UTPL

Universidad Técnica Particular de Loja

Sistemas Informáticos y Computación

Integrantes:

- Kevin José Quito Medina
- Freddy Stalin Villavicencio Espinoza

Asignatura: Sistemas Basados en Conocimiento

Docente: Ing. Janneth Alexandra Chicaiza Espinosa

INTRODUCCIÓN

RDF es un método para expresar el conocimiento en un mundo descentralizado y es el fundamento de la Web Semántica, en el que las aplicaciones informáticas utilizan información estructurada distribuida por toda la Web. Proporciona interoperabilidad entre aplicaciones que intercambian información legible por máquina en la Web. RDF destaca por la facilidad para habilitar el procesamiento automatizado de los recursos Web.

RDF puede utilizarse en distintas áreas de aplicación: como en el presente trabajo, en donde se realizo un proceso de obtención y limpieza de datos acerca de los casos de COVID-19 en el continente Africano para la posterior generación y almacenamiento de datos RDF, culminando con el desarrollo de una aplicación web en donde se evidencia los resultados obtenidos.

El siguiente informe está estructurado de acuerdo con las etapas del proyecto de la siguiente manera:

- Obtención de datos: En esta etapa se describen las fuentes de donde se tomaron los datos para la posterior generación de datos RDF.
- Transformación de datos RDF: Aquí se describen las tareas de limpieza, selección de patrones URIs, la lógica de transformación de datos con JENA y los resultados de la transformación. Además de indicar el repositorio elegido para el almacenamiento de datos RDF.
- Aplicación: En este apartado se describen los trabajos relacionados, además de la arquitectura y tecnologías empleadas para el desarrollo de la aplicación.
- Conclusiones: en esta etapa se indican las conclusiones del proyecto.

DESARROLLO

1. OBTENCIÓN DE DATOS

En el presente proyecto se trabajaron con dos fuentes de datos, que se describen a continuación:

Fuente	URL	Formato de datos	Actualización
COVID-19 África	http://covid-19-africa.sen.ovh/index.php?isnc=2	CSV	Actualización diaria
Coronavirus COVID-19 (2019-nCoV) Data Repository for Africa	https://github.com/dsfsi/covid19africa	CSV	Actualización hace un mes

La fuente **COVID-19 África** nos brinda información acerca de estadísticas de casos confirmados, recuperados y fallecidos de cada uno de los países de África. La fuente **Coronavirus COVID-19 (2019-nCov) Data Repository for África** nos ofrece información sobre los casos confirmados (Género, Fecha de confirmación, Edad, etc). Realizamos la descarga de la data en formato CSV desde las fuentes mencionadas. La data se encuentra actualizada hasta el día 16/06/2020.

2. OBTENCIÓN DE DATOS RDF

2.1. Tareas de limpieza

En base al modelo ontológico definido, se estableció un modelo de base de datos relacional que se adapta a las fuentes de datos recolectadas mencionadas anteriormente (Figura. 1).

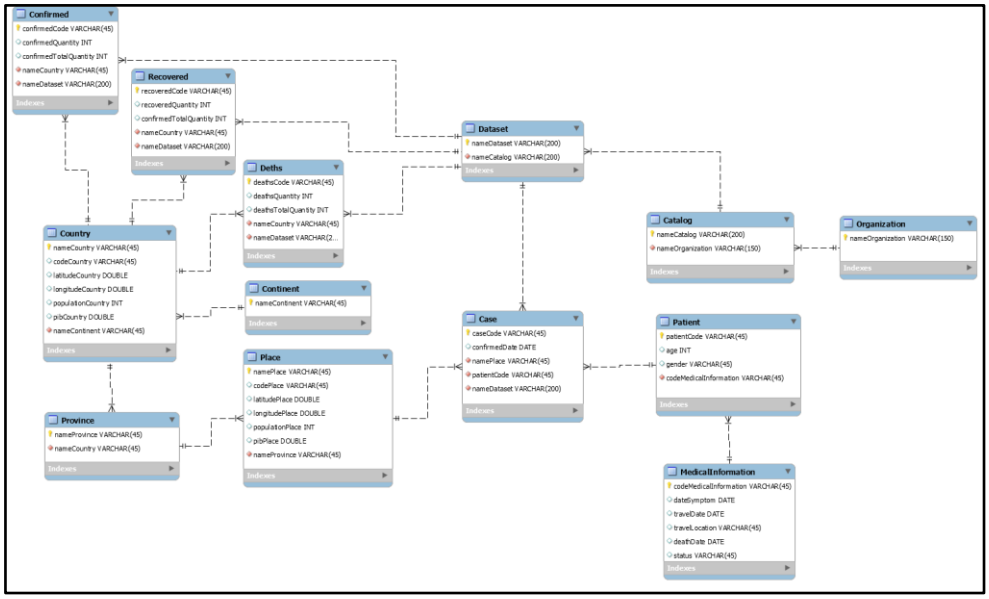
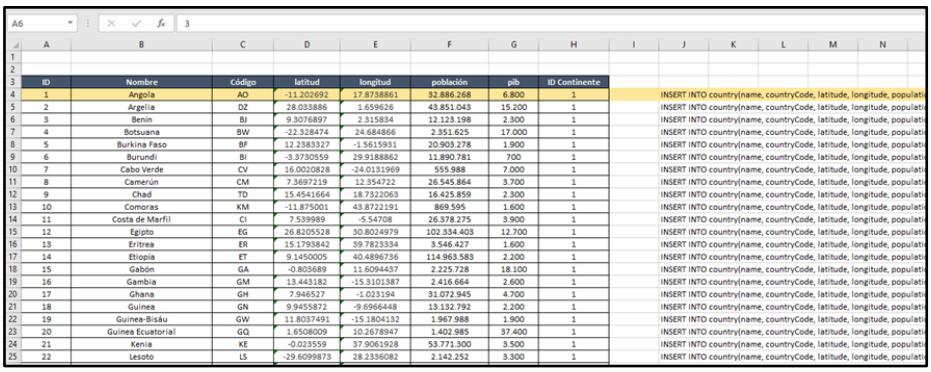


Figura. 1. Modelo relacional de base de datos.

