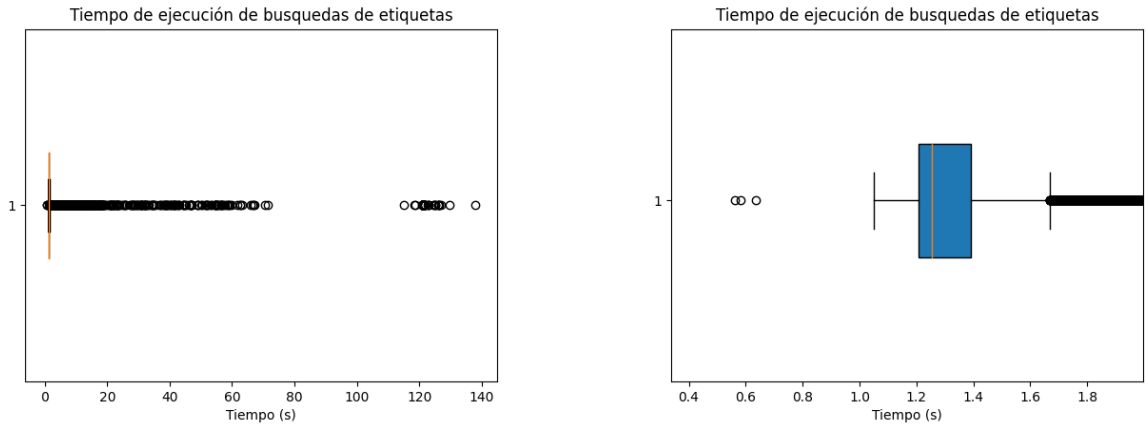

Figura 5.5: *Boxplots* de los tiempos de ejecución de la búsqueda de las etiquetas de los tipos georreferenciables de Wikidata (el *boxplot* de la derecha es el mismo que el otro, pero con zoom para visualizar mejor los resultados)

Tiempo (s)	Totalidad	*Status* 200	*Status* 500
promedio	2,59	1,78	120,16
desv. estándar	10,35	3,44	8,88
máximo	138,08	66,48	138,08
mínimo	0,56	0,56	66,87

Tabla 5.1: Estadísticas de los tiempos de obtención de las instancias sin límite de los tipos georreferenciables. Se divide la tabla en los resultados para la totalidad de los datos, es decir los más de 21 mil tipos obtenidos; los que entregaron respuesta del servidor (*Status* 200) y los que entregaron errores (*Status* 500)

.

De la tabla se puede inferir, que en general los tiempos de respuesta que tiene la API para devolver las instancias de un tipo sin límite son muy positivos, ya que en promedio las respuestas para aquellos tipos que no generan errores es de 1,779 segundos. Además la desviación estándar es baja, es decir existe una concentración de tiempos muy bajos, en otras palabras no hay una alta dispersión. Por otra parte para los tipos que entregan errores es algo inusual que el promedio sea aproximadamente el doble de lo que demora WDQS en devolver un error de *Timeout* (60 segundos). Quizás esto tenga que ver con una repetición de petición de datos que hace la API al servidor de WDQS. Para impedir que el sistema demore más de lo debido, se implementará un manejo de errores más robusto en cuanto al tiempo que toman las consultas y peticiones al servidor en ser ejecutadas.

5.4. Percepción de usuarios

Para realizar la evaluación con usuarios, lo que se hizo fue crear un cuestionario que en una primera parte contenía distintas tareas de usabilidad del sistema y en una segunda parte una encuesta para medir la usabilidad. Este cuestionario fue compartido en el foro de alumnos de un curso dictado por el Departamento de Ciencias de la Computación (DCC) de la Universidad de Chile que se relaciona directamente con los conceptos y herramientas del proyecto, el cual es La Web de Datos. También fue compartida en el foro de la comunidad del DCC, con más gente en otros espacios académicos como cursos de la universidad y con parte de la comunidad de Wikidata mundial.

Las tareas que debían ejecutar los usuarios en la primera parte fueron las siguientes:

1. Buscar entidades de tipo *stadium*.
2. Buscar entidades de tipo *metro station*.
3. Buscar entidades de tipo *clandestine center of detention and torture*.
4. Buscar entidades de tipo *educational organization* (como sugerencia se puede agregar límite).
5. Navegar libremente por el sistema por unos minutos.

