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

# SISTEMA PARA BUSCAR Y VISUALIZAR EXTRACTOS DE CONSTITUCIONES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

FRANCISCO JAVIER LECAROS DEL PRADO

PROFESOR GUÍA:
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
CLAUDIO DOMINGO GUTIÉRREZ GALLARDO
JOSÉ MIGUEL PIQUER GARDNER

SANTIAGO DE CHILE
2022

# Resumen

Debido al estallido social del año 2019, el plebiscito del año 2020 y la redacción de una nueva constitución ha aumentado de gran manera el interés de la gente de Chile sobre que es una constitución, que es lo que esta contiene y como su contenido puede afectarla directamente. Además, con la redacción de una nueva constitución ha aumentado el debate sobre los contenidos que esta debería tener y eso conlleva a querer saber los contenidos de las constituciones de otros países para comparar.

Es por eso que se plantea la idea de crear una aplicación web que permita buscar extractos de constituciones de distintos países que traten algún tema en especifico y poder compararlos. Esto debe poder hacerse de manera fácil y rápida.

Para implementar esta idea se utilizaron los datos de ofrecidos por *Constitute*, una ong que se dedica a dejar disponibles las constituciones de la mayoría de los países del mundo en un solo lugar. *Constitute* traduce las constituciones a distintos idiomas para que estas puedan ser leídas por mas gente y además clasifica los artículos de todas las constituciones según el tópico que tratan. Un formato en el que se encuentran disponibles todos estos datos es el formato RDF.

Para la implementación de esta aplicación se utilizo Fuseki para el *back-end* y Vue.js para el *front-end*. Fuseki es un servidor para hacer consultas SPARQL a datos en formato RDF y es ofrecido por la Apache Software Fundation. Vue.js es un *frame-work* para crear interfaces web.

La aplicación desarrollada permite elegir un tópico y dos países. Luego, esta aplicación muestra los extractos de cada constitución de los países elegidos que traten el tópico seleccionado. De esta forma es fácil comparar como distintos países abordan un mismo tema. Durante el desarrollo de la aplicación surgieron problemas que atrasaron el trabajo.

Se cumplió el objetivo de que fuera simple y fácil de usar. A pesar de lo anterior existen fallas que empeoran la experiencia del usuario. Un ejemplo de estas fallas es que en el selector de países se muestran países que no tiene disponible su constitución en español por lo cual los resultado sobre ese país siempre son vacíos.

Se concluye que se cumplió con la mayoría de los objetivos planteados para este trabajo y que esta es una buena primera versión de la aplicación pero que es necesario hacer múltiples mejoras para que esta sea atractiva para los usuarios.

*Dedicado a todas aquellas personas que me vieron sufrir algún ramo, me tuvieron fe y me vieron salir adelante contra todo pronóstico.*

# Agradecimientos

A mi profesor guía, Aidan Hogan, por ayudarme a sacar este trabajo adelante y por su gran disposición pese a mis múltiples atrasos.

A mi familia por siempre estar ahí para mí, al samy por siempre traerlee alegría a mi vida y en especial a mi papa y mi mama por nunca presionarme y por decirme que no me preocupara ya que sin importar el resultado siempre me seguirían apoyando.

A los cabros del pasillo, los frens y los ofisalites; 3 grupos de amigos que llegaron en distintas etapas de mi vida universitaria y que sin ellos no habría sacado esto adelante.

Al seba y a la vale por su amistad. En algún momento creí que no iba a pillar amigos así de cercanos, me alegro haberme equivocado.

Y finalmente, gracias a la vida por hacerme de la católica.

# Tabla de Contenido

# Índice de Ilustraciones

# Capítulo 1

# Introducción

## 1.1. Contexto

Existen múltiples definiciones de la palabra "constitución", pero todas comparten la misma idea general. Una constitución es la ley fundamental de un estado que regula la organización y funcionamiento de los poderes públicos, limitando el poder político y garantizando derechos y libertades a las personas [17]. Además, la constitución condiciona la creación del resto de las normas jurídicas de un estado como, por ejemplo, las leyes, los decretos y los tratados internacionales a los que este adhiere.

Al igual que la mayoría de los países, Chile se guía por una constitución, sin embargo, la creación de esta fue bajo condiciones no democráticas. Actualmente en Chile está vigente la "Constitución Política De La República De Chile" la cual fue promulgada en octubre del año 1980 [3] y vino a reemplazar la constitución que regía desde el año 1925. Dicho texto fue redactado en gran parte por la "Comisión de Estudios de la nueva Constitución Política del Estado", establecida por la Junta Militar de Gobierno el año 1973. El plebiscito de septiembre del 1980 aprobó dicha constitución con un 67,71 % de los votos a favor. Todo lo anterior bajo el contexto de la dictadura que se vivía en el país bajo el mando de Augusto Pinochet.

A pesar de los múltiples cambios y reformas que se le han hecho a la constitución chilena desde que se acabó la dictadura se ha cuestionado su legitimidad dadas las condiciones en las que fue creada y en las cuales se llevó a cabo el plebiscito que la aprobó. Esta ilegitimidad llevó a que se intentara crear una nueva carta magna, llegando incluso a tener un borrador el año 2018. Aunque finalmente dicho proceso no llegó a buen puerto y fue desechado.

Al siguiente año Chile vivió una de sus revoluciones sociales más masivas e importantes de su historia moderna. En octubre del 2019 el país entró en un periodo de inestabilidad política y social producto de múltiples manifestaciones que se extendieron por un mes [8]. Durante este mes la idea de una nueva constitución volvió a tomar fuerza y se volvió la demanda más recurrente en las distintas protestas. Finalmente, el 15 de noviembre, el gobierno de Chile, junto con varios líderes de partidos de todo el espectro político, firmaron un acuerdo para llamar a un plebiscito el mes de abril del 2020 en el cual se harían dos consultas:

- **¿Quiere usted una nueva Constitución?** Las opciones eran "Apruebo" o "Rechazo"
- **¿Qué tipo de órgano debiera redactar la nueva Constitución?** Las opciones eran "Convención mixta constitucional" o "Convención constitucional".

Producto de la pandemia del año 2020 que afectaba al planeta entero, el plebiscito se realizó en octubre del 2020. En la primera consulta ganó la opción "Apruebo" con un 78,27 % de los votos. En la segunda consulta ganó la opción "Convención Constitucional" con un 78,99 %. Con estos resultados se dio inicio al proceso para crear una nueva constitución mediante el mecanismo elegido.

La convención constitucional es el órgano que estará a cargo de redactar la nueva constitución. Esta convención está integrada por 155 miembros, llamados convencionales constituyentes. Estos integrantes fueron elegidos en las elecciones del 15 y 16 de mayo del 2021. Para la conformación de este órgano se aseguró la paridad de genero y la destinación de cupos especiales para pueblos originarios [9].

Una característica importante de la convención constitucional es que para poder postular a ella solo era necesario tener la ciudadanía chilena y tener 18 años o más [4]. Esta regla permitió que más gente tuviera la posibilidad de formar parte de este proceso. Pero una consecuencia de esta regla que puede no ser evidente es que no todos los constituyentes electos tienen el mismo nivel de conocimiento respecto a los temas a tratar. Esto se debe a que algunas personas tuvieron mayor acceso a educación que otras a lo largo de sus vidas.

Este último punto es un problema que no solo aqueja a los constituyentes. La gente de Chile está cada vez más interesada sobre los temas que se tratan en una constitución, entender como una constitución afecta el desarrollo de un país y comparar nuestra constitución a la de otros países.

## 1.2. Problema

Dado el contexto actual por el que atraviesa Chile actualmente surge la necesidad de democratizar la información. Actualmente se puede acceder de manera fácil a la constitución de Chile, ya sea mediante internet o comprándolo en forma física. El problema surge cuando se trata de analizar dicho texto. Alguien que no tenga experiencia puede encontrar complejo encontrar artículos que traten un tema en específico o entender los mismos artículos. Otro problema es que al redactar una nueva constitución es inevitable querer buscar constituciones de otros países para saber cuáles posturas respecto a ciertos temas han funcionado y cuales no pero acceder a estas constituciones se vuelve mas difícil y no existe una buena forma de comparar textos.

## 1.3. Solución

### 1.3.1. Idea General

Para solucionar los problemas mencionados en la sección anterior de este informe se necesita una solución que cumpla las siguientes condiciones:

- De fácil acceso a todos los usuarios y sea simple de utilizar.
- Que provea información sobre la constitución de Chile de manera clara y concisa.
- Que ayude en la búsqueda de artículos que traten temas específicos.
- Que permita comparar la constitución de Chile con la de otros países.

Una solución que cumpla estas condiciones no solo ayudaría a gente que trabaja redactando una nueva constitución si no que también podría servirle a gente que estudia o trabaja en derecho constitucional. Pero mas importante, una herramienta de este estilo le daría la opción a la gente de informarse sobre la carta magna que rige Chile. Lo anterior permitiría que la gente pueda tener una opinión informada y fomentaría el debate, no solo sobre la redacción de la nueva constitución, sino que también sobre cómo cambiarían las leyes chilenas en caso de ser aprobada.

### 1.3.2. Solución formal

Como solución se desarrolló una herramienta que permite acceder a las constituciones de múltiples países. Para la creación de esta herramienta se tuvo en cuenta que su principal objetivo era poder comparar dos constituciones, lado a lado. Con eso en mente la herramienta en si permite seleccionar:

- Un tópico, como por ejemplo, “Medios de Comunicación” o “Política Fiscal”.
- Hasta dos países.

El resultado que se muestra son los artículos de cada país seleccionado (uno al lado del otro) que tratan el tópico seleccionado.

La plataforma también permite hacer una búsqueda de artículos que contenga una palabra clave ingresada por el usuario, en caso de que la lista de tópicos no contenga un tema en específico que el usuario busca.

Para llevar a cabo esta solución se usaran los datos ofrecidos por *Constitute*, una ONG de la que se hablará más adelante.

## 1.4. Objetivos

Considerando la solución descrita, se identifica la necesidad de crear un software que permita comparar constituciones de forma fácil y rápida. A continuación se especificarán los objetivos de este trabajo de memoria.

### 1.4.1. Objetivo General

El objetivo de este trabajo es desarrollar una aplicación web que permita a cualquier persona buscar en la constitución chilena o en la de otros países artículos que traten algún tema de interés. Esta aplicación debe ser desarrollada teniendo en mente los temas más importantes en la contingencia chilena. Además, esta aplicación deber ser fácil y rápida de usar.

### 1.4.2. Objetivos Específicos

1. Descargar los datos de cada constitución desde *Constitute* en formato RDF. Luego juntar estos datos en un solo archivo para poder trabajar con ellos de forma directa.
2. Idear consultas que entreguen información útil y relevante para el proceso constituyente chileno.
3. Implementar un *back-end* usando los datos descargados desde *Constitute*. El *back-end* debe poder hacer consultas a los datos que están en formato RDF.
4. Diseñar e implementar una interfaz que permita hacerlas consultas mencionadas de forma directa, rápida y fácil.
5. Conectar el *back-end* con el *front-end* para poder probar la aplicación.
6. Realizar pruebas de usuario con potenciales usuarios.
7. Utilizar los resultados de las pruebas de usuario para idear mejoras que se puedan hacer a futuro.
8. Dejar disponible un servidor donde se guarde la aplicación, permitiendo que otras personas puedan acceder a ella mediante internet.

# Capítulo 2

# Marco Teórico

## 2.1. Soluciones existentes

Actualmente acceder a una constitución en específico no resulta difícil gracias al avance que ha tenido el internet en los últimos años. Pero cuando se desea tener acceso de forma rápida a múltiples constituciones y además se desea hacer comparaciones entre estas, se pueden generar varios problemas. El primero es que a pesar de que encontrar una constitución en específico puede ser fácil usando un motor de búsqueda, buscar múltiples constituciones se vuelve un trabajo más repetitivo y engorroso. El segundo problema es que muchas constituciones pueden estar en un idioma que el lector no maneja lo cual imposibilita su lectura. Otro problema que puede surgir es que buscar secciones que traten un tema en específico en una constitución puede ser un proceso lento si es que el índice no es claro o si el lector no ha leído este tipo de textos con anterioridad. Todos estos problemas abarcan solo el caso en que se desea analizar una sola constitución. Si se desea comparar dos textos las dificultades solo aumentan.

Existen proyectos que tratan de solucionar en alguna medida los problemas antes presentados. Un ejemplo de esto es el catálogo de constituciones hispanoamericanas de la Biblioteca Virtual Miguel De Cervantes [2]. El problema es que este proyecto solo abarca una pequeña porción de los estados del mundo, teniendo solo constituciones de 21 países distintos. Además, este proyecto no soluciona el problema de poder navegar los textos de forma eficiente.

Una mejor solución a los problemas anteriores es la creada por *The Comparative Constitution Project (CPP)* [1]. Esta organización fue fundada el año 2005 por Tom Ginsburg, Zachary Elkins y James Melton en la Universidad De Illinois. Su objetivo es poner a disposición de quienes trabajan revisando o redactando constituciones información sobre las distintas constituciones del mundo.

El trabajo de la CPP es generar datos comprensibles sobre constituciones de todo el mundo. Dicho trabajo llevó a que el año 2013 lanzaran la plataforma "constituteproject.org" [13], cuyo nombre abreviado es *Constitute*. Esta plataforma da acceso a las constituciones regentes de casi todos los estados independientes del mundo además de ofrecer acceso a constituciones anteriores y borradores. La mayoría de las constituciones se encuentran o fueron traducidas al

inglés. Sumado a eso la plataforma dispone de alrededor de 50 constituciones en o traducidas al español.

Una de las características más importantes de la plataforma es que que los pasajes relevantes de las constituciones están clasificados por tema. Existen alrededor de 70 materias por las cuales buscar entre las cuales se encuentran "educación gratuita", "protección del medio ambiente" y "comisión de derechos humanos". Además se pueden filtrar las constituciones por región o se puede buscar un tema en un país en específico. De esta forma *Constitute* permite acceder a información muy específica de forma fácil y rápida. Además de la aplicación que ofrece, *Constitute* dispone una API para acceder a los datos disponibles. También se encuentran disponibles los datos en formato RDF para poder trabajar con ellos de forma directa. Esto permite que desarrolladores puedan generar sus propias aplicaciones.

Pero a pesar del gran trabajo que ha hecho *Constitute* centralizando las constituciones del mundo, su aplicación web tiene algunos problemas que dificultan su uso. Un ejemplo de estos problemas es que buscar por un tema en específico es menos directo que lo esperado. Al llegar a la *landing page* de la aplicación es necesario hacer click en el botón que dice "Materia". Eso lleva a la vista que se ve en la siguiente figura.