A continuación, se realizó la limpieza de los datos, separándolo por entidades con sus respectivos atributos y relaciones. Además, se realizó el cambio en el formato de las fechas. A sí mismo, se generaron los INSERT SQL de forma automática. Estas tareas se las realizó en Excel (Figura. 2).



ID	Nombre	Código	latitud	longitud	población	pib	ID Continente	
1	Angola	AO	-11.202692	17.8738861	32.886.268	6.800	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
2	Argelia	DZ	28.033886	1.659626	43.851.043	15.200	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
3	Benin	BJ	9.3076897	2.315834	12.123.198	2.300	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
4	Botsuana	BW	-22.328474	24.684866	2.351.625	17.000	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
5	Burkina Faso	BF	12.2383327	-1.5615931	20.903.278	1.900	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
6	Burundi	BI	-3.3730559	29.9188862	11.890.781	700	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
7	Cabo Verde	CV	16.0020828	-24.0131969	555.988	7.000	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
8	Camerún	CM	7.3697219	12.354722	26.545.864	3.700	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
9	Chad	TD	15.4541664	18.7322063	16.425.859	2.300	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
10	Comoras	KM	-11.875001	43.8722191	869.595	1.600	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
11	Costa de Marfil	CI	7.539989	-5.54708	26.378.275	3.900	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
12	Egipto	EG	26.8205528	30.8024979	102.334.403	12.700	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
13	Eritrea	ER	15.1793842	39.7823334	3.546.427	1.600	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
14	Etiopía	ET	9.1450005	40.4896736	114.963.583	2.200	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
15	Gabón	GA	-0.803689	11.6094437	2.225.728	18.100	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
16	Gambia	GM	13.443182	-15.3101387	2.416.664	2.600	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
17	Ghana	GH	7.946527	-1.023194	31.072.945	4.700	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
18	Guinea	GN	9.9455872	-9.6966448	13.132.792	2.200	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
19	Guinea-Bisáu	GW	11.8037491	-15.1804132	1.967.988	1.900	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
20	Guinea Ecuatorial	GQ	1.6508009	10.2678947	1.402.985	37.400	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
21	Kenia	KE	-0.023559	37.9061928	53.771.300	3.500	1	INSERT INTO country(name, countryCode, latitude, longitude, populati
22	Lesoto	LS	-29.6099873	28.2336082	2.142.252	3.300	1	INSERT INTO country(name, countryCode, latitude, longitude, populati

Figura. 2. Preparación de datos

Finalmente se realizó la inserción de los datos en una base de datos MySQL. (Figura. 3).

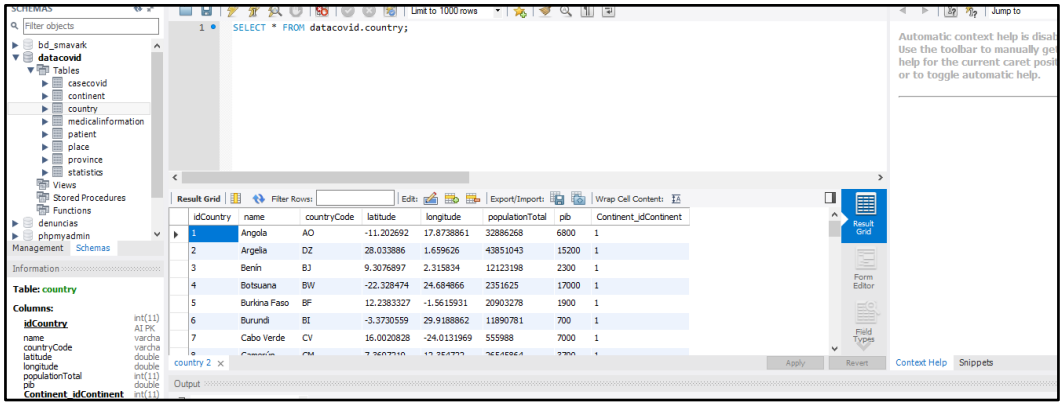


Figura. 3. Datos almacenados en la base de datos MySQL

2.2. Selección de patrones URI y definición de licencia

Para la definición de las diferentes URIs optamos por utilizar URIs significativos y barras URI (303-ASCII) como se menciona en la guía de referencia para realizar la especificación de URIs. Obteniendo así los siguientes elementos URI:

- Estructura base de URI: Para la iniciativa referente a la data de COVID-19 de Linked Data hemos definido el siguiente dominio.

  http://utpl.edu.ec/lod/dataCOVID/

- URI de TBox: Anexamos el nombre del concepto o de la propiedad a la estructura base del URI para incluir conceptos y propiedades disponibles en nuestra ontología.

  http://utpl.edu.ec/lod/dataCOVID/ontology/{concepto o propiedad}

- URI de ABox: Anexamos el nombre del recurso a la estructura base del URI para obtener la información de una instancia de un recurso.

  http://utpl.edu.ec/lod/dataCOVID/resource/{recurso}

Para poder identificar instancias de cada tipo de recurso, se utilizó las siguientes normas:

Clase	Identificador	Ejemplo
Continent	<Nombre del continente>	/África
Country	<Nombre del país>	/Angola
Province	<Nombre de la provincia>	/Luanda
Place	<Nombre del lugar>	/Adrar
Organization	<Nombre de la organización>	/MSPE