Para esta segunda parte se utilizó la Escala de Usabilidad del Sistema, en inglés *System Usability Scale*[2], por su sigla SUS, que proporciona una herramienta confiable y rápida para medir la usabilidad de un sistema. Consiste en un cuestionario de 10 ítems con cinco opciones de respuesta para los encuestados; de **Totalmente en desacuerdo** a **Totalmente de acuerdo**. A continuación se presentan las preguntas de esta herramienta:

1. Creo que me gustaría usar este sistema con frecuencia
2. Encontré el sistema innecesariamente complejo
3. Me pareció que el sistema era fácil de usar
4. Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema
5. Encontré que las diversas funciones en este sistema estaban bien integradas
6. Pensé que había demasiada inconsistencia en este sistema
7. Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápido
8. Encontré el sistema muy engorroso de usar
9. Me sentí muy confiado usando el sistema
10. Necesitaba aprender muchas cosas antes de poder utilizar este sistema

[2] `https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html`

5.4.1. Resultados

El cuestionario recibió 14 respuestas en total. En la tabla 5.2 se puede apreciar la distribución de las respuestas de las preguntas, donde R1, R2, R3, R4 y R5 corresponden a las respuestas a cada una de estas preguntas. Respectivamente serían: "totalmente en desacuerdo", "en desacuerdo", "neutral", "de acuerdo" y "totalmente de acuerdo".

Preguntas	R1	R2	R3	R4	R5
P1	1	0	3	5	5
P2	5	8	1	0	0
P3	0	0	0	8	6
P4	11	2	1	0	0
P5	0	2	5	1	6
P6	9	3	2	0	0
P7	0	1	2	2	9
P8	6	7	1	0	0
P9	0	0	2	4	8
P10	9	4	1	0	0

Tabla 5.2: Distribución de resultados de cuestionario SUS

En general para la primera parte del cuestionario las respuestas fueron bastante positivas. Las primeras que se recibieron estaban enfocadas principalmente a la experiencia de usuario del sistema ya que por ejemplo para poder realizar búsquedas de tipos georreferenciables, se debía seleccionar un ítem de las sugerencias del autocompletado, esto al parecer era algo no intuitivo y mucha gente se impacientaba ya que la página solo cargaba y no mostraba resultados. Gracias a esto se realizó una actualización del sistema para manejar este error y así los siguientes usuarios no sufrieran lo mismo. También se comentó bastante la disparidad de resultados que existían entre continentes, ya que los resultados de algunas búsquedas obtenían muchos resultados en Europa por ejemplo y no en Latinoamérica, como era quizás esperado. Este es un problema que se escapa de las manos de este proyecto ya que esto depende única y exclusivamente de los datos registrados en Wikidata, es decir que si no aparecen por ejemplo algunas organizaciones educacionales (tarea 4) en Latinoamérica pero si en Europa, significa que ha existido gente o máquinas que han ingresado aquellos datos en Wikidata, lo cual no significa que no hayan en Latinoamérica. Esto es un tema no trivial de entender para gente que no conoce las bases de datos de la Web. También otro comentario que se repitió bastante fue la falta de información para algunas entidades, por ejemplo si se busca la etiqueta *stadium* la gente esperaba que en los resultados aparecieran algunas propiedades del mismo, como por ejemplo: su capacidad máxima, año de fundación, arquitecto, entre otras. Esto es sin duda algo muy desafiante ya que supondría que para cada uno de los más de 21 tipos georreferenciables que se encontraron, existiera algún tipo de ranking de propiedades que sirviera para obtener aquellos más populares, sin embargo en la actualidad Wikidata no posee algo por el estilo. Una solución sería para cada tipo hacer una consulta única para obtener aquellas propiedades, sin embargo esto es casi imposible ya que se cuentan con muchísimos y no sería algo eficiente.

Algunas sugerencias de actualización del sistema también fueron propuestas por parte de los usuarios. Junto con presentar mayor información para tipo de entidad georreferenciable, se propuso que el sistema esté disponible en otros idiomas, en español por ejemplo. Esto es algo que se ve factible en el futuro ya que una de las características más importantes de Wikidata es que es multilingüe. También otra actualización útil que fue comentada de parte de un usuario fue la de parametrizar la URL con el identificador único del tipo de entidad que se busca, esto para poder compartir mucho más fácil algunos resultados que entrega el sistema y poder acceder a ellos directamente, evitando la búsqueda del usuario.

