****

**Universidad Técnica Particular de Loja**

**Integrantes:** Ramiro Vivanco, Cecilia Sánchez

Web Semántica y ODS

**Introducción**

El Internet y las tecnologías de la información han facilitado la manera en la que buscamos información y algo que parece un proceso muy simple para los usuarios conlleva varios procesos complejos que hacen que toda la información que encontramos en internet tenga sentido y ahí es donde entra la web semántica de datos.

Los objetivos de desarrollo sostenible representan el progreso de muchos países en vías de desarrollo para tener un futuro mejor. En este documento detallamos el proceso de transformación de información bibliográfica referente a los objetivos de desarrollo sostenible a información legible y que puede ser implementada en la web semántica de datos para futuros trabajos.

**Preparación de datos en origen**

El primer paso en nuestro trabajo consistió en elegir la forma en la que vamos a almacenar los datos temporalmente para ser utilizados con la librería de Jena y para ello decidimos trabajan con una base de datos mysql para ello creamos un modelo de base de datos similar al modelo Scholar proporcionado para la creación del archivo RDF.

Para recolectar la información bibliográfica de los documentos relacionados a los objetivos de desarrollo sostenible elegimos las fuentes de datos google scholar y principalmente **scopus** la cual nos permite mediante su plataforma extraer información dependiendo de los parámetros que se quiera buscar.

La información de las fuentes de datos se descarga en formato csv lo cual requiere un procesamiento para poder ser utilizada posteriormente por ello utilizamos Excel para trabajar los datos eliminando comas, caracteres especiales y utilizando fórmulas que nos permiten eliminar cualquier anomalía que pueda representar un error al momento de generar las tripletas RDF.

Una vez tratados los datos el siguiente paso fue guardar los datos en formato csv con el nombre de la tabla para luego cargar el archivo en la pagina [www.convertcsv.com/csv-to-sql.htm](https://www.convertcsv.com/csv-to-sql.htm) la cual nos permite generar sentencias sql a partir de un archivo csv, posteriormente ejecutamos las sentencias en la base de datos.

**Definición de URIs**

Para la definición de las uris utilizamos el prefijo http://utpl.edu.ec/sbc/ o data: y luego buscamos un o varios atributos que sean representativos de cada instancia y lo unimos al prefijo por ejemplo data:Lowering-barriers-plant

**Transformación y almacenamiento de datos RDF**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabla resumen de datos recolectados** | | | |
| **Clase** | **Instancias** | **Clase** | **Instancias** |
| bibo:Document | 484 | foaf:Organization | 424 |
| foaf:Person | 11671 | vivo:Place | 1 |
| c4o:GlobalCitation Count | 218 | bibo:DocumentPart | 11 |
| bibo:Book | 15 | bibo:BookSection | - |
| skos:Concept | 2430 | vivo:ConferencePaper | 47 |
| bibo:Article | 46 | vivo:Publisher | 66 |
| bibo:Journal | 49 | bibo:Thesis | 50 |
| bibo:Conference | 32 | bibo:Proceedings | 48 |

**Pre-procesamiento de datos**

Para el manejo de datos se tomaron algunas consideraciones:

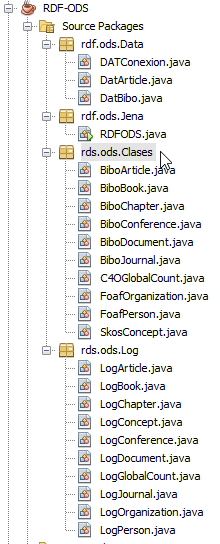
* Se eliminaron comas de los títulos ya que complicaba el manejo de los archivos csv.
* Para saber si un documento era un artículo, libro, chapterbook, journal of conference se descargó una información adicional que fue el tipo de documento el cual nos ayuda a filtrar los datos correspondientes a cada tipo
* El manejo de los datos para las clases Person y Concept fueron los más complejos de hacer ya que los cuales venían seguidos de punto y coma los cuales tuvimos que separarlo en celdas horizontales y luego mediante la herramienta KUTOOLS para excel se pudo cambiar su orientación, pero se obtuvieron muchas columnas.