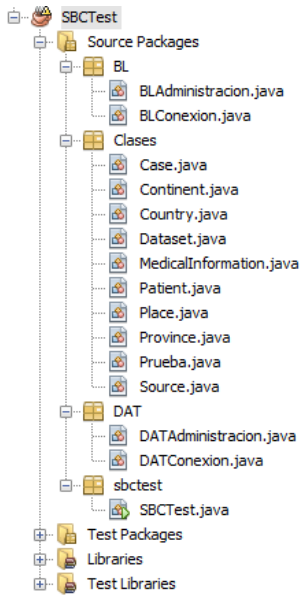
Catalog	<Nombre del catálogo>	/MSPE-covid19
Dataset	<Nombre del dataset>	/Covid19-Africa
MedicalInformation	<"Information-" + número de 6 dígitos>	/Information-000001
Patient	<"Patient-" + número de 6 dígitos>	/Patient-000001
CaseCovid	<"Case-" + número de 6 dígitos>	/Case-000001
Confirmed_cases	<"Confirmed-" + número de 6 dígitos >	/Confirmed-000001
Deaths_cases	<"Deaths-" + número de 6 dígitos >	/Deaths-000001
Recovered_cases	<"Recovered-" + número de 6 dígitos >	/Recovered-000001

El proyecto se desarrolló bajo licencia ***CREATIVE COMMONS*** puesto que nos permite la distribución de obras con derechos de autor. Cabe mencionar que las fuentes de datos que definimos para el proyecto cuentan con la misma licencia lo que permite reutilizar los recursos especificados.

Entre otras cosas, esta licencia permite que otros distribuyan, mezclen, adapten y desarrollen su trabajo, incluso comercialmente, siempre y cuando proporcionen el crédito por la creación original. Esta es la más complaciente de las licencias ofrecidas y es recomendado para la máxima difusión y uso de materiales con licencia.

2.3. Lógica de Transformación de datos con JENA

Para la transformación de los datos se aplicó una lógica y arquitectura en 3 capas, en la cual se encuentra inicialmente la capa de datos que se encarga de gestionar todo lo relativo a la base de datos y a la creación, edición y borrado de datos de ésta.



Luego se realiza la conexión con la siguiente capa que es la capa de negocio en donde se gestiona la lógica de la aplicación. Es donde se dice que se hace con los datos y esta capa estará conectada con la capa de persistencia para poder realizar sus funciones.

Y finalmente en la capa de presentación contamos con una clase en donde realizamos en base a las consultas de base de datos la transformación de estos a formato RDF con el framework JENA.

En la imagen presentada anteriormente en el paquete Clases, se encuentran varias clases .java creadas en base a nuestra ontología sobre datos COVID-19.

A continuación, se realizará la explicación breve del proceso que realizamos para llegar a la transformación de datos:

Primeramente, realizamos la conexión a la base de datos llamada ***datacovid*** como se muestra en la figura 4.

```
public Connection getConnection () throws ClassNotFoundException, SQLException{
    String driver="com.mysql.jdbc.Driver";
    String user="uoxrtksf3lziqgp2";
    String password="qeI2HNeoZj0Tl7cjrHRv";
    String url="jdbc:mysql://uoxrtksf3lziqgp2:qeI2HNeoZj0Tl7cjrHRv@bnyufdabsorwslj0olor-mysql.services.clever-cloud.com:20476
    Class.forName(driver) ;//Diver jdbc para trabajar con access
    con =DriverManager.getConnection(url,user,password);
    return con;//retorna la cioneccion url+ruta bd//retorna la cioneccion url+ruta bd
}
```

Figura 4. Conexión a la base de datos de MySQL

Luego realizamos una consulta multi tabla para obtener todos los datos relacionados a un caso de COVID-19 este proceso se muestra en la figura 5.

```
public ResultSet ConsultarCaso() throws ClassNotFoundException, SQLException{
    PreparedStatement pst = c.AbrirConexion().prepareStatement("SELECT * FROM casecovid cc, continent cont,
    ResultSet rs = pst.executeQuery();//recuper un un ResultSet y envio la varible a executeQuery
    return rs;
}
```

Figura 5. Consulta a la base de datos

Finalmente, como se muestra en la figura 6, procesamos el resultado de la consulta anterior instanciando los objetos de las clases .java para poder crear un objeto de un caso de covid y

almacenarlo en un arraylist del mismo tipo para posteriormente poder manipularlo y crear tripletas RDF en base al mismo.

```
public ArrayList<Case> consultarCaso() throws SQLException, ClassNotFoundException{
    ArrayList<Case> lstcases = new ArrayList<Case>();
    ArrayList<Source> lstsource = new ArrayList<Source>();
    ResultSet rs = mp.ConsultarCaso();
    while(rs.next()){

            //////Continent//////////
            int idContinent = rs.getInt("idContinent");
            String nameContinet = rs.getString("name");
            Continent continent = new Continent(idContinent, nameContinet);
            //////Country/////////////
            int idCountry = rs.getInt("idCountry");
            String nameCountry = rs.getString("name");
            String codeCountry = rs.getString("countryCode");
            double latitudeCountry = rs.getDouble("latitude");
            double longitudeCountry = rs.getDouble("longitude");
            int populationCountry = rs.getInt("populationTotal");
            double pibCountry = rs.getDouble("pib");
            Country country = new Country(idCountry, nameCountry, codeCountry, latitudeCountry,
            //////Province///////////////
            int idProvince = rs.getInt("idProvince");
            String nameProvince = rs.getString("nombre");
            Province province = new Province(idProvince, nameProvince, country);
            ///////Place//////////////////////
            int idPlace = rs.getInt("idPlace");
            String namePlace = rs.getString("name");
```

Figura 6. Procesamiento de consulta de base de datos

En el arraylist almacenamos temporalmente el resultado de la consulta de la base de datos anteriormente mencionada.

```
ArrayList<Case> lstcases = new ArrayList<Case>();
lstcases = manejador.consultarCaso();
```