En definitiva, los comentarios que se recibieron para las tareas encomendadas al usuario fueron muy positivos y sirvieron para realizar actualizaciones importantes al sistema que fueron muy fáciles de implementar, hubo una opinión general de una fluida navegación, una rapidez en la obtención de resultados y un futuro potencial que tendría la aplicación para diferentes contextos, como por ejemplo: uso de material complementario en clases de Historia y Geografía, investigaciones sociales y políticas o ejercicios de memoria que utilicen como referencia los resultados obtenidos para una búsqueda cualquiera.

Por otro lado, la encuesta SUS tuvo una positiva evaluación igualmente. En la tabla 5.2 se puede ver la distribución de las puntuaciones para cada pregunta, de la cual se puede inferir que más de la mitad de los encuestados piensa que le gustaría usar el sistema con frecuencia (Pregunta 1). De igual forma se concluye que los usuarios encontraron que el sistema no era complejo de usar y aprenderían por si solos a usarlo (Preguntas 2, 3, 4, 7 y 8). Se infiere de igual forma que los usuarios encontraron que el sistema era robusto (Pregunta 5 y 6).

Con los puntajes obtenidos de las respuestas de la segunda parte del cuestionario, es decir, con la encuesta SUS, se puede calcular un puntaje general del sistema que va a servir para interpretar los resultados. El cálculo de este puntaje corresponde a 83,4, por lo tanto, se puede decir que el sistema está por sobre el promedio, que es 68, esto quiere decir que el sistema cumple con tener una característica de usabilidad alta.

Se finaliza entonces el capítulo de Evaluación, en donde se vieron los distintos experimentos relacionados principalmente al rendimiento del preprocesamiento y análisis del *dump truthy* de Wikidata, así como también, de su contenido. Además se evaluó el rendimiento que tienen las distintas funciones de la API, como la búsqueda y obtención de las entidades de tipos georreferenciables para su posterior visualización en el mapa y las búsquedas de los tipos georreferenciables para la implementación del autocompletado del sistema. Por último se realizó una evaluación de la usabilidad del sistema en usuarios, para lo cual se creo una encuesta con algunas tareas a ejecutar en el sistema junto con un cuestionario SUS para generar un puntaje de usabilidad el cual fue calculado, obteniendo un resultado alto en usabilidad del sistema. En general los resultados y comentarios fueron bastante positivos y van a servir para en un futuro desarrollar nuevas funcionalidades que hagan que el sistema sea mucho más atractivo y amigable para el usuario. En el siguiente capitulo se verán las conclusiones del trabajo realizado.

Capítulo 6

Conclusiones

A lo largo de este trabajo se abordaron los temas relacionados al desarrollo de un sistema que fuera capaz de geolocalizar instancias de tipos de entidades georreferenciables de Wikidata, tales como estadios, museos arqueológicos, montañas, centros de tortura, organizaciones comerciales, lugares históricos LGBT, entre muchos más. Para conseguir esto, en este proyecto se abordó principalmente el método de acceso vía *dump truthy* para acceder a los datos que provee Wikidata. Con este *dump* lo que se logró fue hacer un preprocesamiento y posterior análisis para la obtención de los tipos de entidades georreferenciables en formato JSON. Paralelamente se trabajó en una consulta SPARQL para la obtención de las instancias de estos tipos comentados anteriormente. Estas partes del trabajo fueron muy importantes a la hora de implementar el autocompletado y la visualización de las entidades en un mapa mundial.

Para la arquitectura del sistema se pensó en el clásico modelo de cliente-servidor. Para ello se implementó una API usando Flask de Python que va a funcionar en el *backend* (servidor), que usará además los servicios de MediaWiki API y WDQS. Por otra parte, se implementó un *frontend* utilizando React.js y Bootstrap para la interfaz del sistema. Estas dos partes de la aplicación se comunican mediante HTTP, mediante peticiones GET a la API de parte del cliente al servidor para la obtención de datos.

En la API se implementó la función de autocompletado, la cual depende del JSON de los tipos generado en el preprocesamiento y análisis previo que se hace de los datos en el servidor. También se implementó la obtención de las entidades que son instancia de algún tipo georreferenciable, para ello se utilizó el *endpoint* de WDQS. Una vez implementadas estas funcionalidades en el *backend*, la aplicación es capaz de realizar la comunicación con el servidor, es decir, es capaz de realizar las búsquedas (con o sin límite) de los tipos georreferenciables otorgando sugerencias de autocompletado al usuario. Una vez que se obtengan resultados, la aplicación es capaz de generar una visualización de las entidades en un mapa mundial gracias a las coordenadas geográficas (latitud y longitud) que se obtienen de Wikidata.