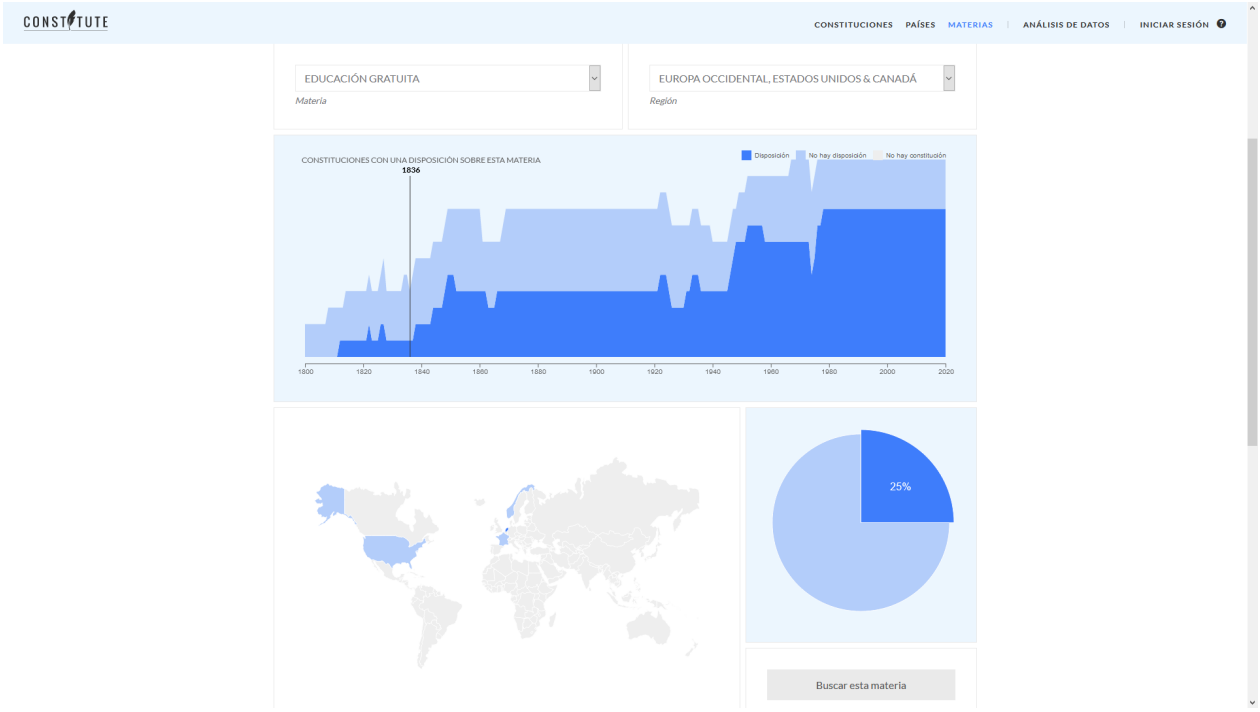


Figura 2.1: Vista para seleccionar una materia

En la figura se pueden ver dos selectores, uno para materias y otro para regiones. Pero al apretar el botón 'buscar esta materia' que se ve en la figura, el sistema busca la materia en todas las constituciones disponibles y muestra estas constituciones en forma de lista en una tercera vista. Esto es debido a que el selector de regiones solo sirve para destacar qué países de una región en específico tratan la materia seleccionada, como se ve en el mapa de la figura. El hecho de tener que pasar por tres vistas distintas para llegar a una lista de constituciones que contienen la materia seleccionada es un proceso que toma más tiempo que lo esperado y eso sin considerar que aún no se logra llegar a ver cómo un país en específico trata el tema.

A lo anterior se suma que la interfaz es poco intuitiva lo cual genera confusión con el selector de regiones siendo un claro ejemplo.

Estos son solo algunos de los problemas que hacen que la aplicación sea difícil de usar. Otros problemas que surgen son el hecho de que hay algunas visualizaciones que no se entienden y que el sistema para filtrar constituciones es muy limitado.

Un aspecto importante de la aplicación de *Constitute* es que esta se enfoca mucho en mostrar datos históricos de constituciones. Un ejemplo de esto es la visualización que se puede ver en la figura 2.1. Esta visualización muestra cuántos países tratan una materia en específico y cuántos no, dependiendo del año. Y a pesar de que este tipo de datos pueden ser de interés para otras áreas, estos no son de relevancia para gente que solo necesita informarse sobre las constituciones vigentes.

En consideración de todo lo anterior, la mejor opción para desarrollar una aplicación web que sirva para comparar constituciones del mundo es usar los datos que ofrece *Constitute*. Mas allá de la api que *Constitute* ofrece, el hecho de que se puedan descargar los datos en formato RDF abre muchas posibilidades. Pero para poder entender lo anterior es necesario tener una noción sobre cómo funcionan los datos en la web. Además para poder desarrollar una aplicación web es necesario saber las herramientas disponibles.

## 2.2. Tecnologías

Para el desarrollo de esta memoria se ocuparon múltiples tecnologías, tanto de *back-end* como de *front-end*. A continuación se describirán mas en profundidad cada una de estas.

### 2.2.1. Web Semantica

Según la *World Wide Web Consortium (W3C)*[1], la web semántica es una web de datos. En el día a día todos buscamos y usamos información que esté disponible en internet. Un aspecto importante que ha sido de gran beneficio para todos es el hecho de que esta información que está disponible en internet no está aislada, todo lo contrario, esta información está interconectada. Se pueden juntar datos de distintas fuentes para poder crear nuevos datos. Todo esto es gracias a la web de datos.

La web semántica se basa en dos ideas. La primera es que la información disponible en la web tenga un formato en común. Esto es importante para lo que se mencionó anteriormente, el poder combinar datos de distintas fuentes. El segundo aspecto de la web semántica es que se enfoca más en el significado de los datos que en la estructura de estos. Esto conlleva a que cómo se estructuran y se consultan estos datos sea distinto a cómo se haría con otros tipos de bases de datos.

Desde un punto de vista más técnico, la web semántica consiste principalmente de tres

[1] `https://www.w3.org/standards/semanticweb/`

tecnologías [7]:

- **RDF(Resource Description Framework)**
- **SPARQL**
- **OWL(Web Ontology Lenguage)**

Considerando el alcance del trabajo que se desea realizar, a continuación solo se describirán en detalle las dos primeras tecnologías mencionadas.

### 2.2.2. RDF

**RDF** o **Resource Description Framework** es un estándar de la W3C para describir recursos y el intercambio de datos. Los recursos son cualquier cualquier cosa que uno podría llegar a describir en forma de datos [15]. Ejemplos de recursos son:

- Entidades virtuales como páginas web.
- Entidades concretas como personas o ciudades.
- Entidades abstractas como categorías o puntos en el tiempo.

Tal como se mencionó antes, el objetivo de RDF es poder describir estos recursos y esto lo hace mediante tripletas de términos RDF. Estas tripletas se basan en una estructura del lenguaje natural muy común, sujeto-verbo-objeto. Tomemos la siguiente frase como ejemplo, *Santiago es la capital de Chile*

Esta frase sigue la estructura antes mencionada con '*Santiago*' como el sujeto, '*es la capital de*' como el verbo o acción y '*Chile*' el objeto. Una tripleta RDF sigue una estructura casi idéntica, sujeto-predicado-objeto. Entonces la frase anterior pasada a formato RDF se vería de la siguiente forma: *(ex:Santiago)(ex:es_la_capital_de)(ex:Chile)*. En esta declaración cada elemento entre paréntesis es un término RDF con *(ex:Santiago)* siendo el sujeto, *(ex:es_la_capital_de)* el predicado y *(ex:Chile)* el objeto.

Estas tripletas se pueden ver como un grafo dirigido, con el sujeto y el objeto siendo los nodos y el predicado siendo la arista, como se puede ver en la Figura 2.

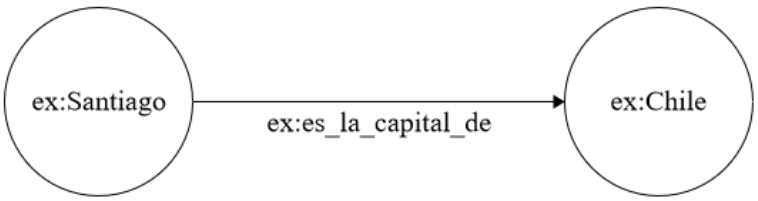


Figura 2.2: Una tripleta RDF como un grafo dirigido

Una de las principales razones para que los datos en la web sean como un grafo es que estos son muy flexibles y permiten agregar datos de manera muy fácil. Esta es una gran ventaja frente a otra forma de guardar datos como las tablas que se usan en las bases de datos relacionales. Para ejemplificar esta ventaja supongamos que estamos armando una base de datos sobre países con datos como el PIB, el tamaño en kilometros cuadrados y la población. Ahora supongamos que para cierto país no existe el dato del PIB. Si los datos están siendo guardados en una tabla esto implicaría que quedaría un campo vació. En cambio, si los datos estuvieran en formato RDF esto no importaría ya que no hay campos, simplemente no se agregaría la tripleta del PIB de ese país en especifico. Esto es muy importante ya que los datos en la web nunca estarán completos; para describir dos cosas del mismo tipo (países, ciudades, personas) siempre existe la posibilidad de que hayan distintos datos para describirlas. Si todos los datos de la web se guardaran en tablas, habrían millones de campos vacíos y navegar estos datos se volvería muy difícil. En cambio, el formato RDF ofrece un esquema que no es fijo, lo cual le permite adaptarse a estas situaciones.

Para que el formato RDF funcione es necesario tener una forma de identificar los recursos, evitando conflictos de nombres. Para esto, en general se usan las *IRI's*[12], una forma de identificar recursos en la web. Un ejemplo seria ***http://www.constituteproject.org/constitution/Afghanistan_1964/***. Este identificador proviene de los datos con los que se trabajó en esta memoria.

Considerando que es posible que un término RDF esté conectado a más de un otro término entonces es posible juntar múltiples tripletas RDF formando un grafo RDF. Un ejemplo de un grafo RDF se puede ver en la figura 2.3. Para la creación de este grafo se usó una pequeña muestra de los metadatos de *Constitute*, en específico, los metadatos de la constitución de Abzkhia y Chile.

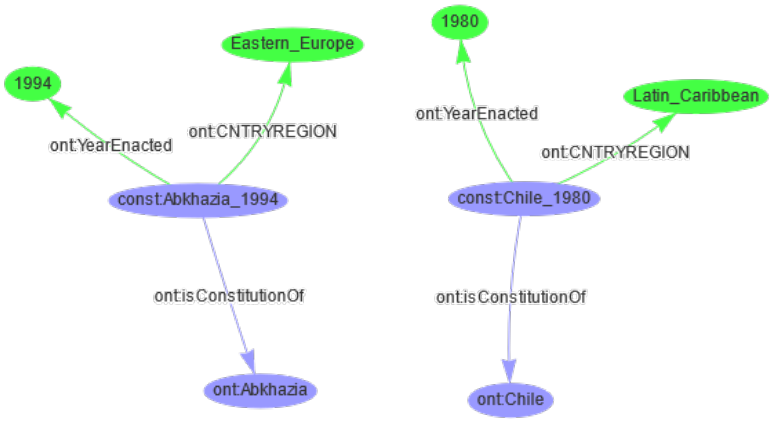


Figura 2.3: Grafo RDF basado en los metadatos de Constitute

Este grafo se generó usando prefijos para los identificadores. Cómo funcionan estos prefijos será explicado en la siguiente sección de este informe.

### 2.2.3. SPARQL

**SPARQL** o **SPARQL Protocol And RDF Query Language** es el lenguaje recomendado por la W3C para hacer consultas para RDF [15]. Uno de los aspectos más importantes de SPARQL es que permite hacer *federated queries*, esto es, poder hacer una consulta y que la solución esté basada en información proveniente de distintas fuentes. Estas distintas fuentes pueden ser, por ejemplo, distintas bases de datos en la web. Es esta habilidad la que permite poder juntar distintos tipos de información y poder generar nuevos datos.

Las consultas en SPARQL consisten en patrones que se comparan con los tripletas RDF de los datos. Las estructuras de estas consultas se asimilan mucho a las consultas SQL que se hacen cuando se trabaja con bases de datos relaciones. A continuación se describirán las principales características de SPARQL que se usaron para para hacer las consultas de esta memoria.

#### Consulta Básica

Teniendo datos en formato RDF se hace la siguiente consulta:

```
SELECT ?objeto
WHERE {
   ?sujeto ont:CNTRYREGION "Latin_Carribean" .
   ?sujeto ont:YearEnacted ?objecto .
}
```

Query 2.1: Consulta basica de SPARQL

Aquí se está consultando por el termino *?objeto* que seria todo termino que esté en la posición del objeto en una tripleta RDF. Si consultáramos el grafo RDF descrito en la sección anterior que se puede ver en la figura 2.2, el resultado sería *1980*. De esta manera se podría cambiar el termino a consultar, por ejemplo, cambiando *?objeto* por *?sujeto* en la primera línea de la consulta. Aquí el resultado usando el grafo RDF anterior sería *ont:Chile*. También se puede consultar por más de un termino.

#### SELECT DISTINCT

Al igual que SQL, SPARQL ofrece la posibilidad de asegurar que todos los resultados sean distintos agregando la palabra DISNTINCT a la consulta, de la siguiente forma:

```
SELECT DISTINCT ?objeto
WHERE {
    ?sujeto ?verbo ?objeto.
}
```

Query 2.2: Consulta con DISTINCT

### FILTER y CONTAINS

SPARQL también permite filtrar utilizando FILTER y el resultado puede depender de si cierto termino es mayor o menor a algún valor, si el termino es de algún tipo o si el termino en si contiene algún string en específico (solo funciona si el termino también es un string). Este ultimo caso se usó múltiples veces en el desarrollo de esta memoria y se hace mediante el uso de CONTAINS. Un ejemplo combinando de FILTER y CONTAINS se puede ver a continuación:

```
SELECT DISTINCT ?objeto
WHERE {
    ?sujeto ?verbo ?objeto.
    FILTER(CONTAINS(?objeto),"Chile").
}
```

Query 2.3: Consulta con FILTER y CONTAINS

Aquí se esta consultando por el termino *?objeto* pero solo se mostraran los resultados donde *?objeto* contenga el string *Çhile"*.

### ORDER BY y LIMIT

Como última herramienta que se usó en esta memoria, SPARQL también da la posibilidad de ordenar los resultados dependiendo del termino que se especifique y la cantidad de resultados que se desea. Esto se logra usando ORBER BY y LIMIT en la consulta, de la siguiente manera.

```
SELECT DISTINCT ?objeto
WHERE {
    ?sujeto ?verbo ?objeto.
}
ORDER BY ?objeto
LIMIT 10
```

Query 2.4: Consulta con ORBER BY y LIMIT

Aquí, SPARQL primero ordenara los resultados de forma ascendente y luego mostrara solo los 10 primeros.