Para creación de las diferentes propiedades que corresponden al modelo ontológico definimos los prefijos necesarios como se muestra en la figura 7.

```
//Set prefix for the URI base (data)
String dataPrefix = "http://utpl.edu.ec/lod/dataCOVID/";
model.setNsPrefix("data", dataPrefix);
//Vocab and models present in JENA
//SCHEMA
String schema = "http://schema.org/";
model.setNsPrefix("schema", schema);
Model schemaModel = ModelFactory.createDefaultModel();
//Dbpedia Ontology- DBO
String dbo = "http://dbpedia.org/ontology/";
model.setNsPrefix("dbo", dbo);
Model dboModel = ModelFactory.createDefaultModel();
//Geonames - gn
String gn = "http://www.geonames.org/ontology#";
model.setNsPrefix("gn", gn);
Model gnModel = ModelFactory.createDefaultModel();
//Dublincore - DBR
String dc = "http://purl.org/dc/elements/1.1/";
model.setNsPrefix("dc", dc);
Model dcModel = ModelFactory.createDefaultModel();
//DCat - dcat
String dcat = "http://www.w3.org/ns/dcat#";
model.setNsPrefix("dcat", dcat);
Model dcatModel = ModelFactory.createDefaultModel();
//Prov - prov
String prov = "http://www.w3.org/ns/prov#";
model.setNsPrefix("prov", prov);
Model provModel = ModelFactory.createDefaultModel();
//SIO - sio
String sio = "http://semanticscience.org/resource/";
model.setNsPrefix("sio", sio);
Model sioModel = ModelFactory.createDefaultModel();
//newOnto - newOnto
String newOnto = "http://utpl.edu.ec/lod/dataCOVID/ontology/";
model.setNsPrefix("newOnto", newOnto);
Model newOntoModel = ModelFactory.createDefaultModel();
```

Figura 7. Definición de prefijos

A continuación, se presenta la reutilización los métodos proporcionados por el docente para la creación de datos RDF con los datos del arraylist como se muestra a continuación en la figura 8.

```
for (Case ls : lstcases) {
    /*Informacion médica*/
    String estado = ls.getPatient().getMedicalInformation().getCurrentStatus().replaceAll(" ", "_");
    Resource rmi = model.createResource(dataPrefix + estado)
            .addProperty(RDF.type, newOntoModel.getProperty(newOnto, "Medical_Information"))
            .addProperty(newOntoModel.getProperty(newOnto, "date_first_symptom"), ls.getPatient().getMedicalInformation().getDateFirstSymptom())
            .addProperty(newOntoModel.getProperty(newOnto, "travel_history_date"), ls.getPatient().getMedicalInformation().getTravelHistoryDate())
            .addProperty(newOntoModel.getProperty(newOnto, "travel_history_location"), ls.getPatient().getMedicalInformation().getTravelHistoryLocation())
            .addProperty(dboModel.getProperty(dbo, "deathDate"), ls.getPatient().getMedicalInformation().getDeathDate())
            .addProperty(dboModel.getProperty(dbo, "currentStatus"), ls.getPatient().getMedicalInformation().getCurrentStatus());
    /*Informacion médica*/
    Resource pat = model.createResource(dataPrefix + ls.getPatient().getId())
            .addProperty(RDF.type, sioModel.getProperty(sio, "Patient"))
            .addProperty(FOAF.age, String.valueOf(ls.getPatient().getAge()))
            .addProperty(FOAF.gender, String.valueOf(ls.getPatient().getGender()))
            .addProperty(dboModel.getProperty(dbo, "place"), "");
```

Figura 8. Arraylist para creación de datos RDF

Al ejecutar el código anterior se generan las tripletas RDF de cada caso de COVID-19 con sus respectivos datos y relaciones como se muestra a continuación en la figura 9.

```
MODELO RDF------
<rdf:RDF
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ov="http://open.vocab.org/terms/"
    xmlns:prov="http://www.w3.org/ns/prov#"
    xmlns:newOnto="http://utpl.edu.ec/lod/dataCOVID/ontology/"
    xmlns:dbo="http://dbpedia.org/ontology/"
    xmlns:schema="http://schema.org/"
    xmlns:data="http://utpl.edu.ec/lod/dataCOVID/"
    xmlns:sio="http://semanticscience.org/resource/"
    xmlns:dcat="http://www.w3.org/ns/dcat#"
    xmlns:gn="http://www.geonames.org/ontology#"
    xmlns:j.0="http://xmlns.com/foaf/0.1/">
  <newOnto:CaseCovid rdf:about="http://utpl.edu.ec/lod/dataCOVID/Case-028376">
    <gn:locatedIn>
      <dbo:Place rdf:about="http://utpl.edu.ec/lod/dataCOVID/Bilqas_city">
        <dbo:grossDomesticProductNominalPerCapita>0.0</dbo:grossDomesticProductNominalPerCapita>
        <dbo:populationTotal>0</dbo:populationTotal>
        <schema:longitude>0.0</schema:longitude>
        <schema:latitude>0.0</schema:latitude>
        <gn:countryCode></gn:countryCode>
        <dbo:name>Bilqas</dbo:name>
      </dbo:Place>
    </gn:locatedIn>
    <prov:wasDerivedFrom>
      <dcat:Dataset rdf:about="http://utpl.edu.ec/lod/dataCOVID/line-list-egypt">
        <dcat:dateModification>2020-05-30</dcat:dateModification>
        <dc:dateSubmited>2020-03-01</dc:dateSubmited>
        <dcat:downloadURL>https://github.com/dsfsi/covid19africa/blob/master/data/line_lists/line-list-egypt.csv</dcat:downloadURL>
        <dbo:description></dbo:description>
        <dbo:name>line-list-egypt</dbo:name>
      </dcat:Dataset>
    </prov:wasDerivedFrom>
    <newOnto:hasData>
      <sio:Patient rdf:about="http://utpl.edu.ec/lod/dataCOVID/Patient-028376">
        <newOnto:hasData>
          <newOnto:Medical_Information rdf:about="http://utpl.edu.ec/lod/dataCOVID/">
            <newOnto:travel_history_location>Cameroon</newOnto:travel_history_location>
            <newOnto:travel_history_location>France, Italy</newOnto:travel_history_location>
            <newOnto:travel_history_location></newOnto:travel_history_location>
            <newOnto:travel_history_date>2020-03-03</newOnto:travel_history_date>
            <newOnto:travel_history_date>2020-03-06</newOnto:travel_history_date>
            <newOnto:travel_history_date>2020-02-17</newOnto:travel_history_date>
            <newOnto:travel_history_location>Spain</newOnto:travel_history_location>
            <newOnto:travel_history_date>null</newOnto:travel_history_date>
            <newOnto:date_first_symptom>null</newOnto:date_first_symptom>
            <newOnto:date_first_symptom>2020-03-15</newOnto:date_first_symptom>
            <newOnto:travel_history_location>France</newOnto:travel_history_location>
            <newOnto:travel_history_date>2020-03-01</newOnto:travel_history_date>
            <dbo:currentStatus></dbo:currentStatus>
            <newOnto:travel_history_location>Italy</newOnto:travel_history_location>
            <newOnto:travel_history_date>2020-03-14</newOnto:travel_history_date>
```