|  |  |
| --- | --- |
| palabra1 | 1 |
| palabra2 | 1 |
| palabra4 | 2 |
| palabra1 | 2 |
| palabra3 | 3 |
| palabra5 | 3 |

* Se buscó alguna fórmula para que nos ayude con la organización de las mismas, pero no tuvimos éxito ya que no podíamos saber a que idDocument pertenecía cada celda por los que se tuvo que copiar columna por columna hasta tener una sola con el id del documento correspondiente

Datos reorientados

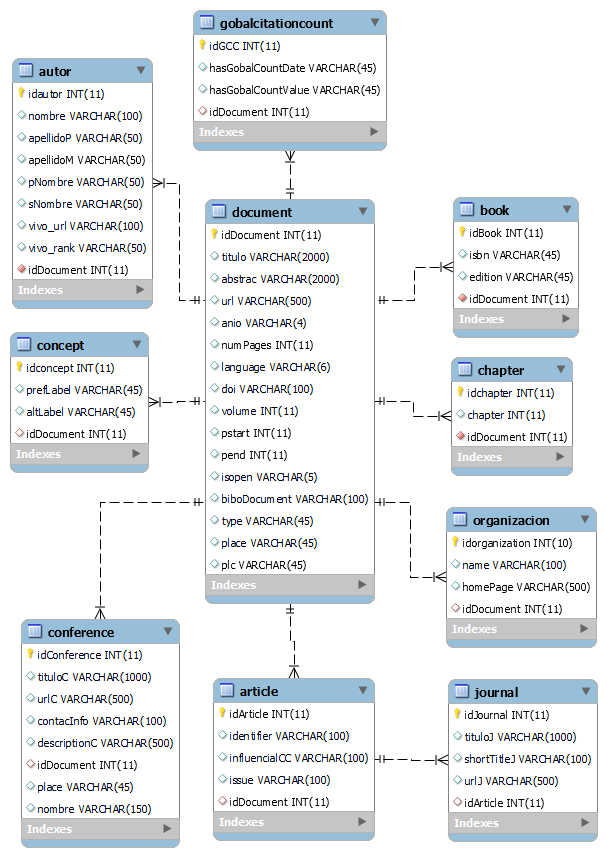
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| palabra1 | 1 | palabra4 | 2 | palabra3 | 3 |
| palabra2 | 1 | palabra1 | 2 | palabra5 | 3 |

**Transformación de datos**

Para la transformación de los datos creamos una arquitectura de 4 capas en la cual esta compuesta por una capa de clases en la cual almacenamos todas las clases referentes a las clases del modelo proporcionado. También tenemos una capa de datos la cual nos permite conectarnos a la base de datos e implementar las consultas.

La capa de lógica nos permite asignar los datos recolectados a un array de objetos y la capa Jananos permite:

* Implementar la lógica para crear los prefijos, los modelos y las propiedades
* Llamar a las consultas a la base de datos y almacenarlos en un array list.
* Crear las propiedades del modelo usando los datos almacenados en los objetos.
* Imprimir el modelo con su propiedad es e instancias.



**Almacenamiento**

Utilizamos una base de datos MySQL ya que nos permite relacionar tablas entre lo cual es útil al momento de trabajar con Jena ya que podemos implementar la lógica para conectarnos a la misma y obtener datos.