Los resultados arrojados de la evaluación del sistema fueron bastante positivos. En la evaluación del rendimiento del análisis del dump de Wikidata se concluyó que la mejor opción para realizar estas tareas era la de utilizar el *dump* en formato gzip en vez de bzip2,

debido al tiempo que demora en ejecutarse, que es 3 veces más rápido que bzip2. Por otra parte en cuanto al contenido del *dump* se logró concluir que no existe mucha diferencia en cuanto al contenido de *dumps* correspondientes a distintos meses, en relación a nuevos tipos que se añadan a versiones más actualizadas. Esto no sería un problema ya que los tipos que más entidades tienen, son difíciles que desaparezcan en una actualización y los nuevos por lo general tienen muy pocas entidades georreferenciables.

Ahora, en cuanto al rendimiento que tiene la obtención de datos desde la API del sistema, se puede concluir positivamente que tanto para el autocompletado de tipos georreferenciables como para la búsqueda de entidades de tipos los tiempos de obtención de resultados se concentran en tiempos mínimos de espera. Existe una excepción en el caso de las búsquedas de entidades, ya que hay algunos tipos de entidades, principalmente aquellos geográficos como montañas, ríos, lagos; o de edificios como hospitales, museos, colegios; que demoran más de lo esperado. Esto es debido a un error que se propaga desde WDQS para este tipo de entidades, razón por la cual se implementó el selector de límite en el sistema.

La percepción que tuvieron los usuarios acerca del sistema fue en su mayoría positiva, gracias a algunos comentarios se pudieron realizar algunas actualizaciones simples al sistema para mejorar su navegación. En general los comentarios fueron de que el sistema entregaba resultados muy rápidamente, era de navegación fácil y sencillo de usar. Muchos de los comentarios servirán como base para realizar nuevas actualizaciones al sistema para hacerlo más robusto, para que sea más atractivo y para que tenga más alcance entre la gente.

En relación a los objetivos específicos del proyecto, se puede decir que se cumplen a cabalidad la mayoría de ellos, ya que, se logró obtener los datos relacionados de los tipos de entidades georreferenciables de Wikidata, se logró obtener información relevante de las entidades georreferenciables, las cuales pudieron ser visualizadas en un mapa mundial; además, se logró diseñar e implementar una interfaz intuitiva que permite realizar búsquedas con autocompletado y visualizar los resultados en el mapa mundial de una manera atractiva y llamativa. Por otra parte, y muy clave para el proyecto, se obtuvieron buenos resultados en términos de eficiencia cuando los resultados de las entidades obtienen muchos resultados, sin embargo esto es algo que solo tiene un buen rendimiento en el *backend* del proyecto, en concreto en la API; ya que, cuando se quiere evaluar el rendimiento de la visualización de instancias de tipos con muchas entidades georreferenciables, el sistema se tarda mucho en responder o simplemente no responde. La solución para esto ya se comentó anteriormente y fue la de implementar un selector de límite. Por último se logró hacer validaciones de la interfaz y percepción de usuarios reales con la aplicación y los resultados fueron en su mayoría positivos; sin embargo se hubiese esperado que como mínimo respondieran la encuesta al menos 20 personas. A pesar de esto de igual forma se pueden sacar buenos réditos del trabajo que se hizo gracias a la cantidad de comentarios positivos y pocos negativos que se obtuvieron.

Algunas de las desventajas que tiene el sistema es que no se le puede sacar el máximo de su potencial por una barrera existente en el *frontend* de la página, debido al mal rendimiento que tiene la generación de los marcadores y/o *clusters* cuando existen muchas instancias de un tipo. Otra desventaja que tiene, es que como está disponible solo en ingles, hay muchos usuarios que ven esto como una limitante para el uso del sistema. Lo bueno que tiene el sistema es que es capaz de obtener más de 21 mil tipos de entidades georreferenciables de