Figura 9. Resultado de la ejecución del código para la generación de datos RDF

2.4. Resultados de la transformación

A continuación, se presenta el número de instancias que se generaron a partir de los datos recolectados para cada clase del modelo ontológico creado.

Clase	Número de instancias
Continent	1
Country	54
Province	123
Place	150
Organization	1
Catalog	2
Dataset	47
MedicalInformation	69023

Patient	69023
CaseCovid	69023
Confirmed_cases	3388
Deaths_cases	1276
Recovered_cases	2263

2.5. Elección de repositorio de almacenamiento

Elegimos **VIRTUOSO** porque tiene una capacidad de virtualización de datos que permite la construcción e implementación de grafos de conocimiento sobre datos existentes expuestos por APIS como HTTP, ODBC, JDBC, entre otros.

También cuenta con una base de datos MySQL que permite almacenar información para la creación de diferentes grafos. Además, tiene la función de importar y almacenar datos RDF directamente para la posterior utilización en forma de grafo.

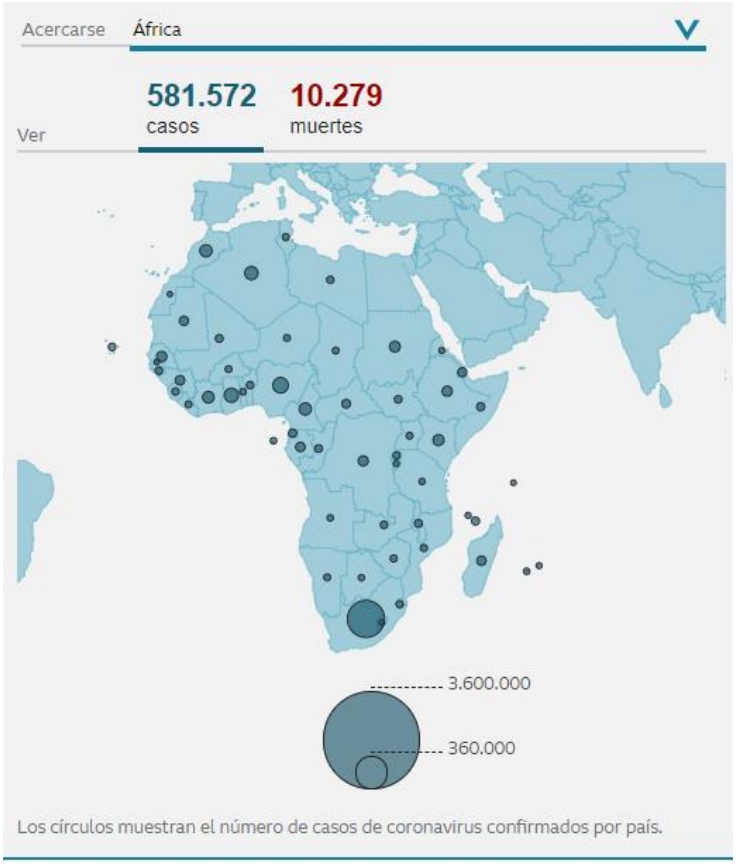
Virtuoso ofrece el servicio de ***Virtuoso Conductor*** que facilita la administración de la base de datos, de los grafos, de las vistas para grafos, tablas, procedimientos almacenados, copias de seguridad, consultas Sparql, administración de dominios locales y directorios.

3. Aplicación

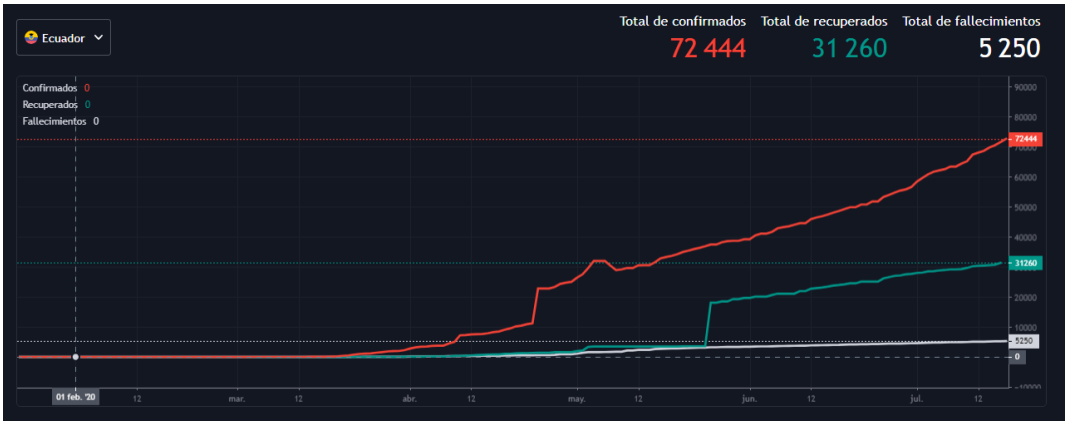
3.1. Trabajos relacionados

CORONAVIRUS MAPEADO: Los casos confirmados por país son representados en un mapa mediante círculos de tamaño proporcional al número de casos confirmados.