Las tablas del modelo son las siguientes:

* Document (titulo,abstrc,anio,url,numpages, doi,languaje,volumen,pstart,pend, is\_open)
* Article (identifier, influenciaccc, issiue)
* Conference (titulo, url, contacinfo, descriopcionm, place)
* Person (nombre, apellido, url , rank)
* GCC (preflabel, altlabel)
* Book (isbn, edition)
* Chapter(chapter)
* Organization (name, homepage)
* Journal (titulo, shortTitle,url)

**Enlazado post-transformación**

Para el enlazado se trabajo en un archivo de Python el cual nos permite enlazar los títulos de los documentos bibliográficos a dbpedia. Su funcionamiento es el siguiente:

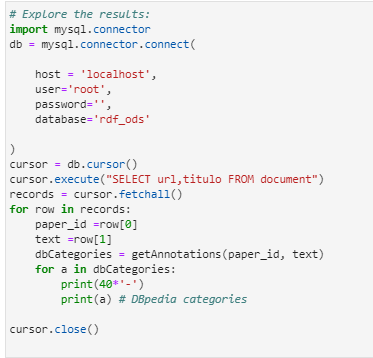
Importamos los enpoints a dbpedia



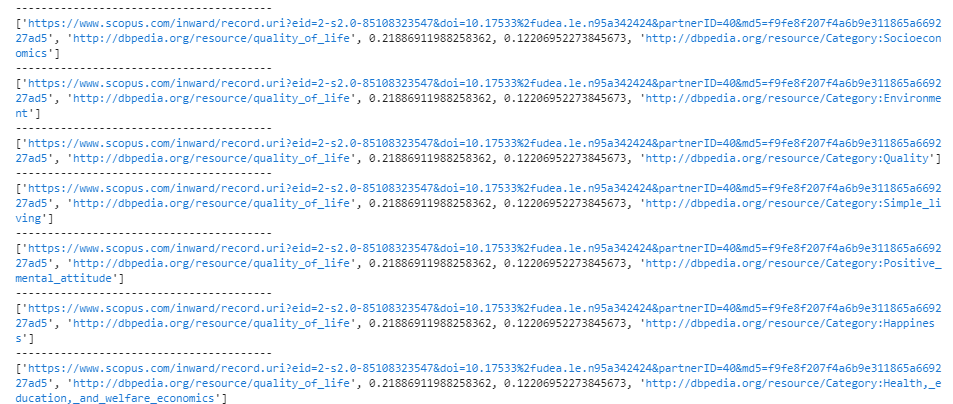
Definimos la función annotations para obtener los enlaces de los datos



Consultamos los datos y lo recorremos, luego enviamos los valores al método get\_Annotations para que obtenga los valore y finalmente imprimimos las tripletas para ese documento.