### Prefijos

Dado que algunos identificadores pueden ser muy largos, SPARQL nos permite crear prefijos para poder hacer las consultas de manera mas fácil y hacerlas mas legibles. Por ejemplo, muchos identificadores de los datos de *Constitute* empiezan *https://www.constituteproject.org/constitution/ontology*. A continuación se puede ver una consulta SPARQL donde se ocupa uno de estos identificadores.

```
SELECT DISTINCT ?const ?country
WHERE {
  ?const
  <http://www.constituteproject.org/ontology/isConstitutionOf>
  ?country
}
```

Query 2.5: Consulta sin prefijos

En caso de que sea necesario hacer una consulta que ocupe múltiples identificadores que partan de la misma manera, escribir dicha consulta se puede volver engorroso. Es por eso que existen los prefijos, que sirven para simplificar las consultas. Ahora veremos la consulta anterior pero usando prefijos.

```
prefix constowl: <http://www.constituteproject.org/ontology/>
SELECT DISTINCT ?const ?country
WHERE {
  ?const constowl:isConstitutionOf ?country
}
```

Query 2.6: Consulta con prefijos

De esta manera las consultas se vuelven mas faciles de escribir y de leer.

Con esto se cubre lo básico que se usó para esta memoria. Las consultas SPARQL hechas para consultar los datos de *Constitute* serán descritas más adelante en este informe.

### 2.2.4. Apache Jena Fuseki

Apache Jena es un framework de Java para construir aplicaciones de Web Semantica[cita]. Este framework fue originalmente diseñado en HP Labs el año 2000 y fue adoptado por la Apache Software Foundation el año 2010. Apache Jena ofrece una gran cantidad de librerias Java para poder desarrollar aplicaciones que manejen datos en formato RDF, OWL y hagan consultas SPARQL.

Una de las herramientas que ofrece Apacche Jena es Fuseki, un servidor de SPARQL. Con Fuseki se pueden subir datos en formato RDF para después poder hacer consultas mediante peticiones HTTP. De esta manera Fuseki permite consultar los datos como si se tratara de una REST API. Fuseki puede ser usado de varias maneras pero para esta memoria se usó como un servidor que viene con una UI ya diseñada. Una gran ventaja de esta UI es que ofrece una interfaz para hacer consultas SPARQL de manera directa. Esto fue de gran utilidad para poder crear las consultas necesarias para esta memoria.

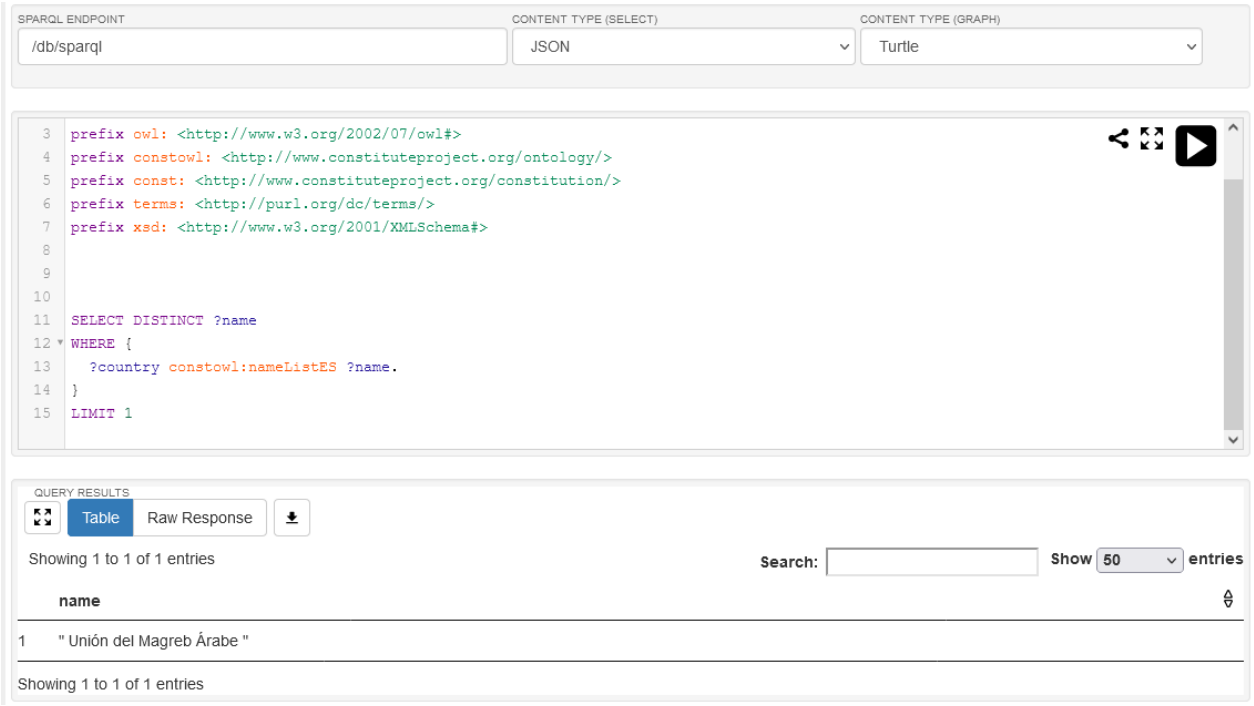


Figura 2.4: Interfaz para hacer consultas SPARQL de Jena

### 2.2.5. HTTP

HTTP o *Hypertext Transfer Protocol* es un protocolo para transmitir documentos. Es un protocolo basado en una estructura cliente-servidor y fue diseñado para lograr la comunicación entre clientes web y los servidores. Hoy en día este protocolo es parte fundamental de como nos comunicamos vía internet.

Una solicitud HTTP inicia con el cliente solicitando la información al servidor. Luego el servidor revisa la solicitud y envía la respuesta al cliente. Esta respuesta puede ser la información solicitada o algún código de error indicando que la solicitud no es valida. Esto se puede deber a un error en la consulta misma o que no se tienen los permisos para acceder a esa información.

Para esta memoria se realizaron peticiones HTTP desde el *front-end* creado en Vue.js al servidor de Jena que contiene los datos de *Constitute*.

### 2.2.6. Vue.js y Axios

**Vue.js** fue creado por Evan You en 2013. You basó su trabajo en Angular.js, *framework* con el que había trabajado durante su tiempo en Google. En una entrevista You declaró que su intención era extraer lo que más le gustaba de Angular y hacer algo realmente liviano [6]. Vue.js ha adquirido gran popularidad durante los últimos años a pesar de no ser apoyado por alguna gran empresa, como es el caso de React y Angular. Algunos aspectos que han ayudado a Vue.js a ganar terreno son su simplicidad, su capacidad de integración con otros

*frameworks* como React y su baja barrera de entrada ya que solo es necesario saber lo básico de *JavaScript*, *HTML* y *CSS* para poder usarlo. Entre las plataformas más famosas que han sido desarrolladas usando Vue.js se encuentran la página de Nintendo, GitLab y 9gag [5].

Desde un lado más técnico, la simplicidad se debe a que al igual que otros frameworks como Angular.js o React, Vue.js funciona con componentes. Esto permite crear distintos elementos de una página web por separado para luego ser juntados. La diferencia es que Vue.js funciona mediante *Single File Components*, un archivo con código que encapsula el HTML, CSS y JavaScript del componente creado. La principal característica de Vue.js es el *declarative rendering*: usando un *template* que extiende el HTML del componente es posible describir como debería verse el HTML en base al código en JavaScript. Si el código en JavaScript cambia, Vue.js se encarga de actualizar de manera automática como se debe ver el HTML. A continuacion se mostraran ejemplos de componentes en Vue.js. Este código proviene del tutorial oficial de Vue.js[2].

App.vue +

```
<template>
  <h1><h1>{{ message }}</h1></h1>
</template>


<script>
export default {
  data() {
    return {
      message: 'Hola Mundo'
    }
  }
}
</script>

```

Figura 2.5: Componente de Vue.js

Este componente al ser rendereado muestra un simple “Hola Mund”. Vue.js permite insertar JavaScript en el HTML mediante el uso de las dobles llaves; de esta manera actualiza de forma automática el HTML si es que cambiamos los datos guardados en script, por ejemplo, cambiando ”Hola Mundo“ por ”Hello World“.

Axios es un cliente HTTP basado en promesas para *Node.js*. Para esta memoria se utilizó Axios para hacer peticciones HTTP desde el *front-end* el servidor de Jena.

### 2.2.7. Vuetify

Vuetify es un UI framework creado sobre Vue.js[cita]. Vuetify ofrece una gran cantidad de competentes ya listos, los cuales fueron creados siguiendo las especificaciones de Material

[2] `https://vuejs.org/tutorial/#step-2`

Design (sistema de diseño creado por Google). Botones, calendarios, tablas y formularios son algunos de los componentes que ofrece Vuetify.

Una gran ventaja de Vuetify es que es muy fácil agregar componentes: basta con solo hacer la llamada correspondiente en el HTML. En la siguiente imagen se puede ver cómo se usa la componente *v-btn* para crear un botón en la parte inferior. También se puede elegir el color, la forma y múltiples otras opciones.

```vue
<template>
  <div class="text-center">
    <v-btn
      rounded
      color="primary"
      dark
    >
      Rounded Button
    </v-btn>
  </div>
</template>
```

VUE


Figura 2.6: Botón creado usando v-btn

# Capítulo 3

# Diseño de la Solución

Para esta sección del trabajo se planteó de forma concreta el problema a solucionar y se analizaron distintas formas de solucionarlo.

## 3.1. Planteamiento del Problema

Con el estadillo social del 2019, el plebiscito del 2020 y la redacción de una constitución ha aumentado el interés del pueblo chileno sobre que es lo que contiene una constitución y como podría afectarles en su vida. Como se ha mencionado con anterioridad en este informe, actualmente no existe una forma fácil para que la gente en Chile pueda informarse sobre la constitución que los rige. Además, para alguien que no ha leído la constitución con anterioridad le puede resultar difícil buscar artículos que traten algún tema de interés. Con la redacción de un nueva carta magna también surge la necesidad de comparar constituciones entre países para ver resultados de distintas formas de tratar un mismo tema. Esta comparación también es difícil de hacer con las herramientas disponibles actualmente.

## 3.2. Diseño de la solución

Para abordar este problema la prioridad fue que cualquier solución que se desarrollara tenia que ser simple y fácil de usar. Con esos en cuenta, la idea general para todas las alternativas ideadas fue siempre la misma: una aplicación web que permitiera comparar extractos de constituciones de distintos países que trataran un mismo tópico.

Para cumplir la condición de que esta aplicación web fuera simple y fácil de usar se decidió que esta aplicación debía tener una sola interfaz principal donde usuario pudiera hacer las consultas deseadas. De esta manera el usuario podría usar esta aplicación de forma inmediata, sin tener que hacer clicks innecesarios. En la siguiente figura se puede apreciar un mock-up hecho en las primeras etapas de este trabajo.

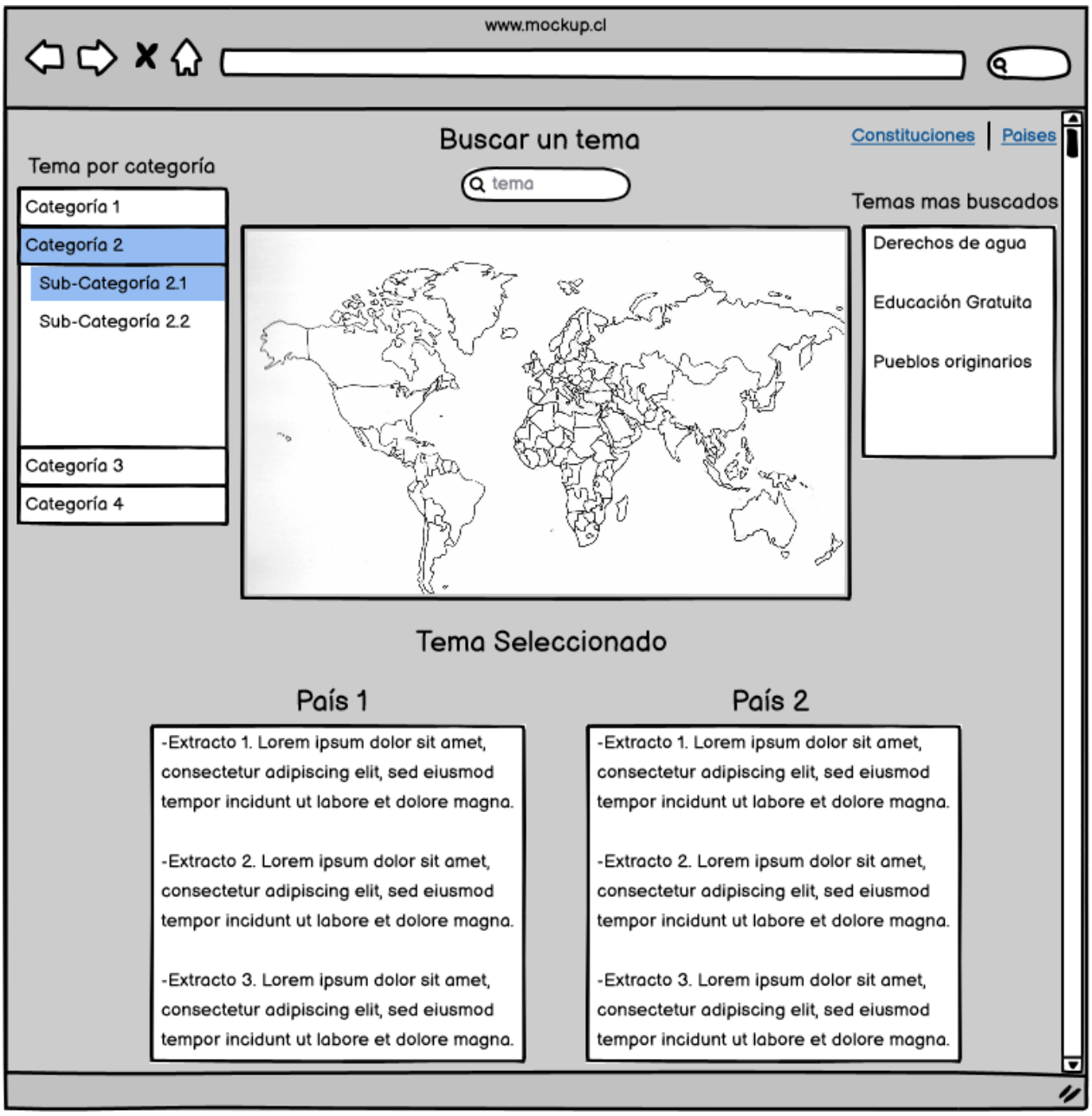


Figura 3.1: Mockup de la vista principal de la aplicación

La estructura principal de la aplicación seria un selector para el tópico que se desea buscar y dos selectores extras, uno por cada país. Los resultados de cada país se mostrarían abajo del selector respectivo. Con los selectores de los países y sus resultados respectivos uno al lado de otro, la comparación se haría de forma inmediata.