Este trabajo fue desarrollado por la Universidad Johns Hopkins (Baltimore, EE.UU.), y autoridades locales. La última actualización de estas cifras se realizó el 3 de julio de 2020.



GRÁFICOS Y ESTADÍSTICAS DEL CORONAVIRUS (COVID - 19): Representa mediante una línea de tiempo la variación de las cifras respecto a los casos confirmados, recuperados y fallecidos a nivel mundial y por país.



3.2. Herramientas utilizadas

Virtuoso: Virtuoso Universal Server es un híbrido de middleware y motor de base de datos que combina la funcionalidad de un sistema tradicional de gestión de bases de datos relacionales (RDBMS), base de datos relacional de objetos (ORDBMS), base de datos virtual,

RDF , XML , texto libre , servidor de aplicaciones web y servidor de archivos funcionalidad en un solo sistema.

Spring Boot: Spring es un framework alternativo al stack de tecnologías estándar en aplicaciones JavaEE.

Dependencia Jena de Spring Boot: Apache Maven es una herramienta para ayudar a los proyectos Java a administrar sus dependencias en el código de la biblioteca, como Jena. Al declarar una dependencia en el núcleo de Jena en el pom.xml archivo de su proyecto, obtendrá también el conjunto consistente de archivos de biblioteca de los que Jena depende.

Angular: Angular es un framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (Single Page Application).

3.3. Consultas Sparql

Se realizaron las siguientes consultas con el fin de dar una respuesta a las preguntas establecidas en la definición del proyecto.

Consulta 1: ¿Qué países cuentan con el mayor número de casos confirmados con covid-19 hasta la última fecha de actualización de datos?

```
PREFIX newOnto:<http://utpl.edu.ec/lod/dataCOVID/ontology/>
PREFIX gn:<http://www.geonames.org/ontology#>
PREFIX schema:<http://schema.org/>
SELECT ?nombre ?cantidad WHERE {
 ?var rdf:type newOnto:Confirmed_Cases ;
 newOnto:totalQuantity ?cantidad ;
 gn:locatedIn ?pais;
 schema:observationDate ?fecha.
 ?pais dbo:name ?nombre.
 FILTER (?fecha = "2020-06-15")
}ORDER BY DESC (xsd:integer(?cantidad))
```

Resultado de la consulta 1:

nombre	cantidad
"Sudáfrica"	"73533"
"Egipto"	"46289"
"Nigeria"	"16658"

Consulta 2: ¿Cuál es el género con mayor número de casos de covid-19?

```
PREFIX sio:<http://semanticscience.org/resource/>
SELECT (sum(if($gender="male",1,0)) as ?hombres)
(sum(if($gender="female",1,0)) as ?mujeres)
{
        {SELECT DISTINCT ?person $gender WHERE {
                ?person rdf:type sio:Patient;
                foaf:gender $gender
        } GROUP BY ?person}
}
```

Resultado de la consulta 2:

hombres	mujeres
300	162

Consulta 3: ¿Qué países presentan el mayor índice de casos fallecidos por covid-19?

```
PREFIX newOnto:<http://utpl.edu.ec/lod/dataCOVID/ontology/>
PREFIX gn:<http://www.geonames.org/ontology#>
PREFIX schema:<http://schema.org/>

SELECT ?cantidad ?nombre  WHERE {
 ?var rdf:type newOnto:Deaths_Cases ;
```

```
 newOnto:totalQuantity ?cantidad ;
 gn:locatedIn ?pais;
 schema:observationDate ?fecha.
 ?pais dbo:name ?nombre.
 FILTER (?fecha = "2020-06-15")
}ORDER BY DESC (xsd:integer(?cantidad))
```

Resultado de la consulta 3:

cantidad	nombre
"1672"	"Egipto"
"1568"	"Sudáfrica"
"777"	"Argelia"
"424"	"Nigeria"
"277"	"Camerún"
"104"	"Kenia"
"91"	"Mauritania"

Consulta 4: ¿Qué países visitaron las personas contagiadas con COVID-19 antes de ingresar al continente africano?

```
PREFIX newOnto:<http://utpl.edu.ec/lod/dataCOVID/ontology/>
PREFIX gn:<http://www.geonames.org/ontology#>
PREFIX schema:<http://schema.org/>

SELECT ?case ?location WHERE {
 ?case rdf:type newOnto:CaseCovid;
 newOnto:hasData ?paciente.
 ?paciente newOnto:hasData ?informacion.
 ?informacion newOnto:travel_history_location ?location.
 }
```

Resultado de la consulta 4:

case	location
http://utpl.edu.ec/lod/dataCOVID/Case-050889	"Norway"
http://utpl.edu.ec/lod/dataCOVID/Case-050890	"UAE"
http://utpl.edu.ec/lod/dataCOVID/Case-050891	"UAE"
http://utpl.edu.ec/lod/dataCOVID/Case-050892	"UAE"

Consulta 5: ¿Cuál el nombre de los países, su longitud, su latitud, su población y el número de casos de covid-19?