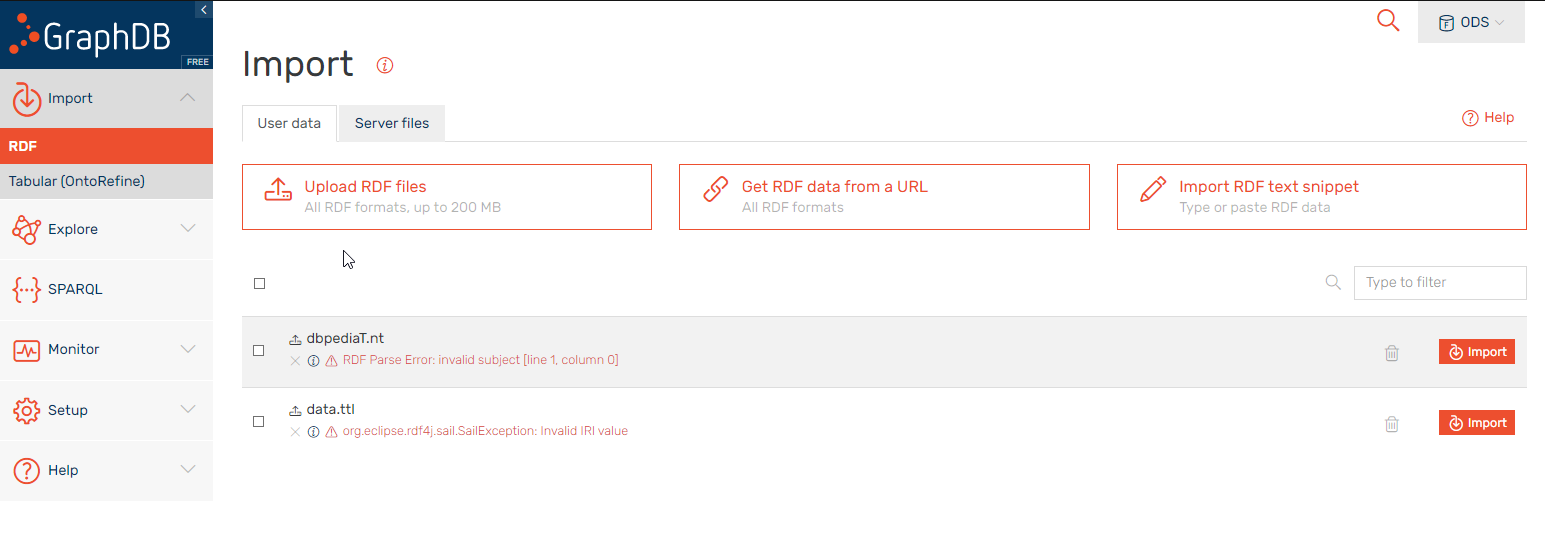


Obtenemos lo valores resultantes

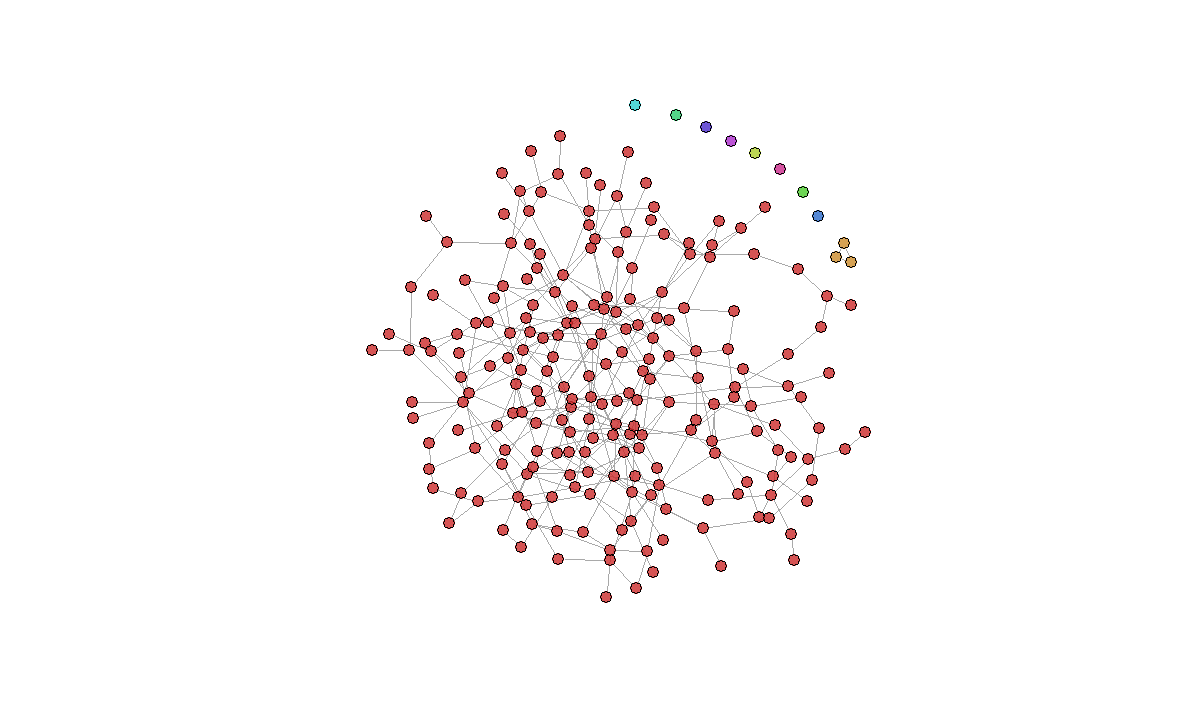


**Implementación de consultas SPARQL que ilustren el valor de los datos enlazados.**

Utilizamos GraphDB para almacenar el grafo de conocimiento y luego implementar consultas.