Al principio de este trabajo también se pensó en otras ideas como:

- Un mapa clickeable para seleccionar los países.
- Un ranking de los temas mas buscados.
- Un menu para los tópicos disponibles.
- Otras interfaces para visualizar constituciones completas.

Estas ideas no se llevaron a cabo, algunas por no aportar a lo que se buscaba, otras por la dificultad de implementarla en una primera versión y otras por falta de tiempo debido a problemas que surgieron durante el desarrollo de esta aplicación.

En el siguiente capítulo se describirá el proceso de desarrollo de esta aplicación, los problemas que surgieron durante el trabajo y el resultado final.

# Capítulo 4

# Implementación de la solución

En este capitulo se describirá cada paso de cómo se implementó la solución diseñada, desde cómo se consiguieron los datos hasta la finalización de esta primera versión de la aplicación. Durante esta descripción también se hablará sobre los problemas que surgieron, las decisiones de diseño que se hicieron y cómo estas llevaron al producto final.

## 4.1. Descarga de los datos

Como se ha mencionado múltiples veces en este informe, para esta memoria se utilizaron los datos que ofrece *Constitute* en su pagina web. El primer problema que surgió es que *Constitute* no ofrece una forma de descargar todas las constituciones juntas. En cambio, *Constitute* tiene una pagina para cada país de la cual se pueden descargar tanto la constitución actual del país como constituciones anteriores o borradores.


Figura 4.1: Interfaz para descargar la constitución de Chile

En la figura 4.1 se puede ver la interfaz que permite descargar la constitución de Chile

en múltiples formatos pero el que nos interesa para este trabajo es el formato RDF(NT). Al hacer click somos llevados a la URL `https://www.constituteproject.org/constitution/Chile_2021.nt` y la descarga comienza. Para cambiar la constitución de Chile que queremos descargar basta con apretar el botón abajo de *AVAILABLE FILES* como se puede ver en la siguiente figura.


Figura 4.2: Interfaz para descargar la constitución de Chile

Este proceso no es un problema si se desea descargar una constitución en específico pero si se desean descargar todas las constituciones disponibles se puede volver un proceso más largo y agotador considerando que hay 260 países disponibles.

Para solucionar este problema se decidió hacer un *script* para automatizar este proceso. Como se describió antes, para descargar una constitución existe una URL específica que es donde se encuentra. Esta URL tiene el mismo patrón para todas las constituciones de todos los países que seria: *`https://www.constituteproject.org/constitution/[id]`* donde el *id* representa la constitución que se desea descargar. Bastaba con hacer una petición a cada URL disponible y guardar los contenidos en un solo archivo. En la siguiente figura se muestra el *script* creado para llevar a cabo esta tarea. Este *script* fue programado usando *Python* y se usaron las librerias *requests* y *tqdm*. La primera librería permite hacer peticiones HTTP y la segunda sirve para mostrar el progreso en la consola.

```
1  import requests
2  import tqdm
3  r = requests.get('https://www.constituteproject.org/service/constitutions').json()
4  ids = [el['id'] for el in r]
5  baseurl = "https://constituteproject.org/constitution/{}.nt"
6
7  def get_constitution(id_):
8      req = requests.get(baseurl.format(id_)).text
9      return req
10
11 with open("db.nt", "w") as file:
12     for id_ in tqdm.tqdm(ids):
13         file.write(get_constitution(id_))
14
```

Figura 4.3: *Script* para descargar todas las constituciones

Para el *script* fueron necesarios los *id* de todas las constituciones, los cuales se consiguieron mediante la API que ofrece *Constitute*(linea 3 y 4 del *script*). Luego simplemente se hizo la petición para cada *id* usando la función *get_constitution* y guardando el resultado en el archivo *db.nt*. De esta manera todas las constituciones en formato RDF guardaron en un solo archivo.

Además de las constituciones, *Constitute* también ofrece un archivo de meta-datos y un archivo de la ontología de los datos. El contenido de estos archivos y su utilidad serán descritos mas adelante.

## 4.2. Servidor Fuseki

Como *back-end* de este proyecto se usó el servidor que ofrece Apache Jena llamado Fuseki. Como se mencionó en el marco teorico de este informe, Fuseki es un servidor de SPARQL que permite subir datos en formato RDF y hacer consultas mediante peticiones HTTP. Para este trabajo se decidió usar la versión de Fuseki que viene con una UI incluida que permite hacer consultas SPARQL de manera directa. Para poder usar Jena, primero es necesario bajar el distribución de la pagina de descargas de Apache Jena[10].

Una vez descargado el paquete basta con descomprimir el archivo *.zip* y ejecutar el archivo *fuseki-server.jar* que se encuentra dentro del paquete ya descomprimido. El servidor es lanzado de forma local en el puerto 3030. La *landing page* se puede ver en la siguiente figura.

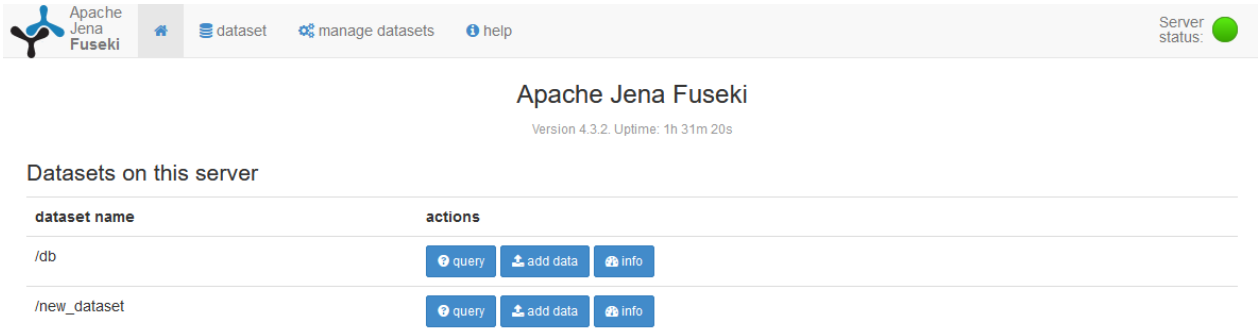


Figura 4.4: *Landing page* de Fuseki

Lo primero que salta a la vista es que Fuseki lista todos los *datasets* que han sido subidos al servidor y ofrece tres opciones:

- *Query*: este botón lleva a la interfaz que se mostró en el marco teórico de este informe.
- *Info*: aquí se pueden ver los servicios disponibles para el *dataset* seleccionado. Además se puede ver informacion del *dataset*, como por ejemplo la cantidad de tripletas RDF que tiene. En específico, el *dataset* de este proyecto consta de 2.504.085 tripletas.
- *Add data*: este botón lleva a la interfaz para subir archivos al *dataset*.

Por simplicidad, trabajaremos con un *dataset* ya creado llamado *new_dataset*. Ahora es necesario subir los archivos con los datos a este nuevo *dataset*. Esto se hace mediante la siguiente interfaz, a la que se llega con el botón *add data* que se ve en la figura 4.5.

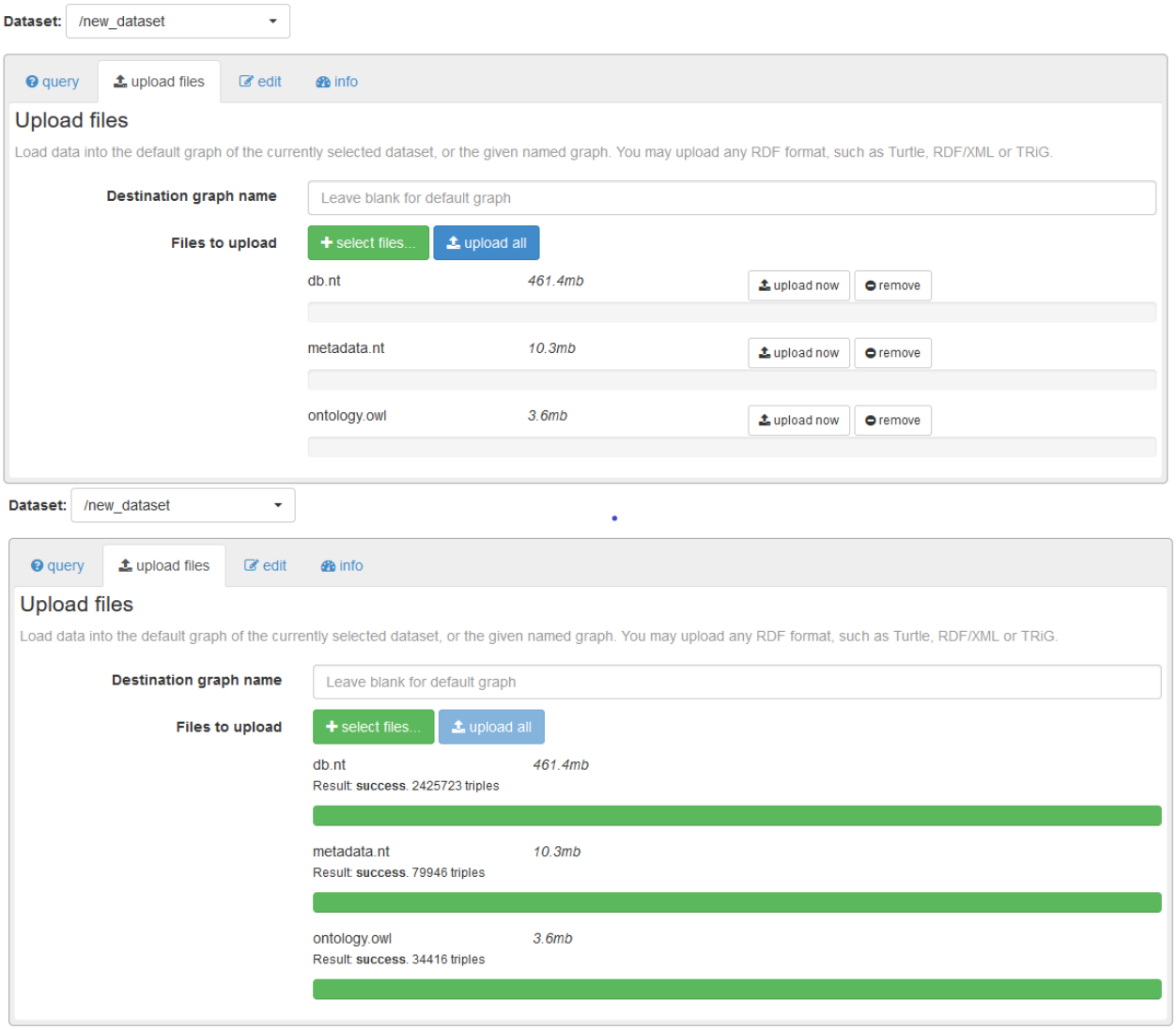


Figura 4.5: Interfaz para agregar datos al *dataset*.

Para subir los datos basta con seleccionar los archivos usando el botón *select files* y una vez selecciondos hay que hacer click en el botón *upload all* y esperar. Fuseki confirmara que los

archivos se subieron de manera correcta y mostrara la cantidad de tripletas RDF agregadas, tal como se ve en la figura 4.6. Al subir los datos se puede seleccionar el grafo de destino pero el concepto de múltiples grafos RDF escapa al trabajo de esta memoria por lo cual solo se ocupa un solo grafo, que en este caso se llama *default graph*.

En la práctica, *new_dataset* es idéntico a *db*, que es el *dataset* con el que se trabajó en el proyecto.

## 4.3. Ontología de los datos

Con el servidor ya arriba y los datos subidos, el siguiente paso en el proyecto fue familiarizarse con los datos de *Constitute* para poder hacer las consultas necesarias para llevar a acabo el proyecto. Para poder entender mejor los datos primero hablaremos de los tres archivos que se subieron a Fuseki: *db.nt*, *metadata.nt* y *ontology.owl*. Se describirá brevemente el tipo de datos que guardan para luego en la siguiente sección describir más en detalle algunos de los términos RDF que los componen, cuando se describan las primera consultas que se hicieron.

A partir de ahora se mostrarán múltiples consultas SPARQL, por lo cual a continuación se mostraran los prefijos que fueron creados para hacerlas. Estos prefijos son los mismos para todas las consultas por lo cual se mostrarán solo aquí.

```
PREFIX prefix: <http://prefix.cc/>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix constowl: <http://www.constituteproject.org/ontology/>
prefix const: <http://www.constituteproject.org/constitution/>
prefix terms: <http://purl.org/dc/terms/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
```

Query 4.1: Prefijos de las consultas

### db.nt

El archivo *db.nt* es el archivo que fue creado usando el *script* descrito en la primera parte de esta capitulo. Este archivo guarda el contenido mismo de las constituciones y borradores de todos los países. Esto quiere decir que solo contiene los artículos de las constituciones y algunas características de estos como el *id* del articulo o la constitución a la que pertenece.

### metadata.nt

El archivo *metadata.nt* contiene datos más detallados sobre cada constitución. Algunos datos que se pueden obtener son:

- El año que fueron creadas.
- El titulo oficial.
- El ultimo año en que fue actualizada

## ontology.owl

El archivo de *ontology.owl* es la ontología de los datos de *Cosntitute*. Una ontología es como un vocabulario de la web semántica y define conceptos y relaciones para describir un área de interés[14]. En especifico, en este archivo se definen las propiedades de los datos.

A continuación mostraremos como se define una propiedad usando este "lenguaje"pero ahondar mas en la descripción de OWL se escapa de la finalidad de este informe.


Figura 4.6: Definición de la propiedad *isConstitutionOf*

Como se ve en la figura 4.7, se define una propiedad: *isConstitutionOf*. Entonces un ejemplo de una consulta en SPARQL usando este termino seria el siguiente.

```
SELECT DISTINCT ?const ?country
WHERE {
  ?const constowl:isConstitutionOf ?country
}
LIMIT 1
```

Query 4.2: Consulta usando *isConstitutionOf*

El resultado de la consulta anterior es el siguiente:

| ?const | ?country |
|---|---|
| const:Abkhazia_1994 | constowl:Abkhazia |

El resultado significa que la constitución del 1994 de Abkhazia(nombre en ingles del país) pertenece al país Abkhazia.

De esta manera, el archivo *ontology.owl* define muchas propiedades, además de relaciones entre estas mismas. Además, *ontology.owl* contiene información respecto a los países disponibles como el producto interno bruto, el nombre oficial de la moneda y más.

En general, los tres archivos posen muchos datos de distinta índole. Es por eso que de ahora en adelante los nuevos términos RDF se irán explicando a medida que se vayan usando.

## 4.4. Primeras consultas

Antes de empezar a hacer las consultas primero se hizo una revisión superficial de los archivos descritos en la sección anterior. Una vez que se tuvo una idea general de los contenidos de cada uno, se hizo la primera consulta.

```
SELECT DISTINCT ?predicate
WHERE {
    ?subject ?predicate ?object
}
```

Query 4.3: Consulta inicial

A pesar de ser una *query* muy simple, esta es de gran utilidad ya que muestra todos los términos RDF que actúan de conector entre otros dos términos.