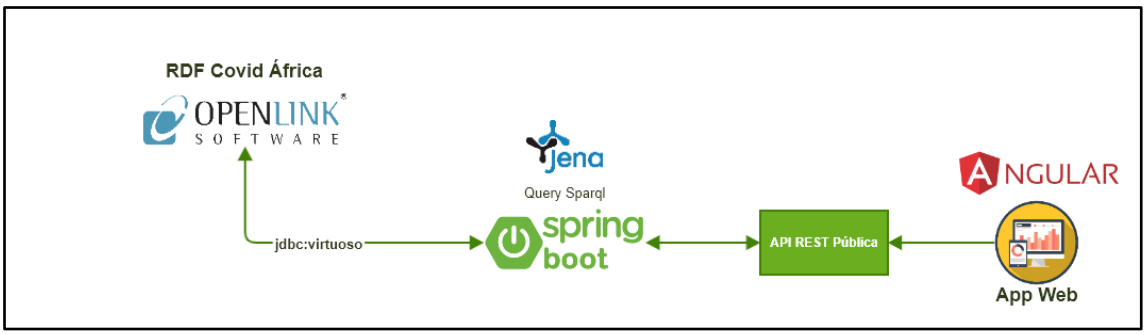
```
PREFIX newOnto:<http://utpl.edu.ec/lod/dataCOVID/ontology/>
PREFIX gn:<http://www.geonames.org/ontology#>
PREFIX schema:<http://schema.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?nombre ?latitud ?longitud ?cantidad ?poblacion ?fecha WHERE {
 ?var rdf:type newOnto:Confirmed_Cases ;
 newOnto:totalQuantity ?cantidad ;
 gn:locatedIn ?pais;
 schema:observationDate ?fecha.
 ?pais dbo:name ?nombre;
 schema:latitude ?latitud;
 schema:longitude ?longitud;
 dbo:populationTotal ?poblacion.
 FILTER (?fecha = "2020-06-15")
}ORDER BY DESC (xsd:integer(?cantidad))
```

Resultado de la consulta 5:

nombre	latitud	longitud	cantidad	poblacion	fecha
"Sudáfrica"	"-30.559482"	"22.937506"	"73533"	"59308690"	2020-06-15
"Egipto"	"26.820553"	"30.802498"	"46289"	"102334403"	2020-06-15
"Nigeria"	"9.081999"	"8.675277"	"16658"	"206139587"	2020-06-15
"Argelia"	"28.033886"	"1.659626"	"11031"	"43851043"	2020-06-15
"Camerún"	"7.369722"	"12.354722"	"10140"	"26545864"	2020-06-15

3.4. Lógica de la app

A continuación, se describe la arquitectura de la aplicación:



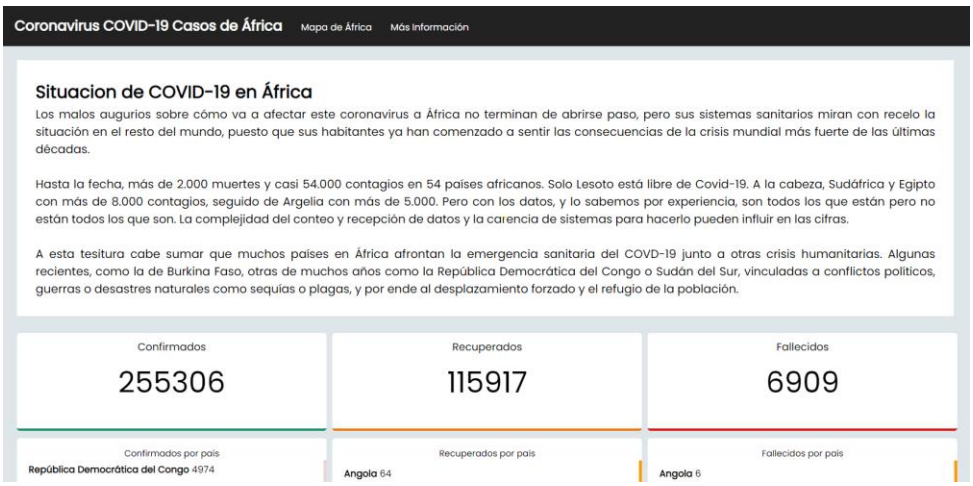
En un principio se cuenta con el rdf (Covid-19 África) generado en Jena en la fase anterior del proyecto. Este rdf se encuentra almacenado en Virtuoso OpenLink Software y se lo consume desde Spring Boot (Framework JAVA) mediante conexión con el jdbc:virtuoso para realizar consultas Sparql para la posterior exposición de los resultados de esta mediante una API REST Pública para finalmente ser consumida desde cualquier aplicación.

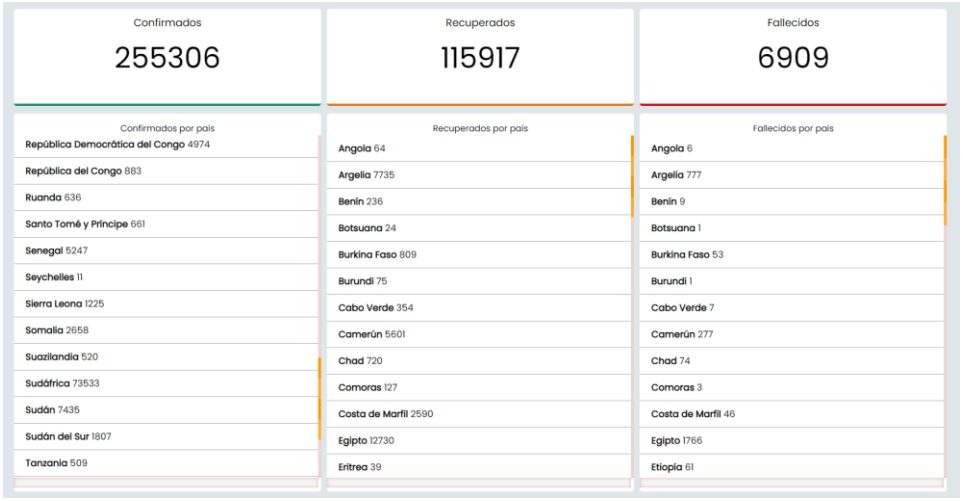
La API generada con los resultados de una consulta Sparql tiene la siguiente estructura:

```
[
    {
        "nombre": "Sudáfrica",
        "latitud": -305.594,
        "longitud": 229.375,
        "cantidad": 73533,
        "poblacion": 59308,
        "fecha": "2020-06-15"
    },
    {
        "nombre": "Egipto",
        "latitud": 268.205,
        "longitud": 308.024,
        "cantidad": 46289,
        "poblacion": 102334,
        "fecha": "2020-06-15"
    },
    {
        "nombre": "Nigeria",
        "latitud": 90.819,
        "longitud": 86.752,
        "cantidad": 16658,
        "poblacion": 206139,
        "fecha": "2020-06-15"
    },
    {
        "nombre": "Argelia",
        "latitud": 28.033,
        "longitud": 1.659,
        "cantidad": 11031,
        "poblacion": 43851,
        "fecha": "2020-06-15"
    },
```


Las APIs generadas se las consumen desde el Front-end de la aplicación:

En la página inicial de la aplicación se puede visualizar una breve descripción de la situación de COVID-19 en África, seguido del número de casos confirmados, recuperados y fallecidos y tablas en donde se muestra estos datos por país.