**

**Implementar la aplicación**



**Describir el framework base**

IGraph es una colección de bibliotecas para crear y manipular gráfos y analizar redes. Está escrito en C y también existe como paquetes Python y R.

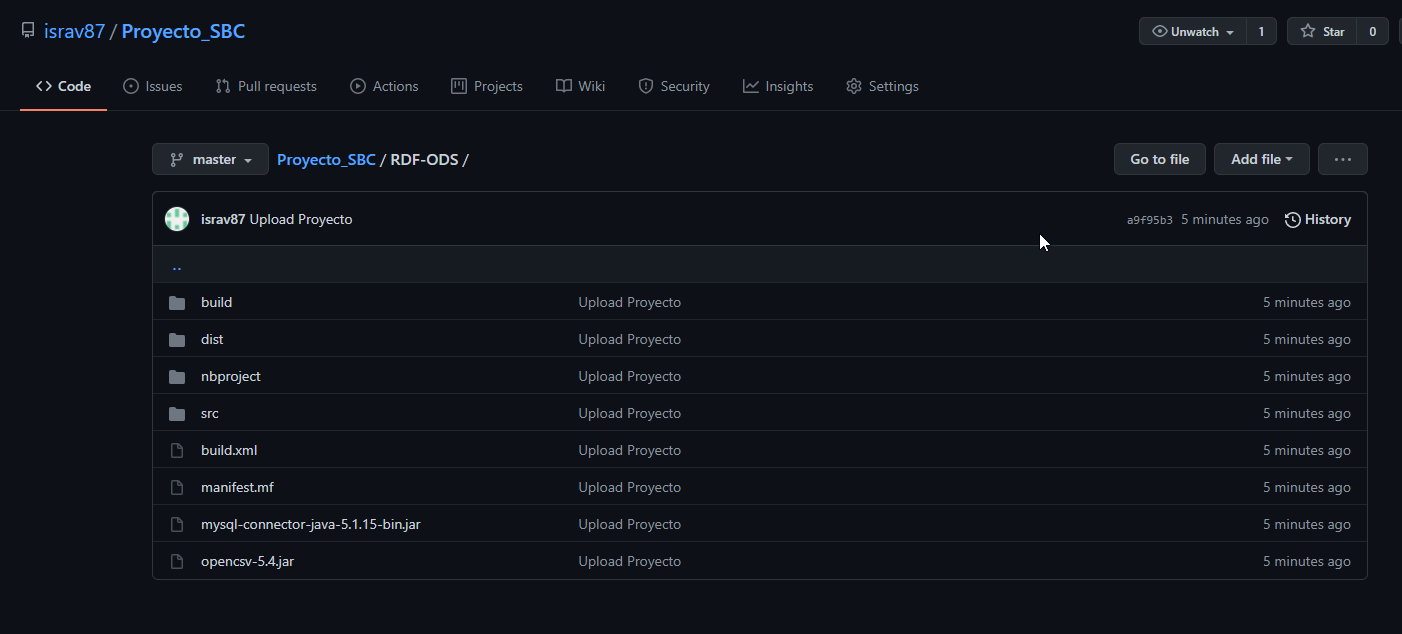
Además, existe una interfaz para Mathematica. El software se utiliza ampliamente en la investigación académica en ciencia de redes y campos relacionados.

Hay varios paquetes de software de código abierto que utilizan funciones de igraph y para nuestra aplicación usaremos el paquete de Python.