Aquí surgió el primer problema con los datos que ofrece *Constitute*. La mayoría de estos términos no tienen una definición formal, esto quiere decir que su significado es inferido de su "nombre", como es el caso de *constowl:isConstitutionOf*. Esto es posible de hacer en muchos casos pero también hay términos que no son tan explicativos por sí solos. En estos casos al única opción es deducir el significado el termino viendo los resultados de alguna *query*.

También existen términos que le dan contexto o dan información sobre otros. En general, son términos que describen características de países y están definidos en el archivo *ontology.owl*. Un ejemplo son los términos *constowl:GDPTotalInCurrentPrices* y *constowl:GDPNotes*. El primero indica el producto interno bruto de un país y el segundo indica la fuente de la información.

A pesar de las dificultades mencionadas, de esta primera *query* fue posible sacar crear una lista términos RDF de interés que podrían proporcionar información importante. Los términos fueron los siguientes:

- constowl:nameListES

- constowl:isSectionOf
- constowl:isConstitutionOf
- constowl:InForceES
- constowl:ShowES

Considerando que para llevar a cabo el proyecto era necesario tener todos los países que tenían su constitución disponible en español, el termino *constowl:isConstitutionOf* fue el que mas llamó la atención en un principio por lo que se procedió a hacer una consulta con él. La *query* hecha fue la misma que en *query* 4.2 cambiando el limite por tres. Los resultados de esta *query* fueron:

| ?const | ?country |
|---|---|
| const:Abkhazia_1994 | constowl:Abkhazia |
| const:Afghanistan_1964 | constowl:Afghanistan |
| const:Afghanistan_2004 | constowl:Afghanistan |

Como se puede apreciar, todos los términos que representa *?const* son constituciones y tienen la estructura *const:[país]_[año]* mientras que todos los términos que representa *?country* son países y tienen la estructura *constowl:[país]*. Ahora, si queremos saber qué término RDF describen a los países, podemos hacer la siguiente *query*:

```
SELECT DISTINCT ?predicate
WHERE {
    ?const constowl:isConstitutionOf ?country.
    ?country ?predicate ?object.
}
```

Query 4.4: Consulta para buscar los terminos que describen a un país

Algunos de los resultados de esta consulta son:

| ?predicate |
|---|
| constowl:nameListES |
| constowl:nameListEN |
| constowl:nameCurrencyES |
| constowl:nameOfficialES |

A pesar de que los nombres son intuitivos se hicieron las consultas correspondientes con cada uno de estos para determinar sus significados. Por ejemplo, para el primer termino la consulta seria:

```
SELECT DISTINCT ?p
WHERE {
    ?country constowl:nameListES ?name.
}
LIMIT 3
```

Query 4.5: Consulta para comprobar el significado de *constowl:nameListES*

Esta *query* sin el límite arroja 260 respuestas. Con el limite de trees, los primeros tres resultados son:

| ?name |
|---|
| Unión del Magreb Árabe |
| Afganistán |
| Africa |

Viendo los resultados de esta *query* y haciendo la misma *query* para los otros términos de la tabla se llega a que :

- *constowl:nameListES*: el nombre en español del país.
- *constowl:nameListEN*: el nombre en ingles del país.
- *constowl:nameCurrencyES*: el nombre de la moneda del país en español.
- *constowl:nameOfficialES*: el nombre oficial del país en español. En el caso de Chile seria "La República de Chile".

El tipo consulta que se ve en la *query* 4.4 se puede hacer para buscar los términos RDF que describen a los distintos "tipos"de términos (países, constituciones, secciones, tópicos, etc). De ahora en adelante se asumirá que todos los términos nuevos que vayan apareciendo en las siguientes consultas fueron encontrados siguiendo esta misma lógica.

Se hicieron múltiples más consultas con las cuales se logró armar un grafo RDF que describe en parte los datos y entrega la información suficiente para hacer las consultas necesarias para el proyecto. Cabe mencionar que este es solo un sub-grafo del grafo completo de los datos ya que el grafo completo no es posible visualizarlo dado la cantidad de términos y conexiones entre ellos.

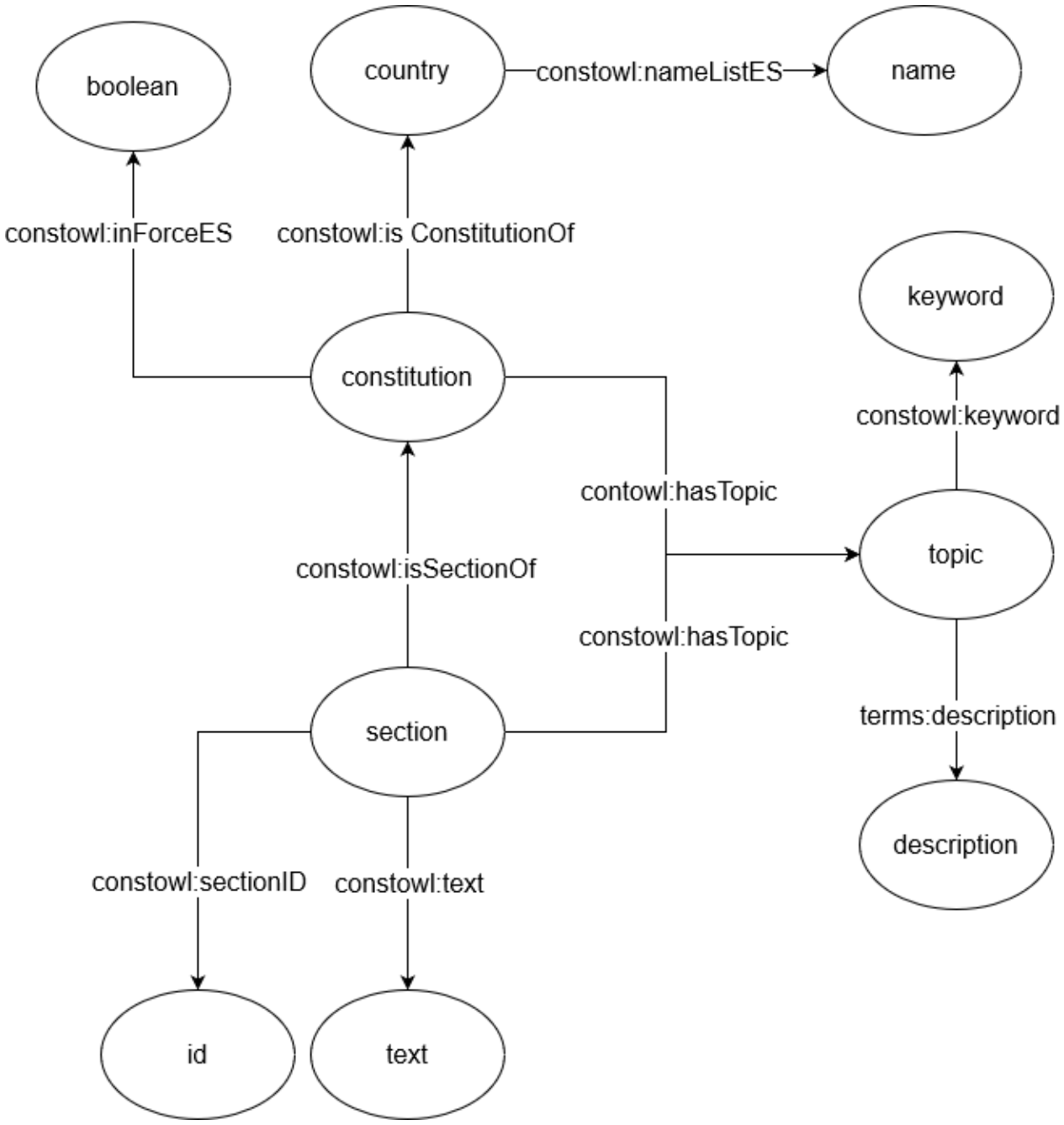


Figura 4.7: Sub-grafo RDF que describe la estructura de los datos relevantes de *Constitute*.

Describiendo los datos de manera mas informal se tiene que:

- Una constitución pertenece a un país, trata uno o mas tópicos y tiene una propiedad que indica si la constitución en si esta vigente o no[1].
- Un país tiene su nombre en español.
- Una sección pertenece a una constitución, tiene su texto en español, una *id* que lo identifica y trata uno o mas tópicos.

[1]Al avanzar en el trabajo se describió que esta ultima conclusión no era del todo correcta.

- Un tópico tiene su palabra clave en español que lo representa y tiene su descripción en español.

## 4.5. Creación del front-end

Para el *front-end* de este proyecto se usó el framework Vue.js. La razón para esta elección fue que se buscaba que la aplicación fuera compacta, no muy compleja y fácil de usar, y Vue.js es ideal para proyectos de este tipo gracias a su enfoque en la modularidad y sus *single file components*. Siguiendo esta idea de la simplicidad, tanto para la aplicación en si como el código, se decidió usar Vuetify ya que ofrecía los componentes necesarios para este proyecto.

La aplicación, en esta primera versión, consistiría de:

- Un selector para elegir el tópico que se desea buscar.
- Dos selectores de países.
- Y dos cajas de texto para mostrar los resultados.

Para esta versión se decidió dejar fuera algunas ideas que habían sido sugeridas durante la parte inicial de este trabajo. Una de ellas fue el mapa para elegir países y la razón para no implementarla fue que su aporte al proceso de búsqueda no era mucho y se decidió que con un simple selector/buscador bastaba. Otra idea que no se implemento fue la lista de temas mas buscados. A pesar de que podría aportar de gran manera a la aplicación, implementar esta lista en esta etapa del proyecto sería inútil dado que es necesario que la aplicación se encuentre en producción y a disposición de los usuarios para poder recolectar los datos por lo cual se decidió dejar esta idea para trabajo futuro. Finalmente otras ideas, como la división de tópicos en categorías más grandes, no fueron implementadas debido a que quedaba poco tiempo para finalizar el trabajo.

Con todo lo anterior en consideración, el *layout* pensado para el *front-end* del proyecto fue el que puede ser visto en la siguiente pagina de este informe, en la figura 4.9.

Con el diseño para esta primera versión listo, se procedio a implementer el *front-end*.

Dado que los selectores de los dos países son iguales en funcionamiento solo fue necesario crear una sola componente. Lo mismo aplica para los resultados. En total se crearon tres componentes:

- *TopicSelector.vue*: el selector para elegir el topico que se desea buscar.
- *CountrySelector.vue*: el selector para elegir el país deseado.
- *Extracts.vue*: el cuadro donde se muestran los resultados de la búsqueda.

Para la creación de los selectores se uso el modulo *v-autocomplete* de Vuetify. Esta componente funciona como selector pero también se puede escribir en el cuadro de texto para

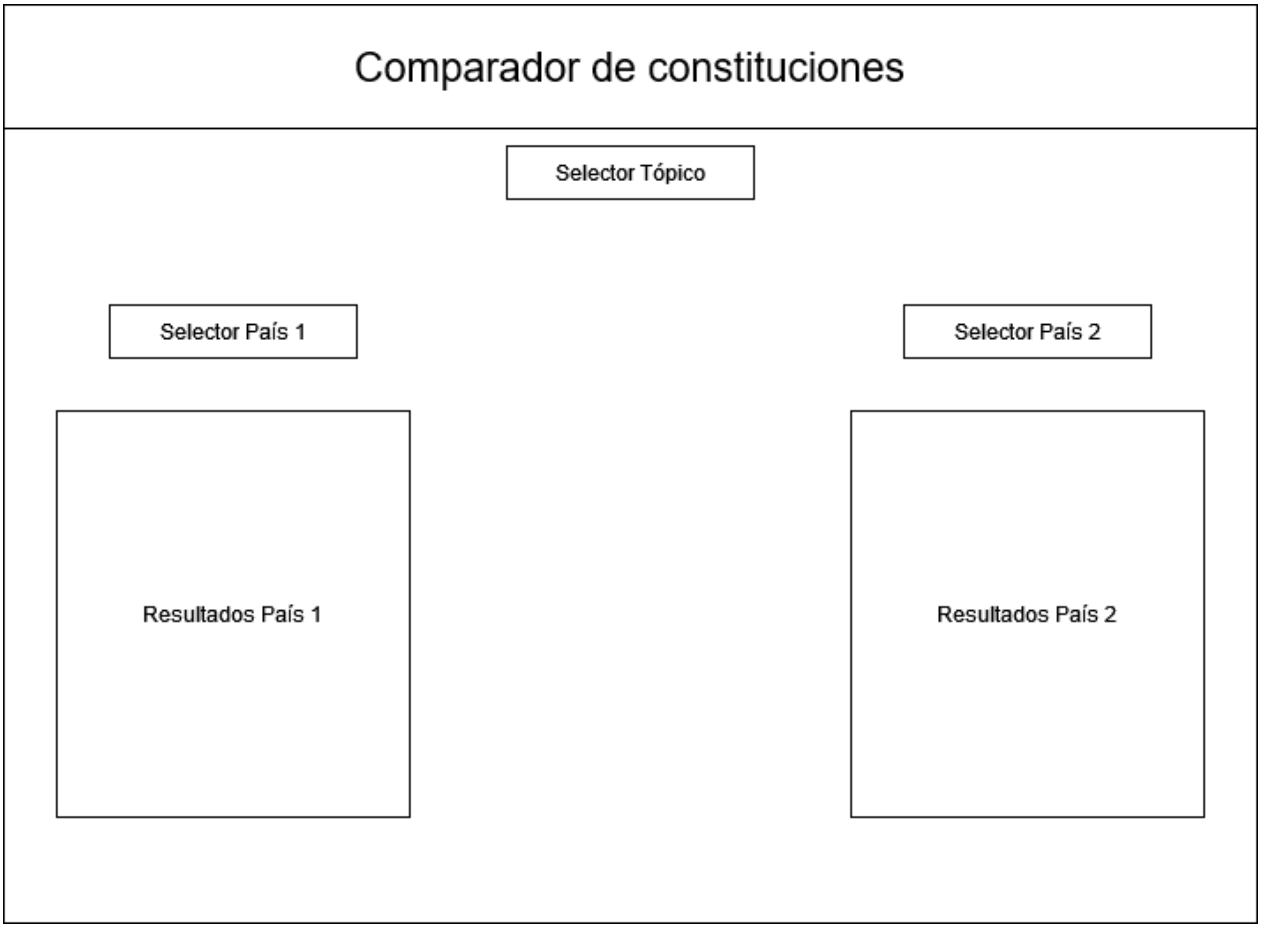


Figura 4.8: *Layout* ideado para el *front-end* de la aplicación

buscar algún item en especifico. Y para el cuadro de texto se uso *v-card*, también una componente ofrecida por Vuetify. Esta componente simplemente es un cuadro que puede contener cualquier otro tipo de componente o HTML puro.