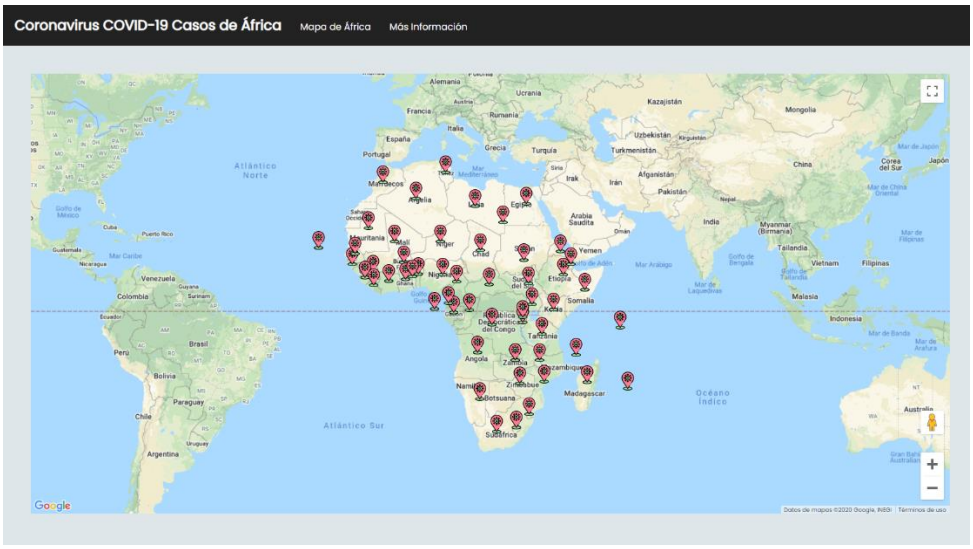
Confirmados	Recuperados	Fallecidos
255306	115917	6909

Confirmados por país	Recuperados por país	Fallecidos por país
República Democrática del Congo 4974	Angola 64	Angola 6
República del Congo 883	Argelia 7735	Argelia 777
Ruanda 636	Benín 236	Benín 9
Santo Tomé y Príncipe 661	Botsuana 24	Botsuana 1
Senegal 5247	Burkina Faso 809	Burkina Faso 53
Seychelles 11	Burundi 75	Burundi 1
Sierra Leona 1225	Cabo Verde 354	Cabo Verde 7
Somalia 2658	Camerún 5601	Camerún 277
Suazilandia 520	Chad 720	Chad 74
Sudáfrica 73533	Comoras 127	Comoras 3
Sudán 7435	Costa de Marfil 2590	Costa de Marfil 46
Sudán del Sur 1807	Egipto 12730	Egipto 1766
Tanzania 509	Eritrea 39	Etiopía 61

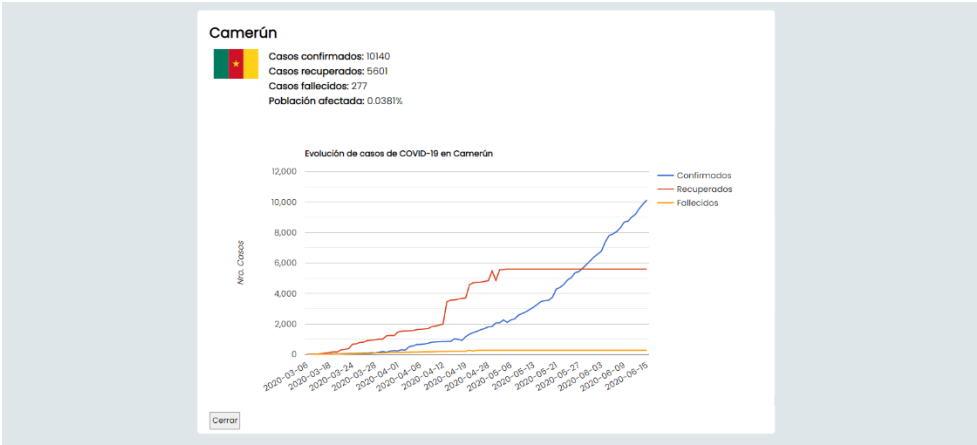
En la siguiente gráfica de barras se puede visualizar el número de casos por paí.

En la siguiente gráfica se muestra los sitios de donde ingresaron las personas con COVID-19.

En el mapa se graficaron indicadores en cada uno de los países afectados por COVID-19 en el continente africano.



Al presionar un marcador del mapa se no abre una ventana en donde podemos visualizar información sobre dicho país: casos confirmados, recuperados, fallecidos, porcentaje de la población afectada y una línea de tiempo en donde se puede visualizar la evolución diaria de la enfermedad en ese país.



4. Conclusiones

El proyecto de generación de datos RDF nos permite evidenciar todo el proceso realizado para llegar a la creación de la aplicación web que contiene todos los datos recopilados y almacenados en el repositorio de datos especificado con anterioridad.

Después de la realización del actual proyecto llegamos a la conclusión de que el proceso realizado con todas la etapas y características aplicadas a cada una de ellas nos permite en cierta parte alcanzar el objetivo de la web semántica de la publicación de datos legibles para aplicaciones informáticas como en nuestro caso la publicación y realización de la aplicación de visualización de datos estadísticos sobre los casos de COVID-19 en el continente africano.