una forma confiable gracias al preprocesamiento que se hace de la base de datos de Wikidata, lo malo de esto es que aún no se logra dar con un buen filtro para el porcentaje de entidades que son georreferenciables de un tipo, ya que muchas búsquedas de instancias entregan pocos resultados o aveces ninguno y de todas formas aparecer en la barra de autocompletado. Otro aspecto positivo que se ve opacado, es la información que se obtiene para cada tipo de entidad. Al tratarse de un atlas, lo que se esperaría es que para cada tipo se obtenga más información que su descripción, imagen o ubicación geográfica; ya que, navegar por la interfaz de Wikidata para una persona que no conoce mucho de bases de datos de la web puede ser poco intuitivo y difícil de entender. Por último, lo mejor que tiene el sistema son los rendimientos que entrega la API a la hora de obtener información sobre entidades o tipos, estos son muy despreciables en su mayoría y hacen que la experiencia de usuario sea muy positiva en particular al utilizar la barra de búsqueda, no así en la visualización del mapa. Es por esto que se debe enfocar el esfuerzo del trabajo futuro en la generación de un mapa que tenga buen rendimiento a la hora de generar marcadores y *clusters* en el mismo cuando existan muchas instancia de un tipo.

A continuación se nombraran algunas de las tareas que se creen son las más importantes de realizar a futuro para mantener y mejorar el buen funcionamiento de la aplicación, tanto en el *backend* como en el *frontend*; también, para que mejore la experiencia del usuario, y para que siga buenas metodologías de diseño que ayuden a mejorar el rendimiento y apoyen a la escalabilidad del sistema.

Tareas para realizar a futuro:

- Servidor: agregar archivo de configuraciones del sistema; implementación y ejecución de *script* para descargar el *dump* de Wikidata actualizado; generar un nuevo archivo JSON de tipos.
- Frontend: mejorar el manejo de errores al momento de hacer *fetch* a la API; mejorar UX/UI; separar las distintas partes del *Home* en componentes de React.js, buscar una manera de generar los marcadores del mapa de una forma más eficiente cuando estos son muchos o dar un límite de tiempo de ejecución de consulta a la API, quizás 30 segundos y mostrar un mensaje de error para que el usuario pruebe nuevamente pero seleccionando un límite cuando esto ocurra.
- Backend: solucionar *bug* de entidades que aparecen como sugerencia de autocompletado y no tienen entidades georreferenciables; mejorar lógica y estructura de respuestas JSON de la API.
- Funcionalidades: agregar opción de seleccionar una región como un continente, país o ciudad del mapa para realizar la visualización, por ejemplo seleccionar la ciudad de Santiago y buscar todas las instancias de *school*; agregar opción para cambiar el idioma del sistema a español (ver si es escalable a otros idiomas), parametrizar la URL de la página con el identificador único de una entidad para poder compartir los resultados para cada tipo que entrega el sistema; ver si es factible para cada tipo de entidad obtener las propiedades más importantes, como por ejemplo de una montaña obtener su elevación sobre el nivel del mar (P2044) o de un estadio su capacidad máxima (P1083). El último ítem es algo que le daría mucho valor a la aplicación y se acercaría

mucho más a la idea de atlas que supone el proyecto ya que la interfaz de los perfiles de las entidades de Wikidata no es tan intuitiva para obtener la información básica de una entidad cualquiera. Esto sin embargo es una tarea muy desafiante por lo que se priorizaran las demás primero y luego se realizará una evaluación de esta última para ver si es factible de implementar.

Bibliografía

[1] Denny Vrandecic and Markus Krötzsch. *Wikidata: a free collaborative knowledgebase. Commun. ACM*, 57(10):78–85, 2014.

[2] Kim, Hong-Gee. Semantic Web. *Recuperado de http://semanticweb. org/wiki/Main_Page. html*, 2003.

[3] Hogan, Aidan. Web of data. In *The Web of Data*. Springer, 2020.

[4] Internationalized Resource Identifiers (IRIs). `https://www.ietf.org/rfc/rfc3987.txt`.

[5] Barrett, Daniel J. *MediaWiki: Wikipedia and beyond*. .º'Reilly Media, Inc.

, 2008.

[6] Elasticsearch, BV. Elasticsearch. *Internet: https://www. elastic. co/pt/,[Sep. 12, 2019]*, 2018.

[7] Claus Stadler and Jens Lehmann and Konrad Höffner and Sören Auer. LinkedGeoData: A core for a web of spatial open data. *Semantic Web*, 3(4):333–354, 2012.

[8] Felipe Nicolás Manen Núñez. Poniendo las ciencias de la computación en el mapa: desarrollando un sistema para geolocalizar investigación. 2021.

[9] Pérez, Jorge and Arenas, Marcelo and Gutierrez, Claudio. *Semantics and complexity of SPARQL. ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.