Con las componentes creadas, se creó la componente principal de la aplicación donde se llamarían las otras componentes. En esta componente, llamada *App.vue*, se invocó una vez a *TopicSelector.vue*, dos veces a *CountrySelector.vue* y dos veces a *Extracts.vue*. A continuación se puede ver el resultado de esta implementación.

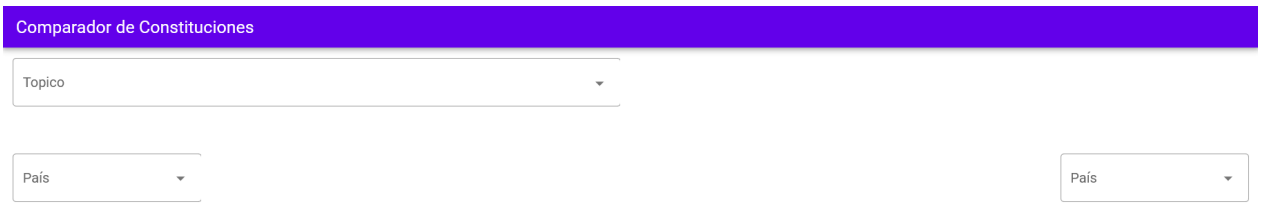


Figura 4.9: *Front-end* de la aplicación.

Un aspecto importante de Vue.js es que todas las componente son capaces de guardar variables y pasárselas entre ellas. Por ejemplo, si seleccionamos un tópico, el valor seleccionado

es guardado en la componente *TopicSelector.vue* y esta componente es capaz de enviar dicho valor a *Extracts.vue*.

## 4.6. Consultas definitivas y conexión del front-end con el back-end

Como aclaración, dado que la aplicación cuenta con dos selectores de países y dos cuadros para mostrar resultados solo se explicara una vez cada uno. En otras palabras, solo se describirá cómo se selecciona un tópico, un país y cómo se buscan y muestran los extractos de la constitución del país seleccionado que traten el tópico seleccionado. Esto ya que el proceso es el mismo para el segundo país.

Con el *front-end* ya implementado, se procedió a crear las consultas definitivas para el correcto funcionamiento de la aplicación. Para esto eran necesarias tres consultas:

- Una consulta para buscar todos los tópicos disponibles para rellenar el selector de tópicos.
- Una consulta para buscar todos los países disponibles para rellenar los selectores de países.
- Una última consulta para buscar los extractos.

Las *queries* para rellenar los selectores se ejecutan de manera inmediata apenas se entra a la página. De esta manera el usuario no debe esperar más de la cuenta.

#### Query hecha en la componente TopicSelector.vue

Para conseguir todos los tópicos disponibles para rellenar el selector de tópicos se hace la siguiente *query*:

```
SELECT DISTINCT ?keyword ?description
WHERE {
  ?const constowl:hasTopic ?topic.
  ?topic constowl:keyword ?keyword.
  ?topic terms:description ?description.
  FILTER (lang(?keyword) = "es").
  FILTER (lang(?description) = "es")
}
```

Query 4.6: Consulta para conseguir todos los tópicos y sus definiciones.

Con esta *query* se busca tanto la palabra en español que representa el tópico como la definición del tópico(también en español). Esto se hace ya que algunos tópicos no son muy claros con su nombre y se decidió mostrar su definición para ayudar al usuario.

Esta *query* se hace desde la componente *TopciSelector.vue*, enviando una solicitud HTTP al servidor de Fuseki. El servidor luego envía las respuestas al *front-end* en formato JSON. Usando *JavaScript* se accede a la respuesta, guardando los datos solicitados. Este proceso se repite con otras *queries* hechas desde otras componentes.

### Queries hechas en CountrySelector.vue

En la componente *CountrySelector.vue* se hacen dos consultas. La primera es:

```
SELECT DISTINCT ?name
WHERE {
  ?country constowl:nameListES ?name.
}
```

Query 4.7: Consulta para conseguir la lista de países

Con esta consulta se consigue la lista con los nombres en español de todos los países disponibles para rellenar el selector de países. La segunda *query* se ejecuta cuando un país es seleccionado. La *query* es:

```
SELECT ?const
WHERE{
  ?country constowl:nameListES ?name.
  ?const constowl:isConstitutionOf ?country.
  FILTER(CONTAINS(?name,'${country}'))
}
LIMIT 1
```

Query 4.8: Consulta para conseguir una constitución del país seleccionado

El termino *$*{*country*} representa el país seleccionado. El resultado de esta *query* es una constitución y tiene la forma *[prefijo del país]_[año]*. Con 'prefijo del país' nos referimos al *string* que representa al país en un termino RDF de una constitución. Por ejemplo, el prefijo de República Dominicana es 'Dominican_Republic'. Entonces si seleccionamos República Dominicana el resultado de la *query* tendría la forma *Dominican_Republic_[año]*.

El año del resultado no nos importa ya que lo que se desea saber es justamente el prefijo del país que fue seleccionado. Para conseguir el prefijo se usa *JavaScript* para conseguir el *substring* deseado. Este prefijo es el valor que será pasado desde la componente *CountrySelector.vue* a *Extracts.vue* para que desde esta última se pueda hacer la *query* para buscar los extractos de la constitución del país seleccionado que traten el tópico seleccionado. La razón para hacer esta segunda *query* y pasar el prefijo del país en vez de el nombre del país seleccionado será descrita en breve.

### Queries para conseguir los extractos

Cuando se selecciona un tópico, la componente *TopicSelector.vue* envía la palabra que representa al tópico seleccionado a la componente *Extracts.vue*. Y cuando se selecciona un país, la componente *CountrySelector.vue* envía el prefijo del país seleccionado también a la componente *Extracts.vue*. Con estos datos, desde *Extracts.vue* se hacen las consultas para buscar los extractos de la constitución del país seleccionado que traten el tópico del país seleccionado.

En esta componente se hacen dos *queries* . La primera es para conseguir la constitución vigente del país seleccionado y la segunda para buscar los extractos en la constitución que se consiguió con la primera. En un principio, para lograr esto, solo iba a ser necesario el nombre del país seleccionado, por lo cual no hubiera sido necesario hacer la segunda *query* en *CountrySelector.vue* para conseguir el prefijo del país. Pero al tratar de implementar este plan surgieron problemas.

La primera *query* que se intentó hacer fue la siguiente:

```
SELECT DISTINCT ?const
WHERE {
  ?const constowl:isConstitutionOf constowl:Chile.
  ?const constowl:InForceES "True".
  ?const constowl:YearEvent ?year.
}
ORDER BY DESC(?year)
LIMIT 1
```

Query 4.9: Consulta para conseguir la constitución vigente del país

En este ejemplo se usó a Chile como el país seleccionado. Esta *query* se hizo creyendo que el termino RDF *constowl:inForceES* indicaba que el resultado seria la constitución vigente del país. El primer problema fue que si no se pone LIMIT 1, la *query* arrojaba mas de una constitución. En el caso de Chile los resultados eran tres: *const:Chile_2012*, *const:Chile_2014*, *const:Chile_2015*. A pesar de esto se decidió seguir adelante seleccionando la constitución más reciente de la respuesta.

Luego, con la constitución encontrada, se hizo la *query* para encontrar los extractos. Siguiendo con el ejemplo de Chile, la *query* fue la siguiente:

```
SELECT DISTINCT ?text
WHERE {
  ?section constowl:isSectionOf const:Chile_2015.
  ?section constowl:sectionID ?id.
  ?section constowl:text ?text.
  ?section constowl:hasTopic ?topic.
  ?topic constowl:keyword ?k
  FILTER (CONTAINS(?k,'${topic}'))
  FILTER (lang(?k) = 'es')
  FILTER(lang(?text) = 'es')
}
ORDER BY (xsd:int(?id))
```

Query 4.10: Consulta para conseguir los extractos

Con esta *query* el plan era conseguir los extractos de la constitución de Chile que trataran el tópico seleccionado, representado por $\$\{topic\}$. El problema es que el resultado de la *query* fue vació. Para ver cual fue el problema se probo con una *query* mas simple.

```
SELECT DISTINCT ?section
WHERE {
  ?section constowl:isSectionOf const:Chile_2015.
}
```

Query 4.11: Consulta simplificada

Nuevamente el resultado fue vació. Esto significaba que a pesar de ser clasificada como una constitución, esta no poseía secciones, es decir, es una constitución vacía. Para seguir investigando se hizo una *query* para conseguir todas las constituciones de Chile que no fueran vacías.

```
SELECT DISTINCT ?const
WHERE {
  ?const constowl:isConstitutionOf constowl:Chile.
  ?section constowl:isSectionOf ?const
}
```

Query 4.12: Consulta para conseguir las constituciones no vacías de Chile

Los resultados fueron dos: *const:Chile_1925* y *const:Chile_2018D*. Estos resultados no hacían sentido ya que el primero es una constitución deprecada y el segundo es un borrador. Pero se sabia que debía existir la constitución actual de Chile por lo cual se hizo la siguiente consulta:

```
SELECT DISTINCT ?const
WHERE{
  ?section constowl:isSectionOf ?const.
  FILTER(CONTAINS(STR(?const),'Chile'))
}
```

Query 4.13: Consulta para conseguir constituciones no vacías cuyos terminos RDF contengan la palabra 'Chile'

Aquí los resultados fueron tres: *const:Chile_1925*, *const:Chile_2018D const:Chile_2021*. Es decir que la constitución mas reciente de Chile no es considerada una constitución de Chile según los datos.

Encontrar la razón de este error tomó gran tiempo dado lo ilógico de la situación. Pero con el problema ya identificado fue posible encontrar una alternativa por lo cual ahora se describirá como la componente *Extracts.vue* consigue los extractos.

La *query* que se hace para conseguir la constitución vigente del país seleccionado es la siguiente:

```
SELECT DISTINCT ?const
WHERE{
  ?section constowl:isSectionOf ?const.
  FILTER(CONTAINS(STR(?const),'${this.country}'))
}
ORDER BY DESC(?const)
LIMIT 1
```

Query 4.14: Consulta para conseguir la constitución vigente del país seleccionado

Dado que los términos RDF de las constituciones son de la forma *[prefijo del país]_[año]*, es necesario que *${this.country}* sea el prefijo del país y no su nombre ya que estos son distintos como se vio con el ejemplo de República Dominicana.

Finalmente, la *query* para buscar los extractos es:

```
SELECT DISTINCT ?text
WHERE {
  ?section constowl:isSectionOf ?const.
  ?section constowl:sectionID ?id.
  ?section constowl:text ?text.
  ?section constowl:hasTopic ?topic.
  ?topic constowl:keyword ?k
  FILTER(CONTAINS(STR(?const),"${constitution}"))
  FILTER (CONTAINS(?k,"${this.topic}"))
  FILTER(lang(?text) = 'es') }
ORDER BY (xsd:int(?id))
```

Query 4.15: Consulta para conseguir los extractos

En esta *query*, $\${contitutio}$ representa la constitución encontrada en la *query* anterior y $\${this.topic}$ representa el tópico elegido.

De esta forma se completó la implementación de esta primera versión de la aplicación.

## 4.7. Funcionamiento de la aplicación

Al entrar a la aplicación se llega a la *landing page* que se puede ver en la figura 4.10. Aquí se pueden el selector de tópicos y los dos selectores de países.

Al hacer click en el selector de tópicos se puede ver la lista de tópicos disponibles y al pasar el mouse por encima de alguno se puede ver su definición a la izquierda como se ve en la figura 4.11

Luego al hacer click en uno de los selectores de países se muestra la lista de países disponibles como se ve en la figura 4.12. Otra opción es buscar el país deseado introduciendo el nombre. EN la figura 4.13 se escribió el string "Chi" para buscar a Chile.

Al seleccionar el país, la aplicación muestra los resultados como se ve en la figura 4.14. Luego se puede seleccionar un segundo país para comparar. En este caso se comparó Chile con Estados Unidos de América como se ve en la figura 4.15

En caso de que el resultado sea vacío la aplicación mostrara un mensaje indicando que la constitución del país seleccionado no tiene artículos que traten el tópico seleccionado. Esto se puede observar en la figura 4.16


Figura 4.10: *Landing page* de la aplicación.

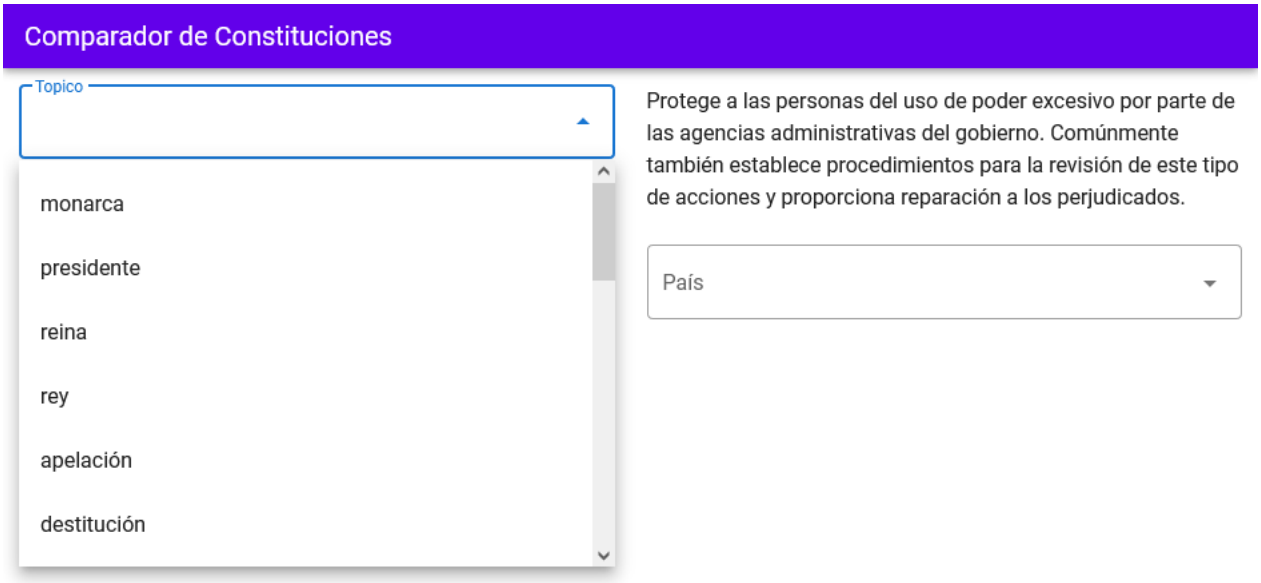


Figura 4.11: Selección de un tópico. A la izquierda se muestra la definición del tópico sobre el que se encuentra el cursor. En este ejemplo el tópico es 'presidente'.

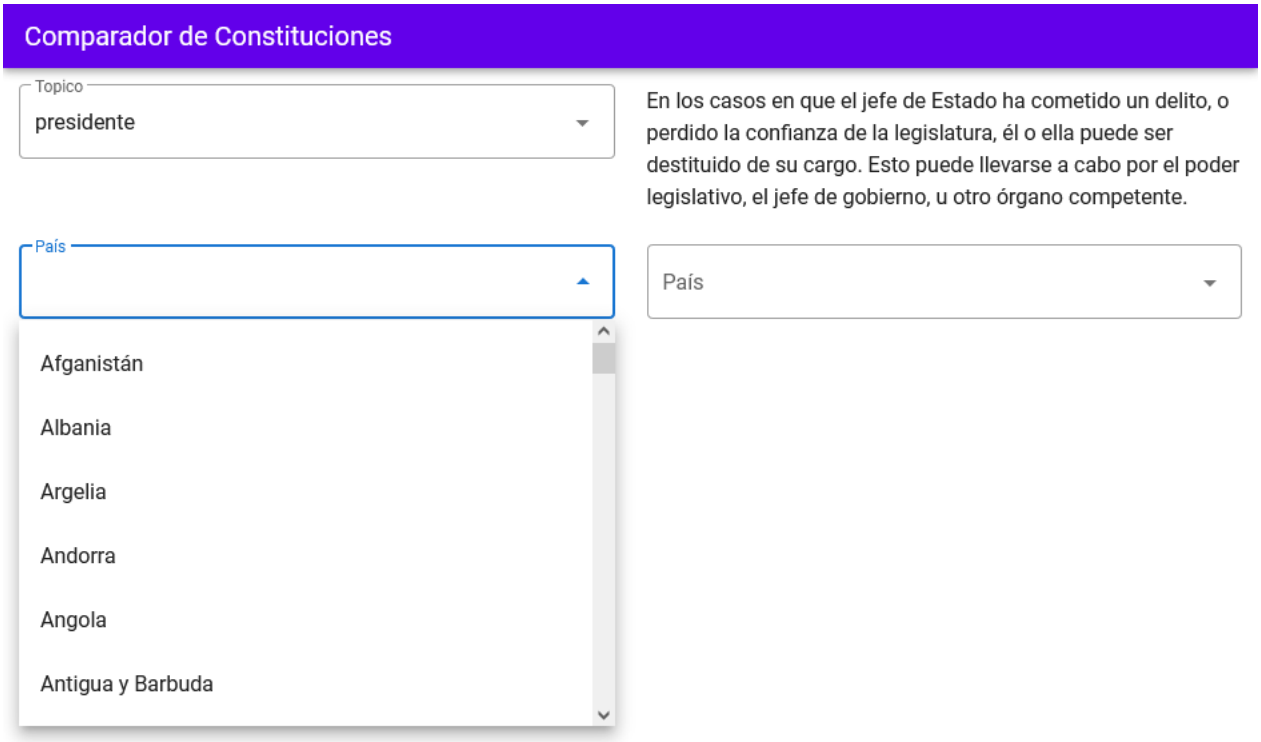


Figura 4.12: Selector de paises

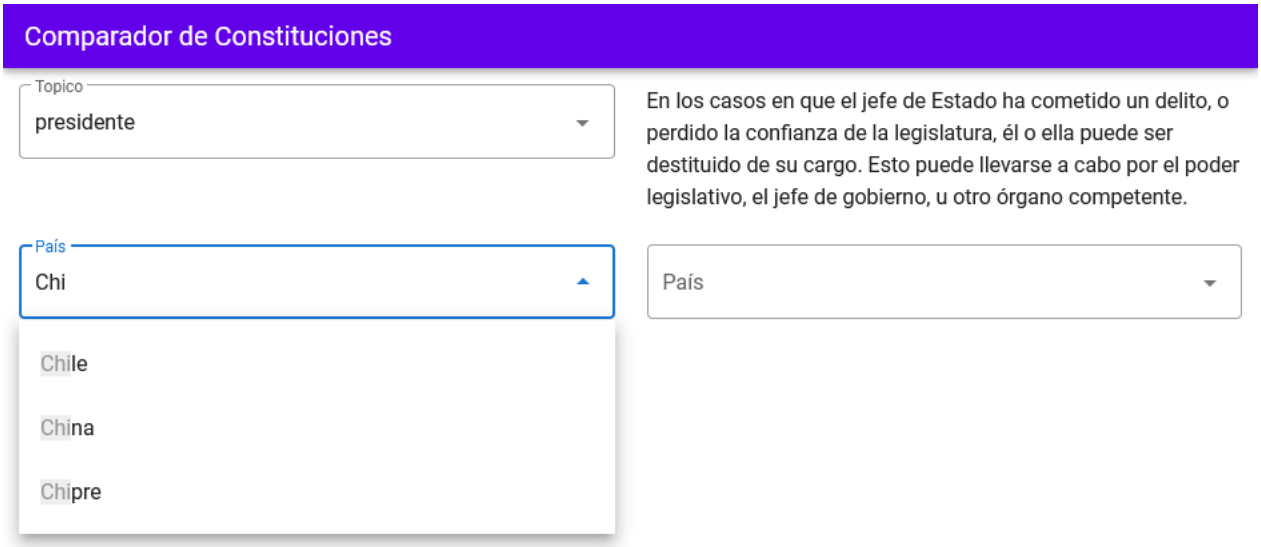


Figura 4.13: Búsqueda por nombre

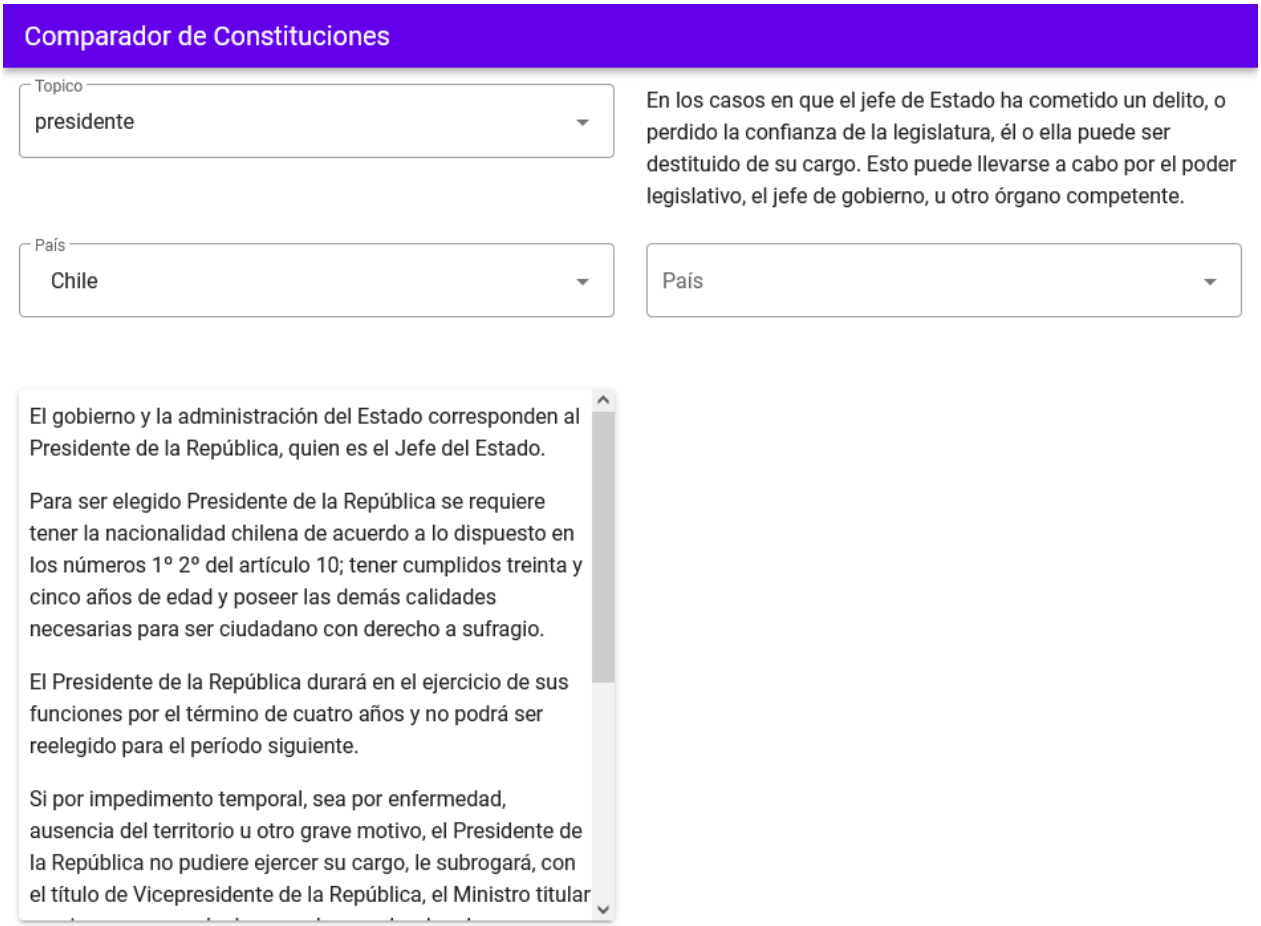


Figura 4.14: Resultados

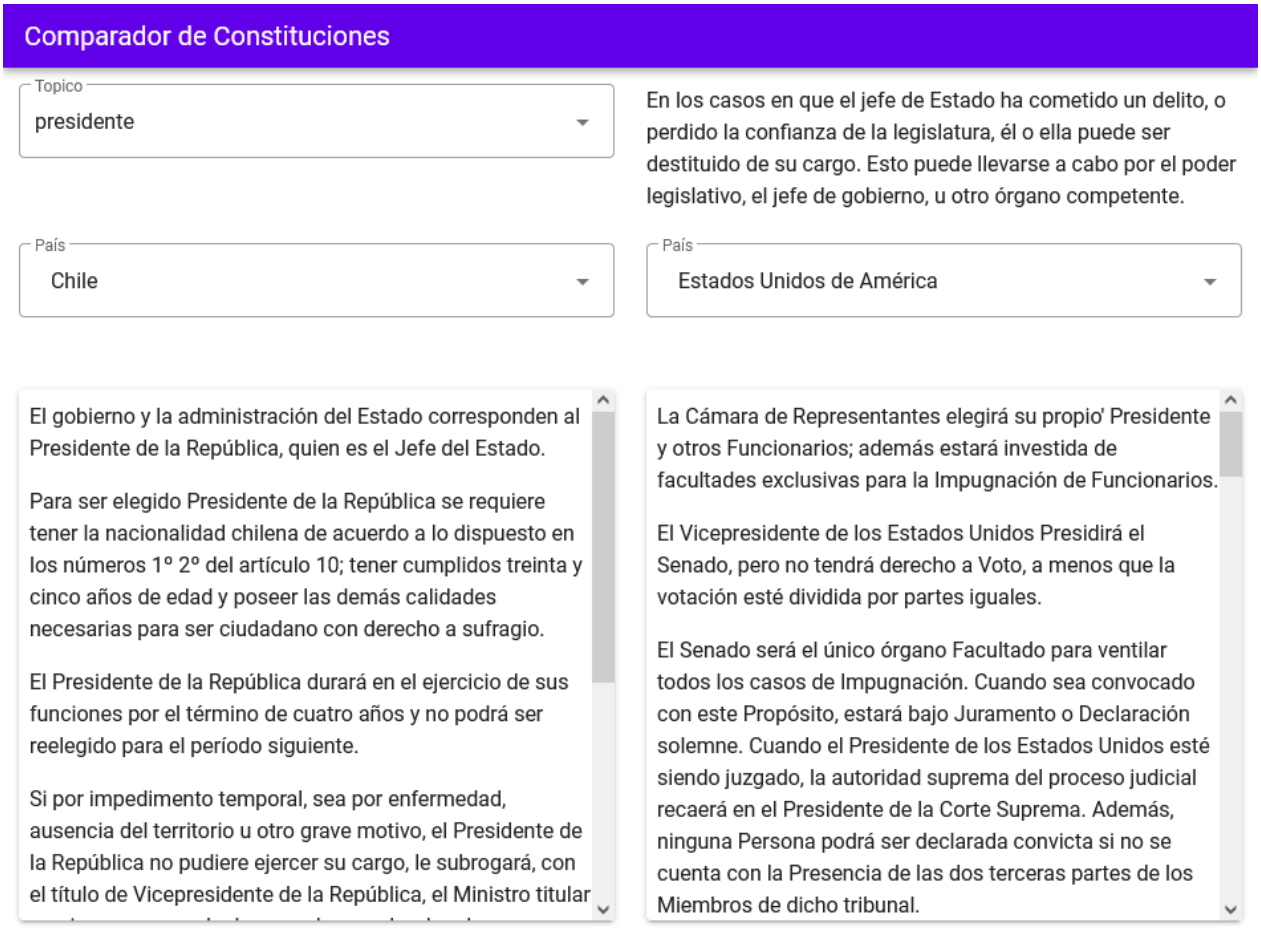


Figura 4.15: Resultados del segundo país

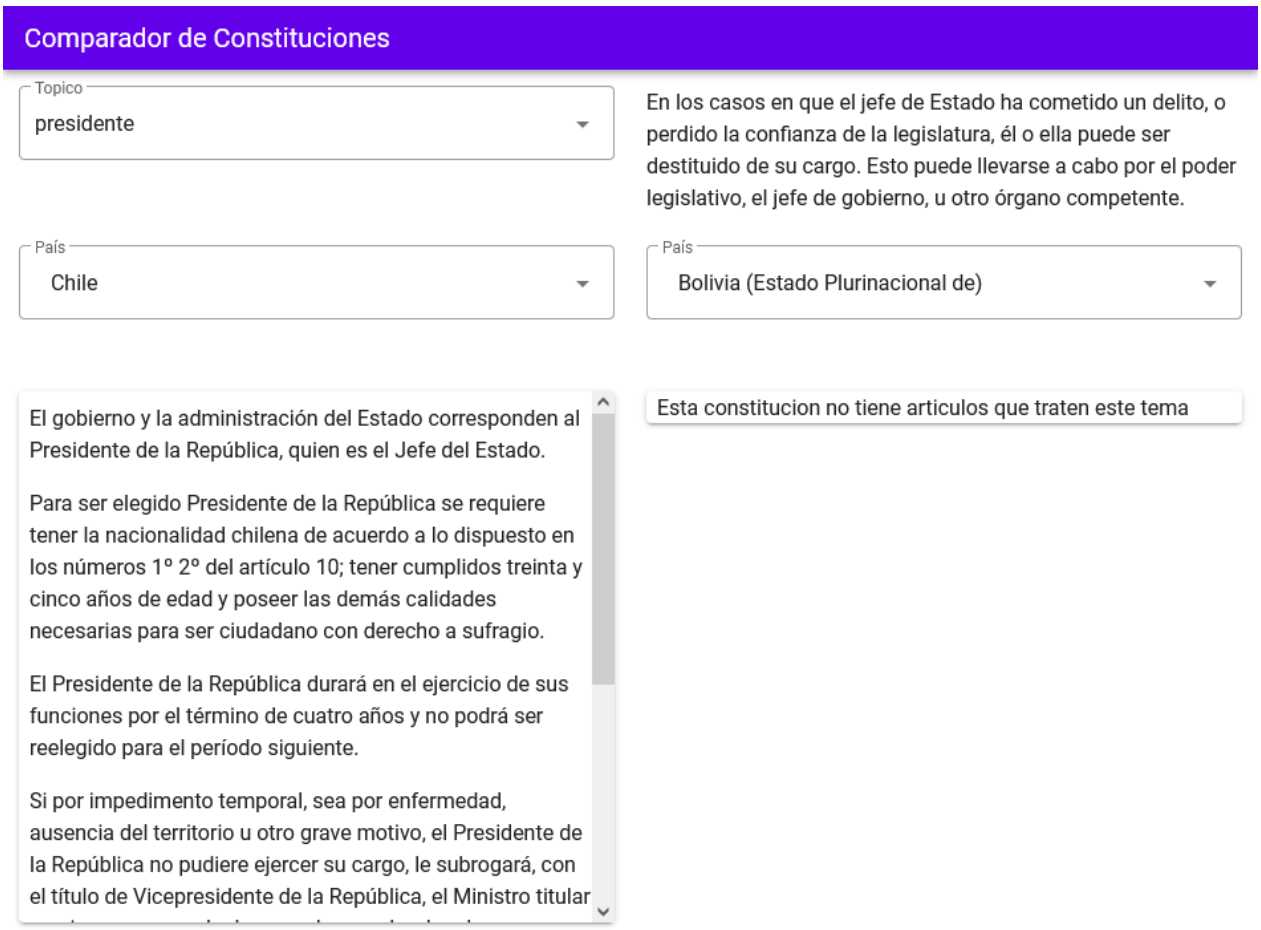


Figura 4.16: Mensaje avisando que no se encontraron resultados

# Capítulo 5

# Validación

Originalmente se planeaba subir la aplicación a un servidor del Departamento de Ciencias de la Computación de la Universidad de Chile para que pudiera ser accedido por cualquier persona. Pero debido a los problemas que surgieron durante la implementación de la aplicación esto no fue posible. En especifico, detectar y solucionar el problema de conseguir las constituciones vigentes tomo bastante tiempo.

A pesar del problema mencionado fue posible hacer pruebas de usuaria probando la aplicación de forma local. Para validar la aplicación se hicieron varias pruebas con distintos usuarios. La primera prueba se hizo con egresado de derecho de la Universidad de Concepción. Las siguientes pruebas se hicieron con gente no familiarizada con el derecho constitucional.

A todos los usuarios se les pidió contestar una encuesta para calificar la aplicación. Esta encuesta se basa en la *System Usability Scale*[16], una herramienta para medir la estabilidad de una aplicación o pagina web. El formato de la encuesta y los resultados serán mostrados más adelante.

Primero se describirán las pruebas de usuarios hechas y las primeras impresiones de estas para luego discutir más en profundidad los resultados de las pruebas y también los resultados de la encuesta.

## 5.1. Validación con estudiante de derecho

La primera prueba de usuario se realizó con un egresado de derecho de la Universidad de Concepción, por lo cual posee conocimiento respecto al derecho constitucional.

Para comenzar la prueba se le comentó al usuario sobre la plataforma y su objetivo principal que es permitir comparar constituciones. Una vez informado, el usuario procedió a usar la aplicación por aproximadamente 10 minutos. El usuario hizo varias consultas, eligiendo distintos tópicos y países. Una vez terminado los 10 minutos se le hizo un pequeña entrevista al usuario para saber sus primeras impresiones del sistema, si lo encontró fácil o difícil de usar, si encontró alguna falla importante y cuál cree él que podría ser la utilidad de esta

herramienta.

A grandes rasgos el usuario calificó como buena su experiencia usando la plataforma comentando que es muy fácil de usar y que se agradece la simplicidad de la interfaz. Un punto bajo que menciono el usuario es que para muchos países no existían resultados para ningún tópico que había elegido.

Se converso más en profundidad sobre las fallas del sistema que se dejará para la sección de discusión de este capitulo.

## 5.2. Validación con usuarios sin conocimiento previo

El resto de todas las pruebas de usuarios fueron con gente no familiarizada con el derecho constitucional. En total se hicieron 6 pruebas de usuarios de este tipo. En general estas pruebas fueron más cortas, con los usuarios usando la aplicación por unos 7 u 8 minutos para luego dejar un comentario y responder la encuesta.

En general los comentarios fueron positivos, destacando nuevamente la facilidad para usar el sistema. Algunos usuarios comentaron que algunas definiciones de tópicos no eran muy claras. Además, algunos usuarios también tuvieron el problema de que muchas constituciones no contenían artículos de los tópicos que buscaban.

## 5.3. Discusión

A pesar de que la cantidad de pruebas de usuarios que se llevaron a cabo es pequeña fue posible conseguir información valiosa e identificar fallas en el sistema.

La primera prueba de usuario con el egresado de derecho fue de gran importancia ya que dio comentarios importantes sobre algunos resultados.

En primer lugar, el usuario comentó sobre la lista de tópicos y sus definiciones. El usuario encontró que la lista de tópicos estaba bien compuesta pero que habían algunos tópicos que eran demasiado específicos y que muy pocas constituciones los tratarían. Un ejemplo seria el tópico de "derechos de autor". Durante toda la prueba no se logró encontrar ninguna constitución que tratara este tópico.

Además el usuario mencionó que algunas definiciones de tópicos eran muy difíciles de entender y que algunas definiciones parecían no describir el tópico correspondiente. Este problema también fue mencionado por el resto de los usuarios que hicieron pruebas. Investigando sobre los datos se logró encontrar el origen del problema. En un principio se creía que un tópico tenia una sola descripción, como se mostró en la figura 4.8 ubicada en la sección 4.4 de este informe. Se hicieron consultas SPARQL para comprobar esto pero se descubrió que un tópico en realidad tenía varias descripciones y el sistema solo muestra una de estas. Para poder entender mejor este problema y solucionarlo es necesario un mayor entendimiento de

los datos por lo cual esto quedaría para trabajo futuro.

Otro comentario que hizo el egresado de derecho fue que los extractos de algunas constituciones eran muy pocos descriptivos. Por ejemplo, al consultar por los extractos de la constitución de Alemania que trataran el tópico 'presidente' los resultados son los siguientes:

- Elección
- Incompatibilidades
- Suplencia
- Acusación ante la Corte Constitucional Federal Composición
- Atribuciones en el seno del Gobierno Federal

Pero conversando con el usuario y haciendo unas consultas en la aplicación se llegó a una conclusión. Todos los países que muestran extractos pocos descriptivos son países con estados federados. Estos países tienen constituciones muy amplias y que no tratan temas en profundidad ya que los estados de estos países tiene una mayor independencia y se regulan a ellos mismos. Ejemplos de países de este estilo son Alemania y Estados Unidos. Entonces este no es un problema de la plataforma o de los datos, es querer comparar dos textos que no son del mismo tipo.

El último problema que mencionaron todos los usuarios es que muchas constituciones parecían estar vacías. La razón de este error fue que el selector de países muestra todos los países que existen en los datos y no solo los que tienen su constitución en español. Esto se arregla cambiando la consulta SPARQL.

## 5.4. Resultados de la encuesta

La encuesta realizada consiste de 10 preguntas que los usuarios deben responder. Cada pregunta consiste de una frase y el usuario debe asignarle un valor de 1 a 5 a su respuesta dependiendo que tan de acuerdo esta con la frase, con 1 si están muy en desacuerdo y 5 si están muy desacuerdo. Las preguntas son las siguientes:

1. Creo que me gustaría utilizar este sistema con frecuencia.
2. Encontré el sistema innecesariamente complejo.
3. Pensé 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 de 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 utilizar este sistema muy rápidamente.
8. Encontré el sistema muy complicado de usar.
9. Me sentí muy seguro usando el sistema.
10. Necesitaba aprender muchas cosas antes de empezar con este sistema.

Los resultados de esta encuesta se calculan creando nuevos valores de la siguiente forma:

- A los resultados de las preguntas impares se les resta 1.
- Los resultados de las preguntas pares son restados de 5.

Estos nuevos valores son sumados y el resultado es multiplicado por 2.5. Este resultado se encuentra en el rango 0 a 100. A pesar de que este formato de encuesta es muy usado no existe un estándar para interpretar los resultados, pero se cree que un puntaje mayor a 68 es mejor que el promedio[1].

En total hubo 7 respuestas a esta encuesta cuyos resultados promedian 79 de 100, con 75.5 siendo la nota más baja y 82.5 la más alta. Con este promedio se puede ver que la aplicación esta sobre el promedio de 69 mencionado anteriormente.

Revisando las respuestas, en general los usuarios estaban muy de acuerdo con que el sistema es fácil de usar e intuitivo. Las únicas preguntas que los usuarios estuvieron más en desacuerdo fueron las preguntas 5 y 6. Estas preguntas se relacionan con cómo están integradas las distintas artes del sistema y la consistencia de la aplicación. Se cree que para la pregunta 6 en específico afecto de gran manera el error de mostrar constituciones vacías y que las definiciones de los tópicos no eran muy claras.

Pero en general con estas respuestas se cree que se logró cumplir el objetivo planteado de que cualquier persona pudiera usar la aplicación sin problema.

[1] `https://measuringu.com/sus/`

# Capítulo 6

# Conclusión

Para concluir este informe se hará una evaluación del trabajo realizado y se enumerarán formas en las que se podría mejorar el sistema en un trabajo a futuro.

## 6.1. Trabajo realizado

Con el trabajo realizado durante esta memoria se logró cumplir con la mayoría de los objetivos descritos en la sección 1.4.2 de este informe.

Los primeros cinco objetivos, relacionados con la implementación de la aplicación, fueron cumplidos de forma completa. La aplicación permite que los usuarios comparen extractos de dos constituciones que traten algún tópico en específico. Además se cumplió un objetivo principal del trabajo que era hacer esta aplicación intuitiva y fácil de usar. A pesar de que estos objetivos fueron cumplidos y que la aplicación en sí no presenta errores en su funcionamiento, sí existen errores que afectan la experiencia del usuario. Un ejemplo de esto es que el selector de países muestra países que no tienen sus constituciones en español. Al consultar por estas constituciones el sistema indica de manera correcta que no hay artículos que traten este tema (en español). Pero para una mejor experiencia sería mejor que estos países no estuvieran disponibles en el selector.

El objetivo número 6 fue cumplido de forma parcial ya que dado los problemas que surgieron durante el desarrollo de la aplicación no fue posible hacer la cantidad de pruebas de usuarios que se deseaba. Más en detalle, entender la ontología de los datos tomó más tiempo de lo planeado y esta al ser no tan clara llevó a que solucionar el problema de encontrar las constituciones vigentes fuese muy complejo. Pero es importante destacar que la prueba de usuario hecha con el egresado de derecho aportó de gran manera a la evaluación de la aplicación y ayudó a comprender algunos problemas que no tenían que ver con la aplicación en sí.

El objetivo numero 7 fue cumplido a cabildada y se creó una lista de ideas que pueden ser implementadas a futuro para mejorar la aplicación. Esta lista se puede ver en la siguiente sección de este capitulo.

Dado la falta de tiempo que causaron los problemas descritos en este informe no fue posible dejar la aplicación en un servidor para que cualquiera pudiera usarla por lo cual el objetivo número 8 no pudo ser cumplido.

En resumen, se logro cumplir con el objetivo principal del proyecto que era crear una aplicación para comparar constituciones. A pesar de esto, es importante destacar que esta es una primera versión de la aplicación y que es necesario hacer múltiples mejoras para que esta sea atractivas para los usuarios y pueda aportar a democratizar la información.

## 6.2. Trabajo a futuro

El trabajo realizado tiene varios errores que deben ser arreglados para mejorar la experiencia del usuario. Además existen múltiples ideas que se pueden llevar a cabo para aumentar el valor de la aplicación.

A continuación se listarán las acciones que se pueden llevar acabo para cumplir los objetivos antes mencionados:

- Investigar más a cabalidad los datos ofrecidos por *Constitute* para poder entender de mejor manera la ontología de estos y así poder hacer mejores consultas.
- Relacionado al punto anterior, investigar sobre las definiciones de los tópicos para que al mostrarlas estas aporten a la experiencia del usuario en el uso de la aplicación.
- Arreglar el selector de países para mostrar solo los países cuyas constituciones se encuentren en español. Esto se lograría cambiando la consulta SPARQL que se hace para conseguir esa lista.
- Diseñar un nuevo *look* para la aplicación, manteniendo la simplicidad de la interfaz pero mejorando el aspecto de la pagina para atraer a más usuarios.
- Agregar filtros para la selección de países usando los datos ofrecidos por *Constitute*. Algunos ejemplos son filtrar por región, por tamaño de la población o por el producto interno bruto del país.

Es importante que los futuros trabajos también sean validados por usuarios para así poder seguir mejorando la aplicación y agregando funcionalidades que enriquezcan la experiencia del usuario.

# Bibliografía


[1] Comparative constitutions project homepage. (accedido el 4 de mayo de 2021).

[2] Constituciones hispanoamericanas. (accedido el 4 de mayo de 2021).

[3] Constitución de 1980 y actas constitucionales. (accedido el 4 de mayo de 2021).

[4] Convención constitucional. (accedido el 4 de mayo de 2021).

[5] Discovering the top websites built with vue js. (accedido el 4 de mayo de 2021).

[6] Evan You Interview. (accedido el 27 de julio de 2021).

[7] Getting Started with Semantic Technologies. (accedido el 23 de julio de 2021).

[8] La cronología del estallido social de chile. (accedido el 4 de mayo de 2021).

[9] Plebiscito: Qué es la convención constitucional y cuáles son sus diferencias con la convención mixta. (accedido el 4 de mayo de 2021).

[10] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 74–83. ACM, 2004.

[11] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the Semantic Web recommendations. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th International Conference on World Wide Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 74–83. ACM, 2004.

[12] Martin Dürst and Michel Suignard. Internationalized Resource Identifiers (IRIs). RFC 3987, January 2005. `http://www.ietf.org/rfc/rfc3987.txt`.

[13] Zachary Elkins, Tom Ginsburg, and James Melton. Constitute: The World's Constitutions to Read, Search, and Compare. Online at constituteproject.org, 2021. `http://constituteproject.org`.

[14] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Springer, 2009.

[15] Aidan Hogan. *The Web of Data*. Springer, 2020.

[16] James R. Lewis. The System Usability Scale: Past, Present, and Future. *Int. J. Hum. Comput. Interact.*, 34(7):577–590, 2018.

[17] Universidad de Chile. La Constitución en Debate. Curso online gratuito, 2020. `https://uabierta.uchile.cl/asset-v1:Universidad_de_Chile+REP_01+2020+type@asset+block@programaCD2020FINALpdf.pdf`.