**Arquitectura o pipeline diseñado**

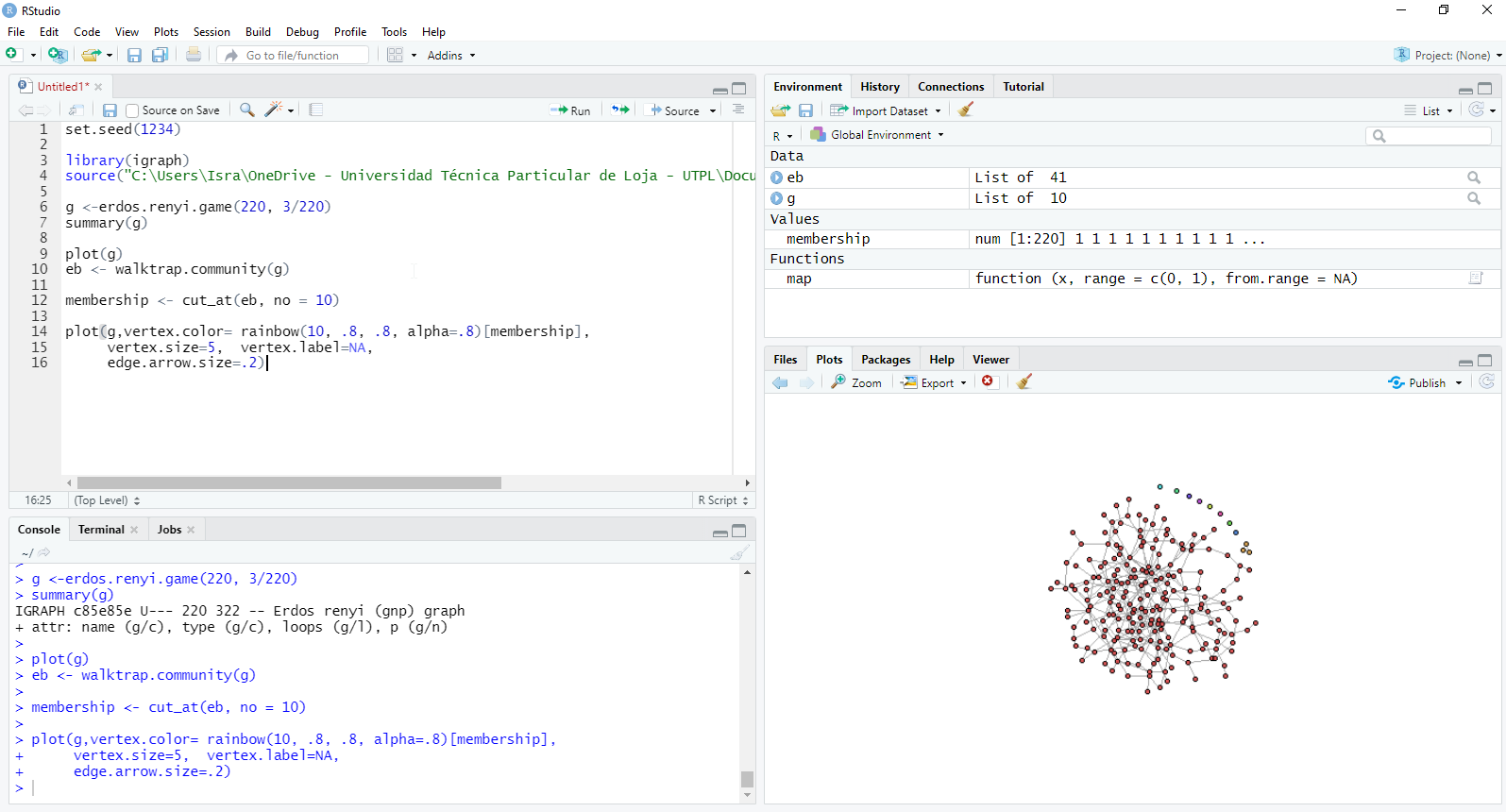
**Configuraciones realizadas**

Se creó un repositorio en GitHub (<https://github.com/israv87/Proyecto_SBC.git> ) donde se suben continuamente los avances del pro (*Pendiente de finalización)*

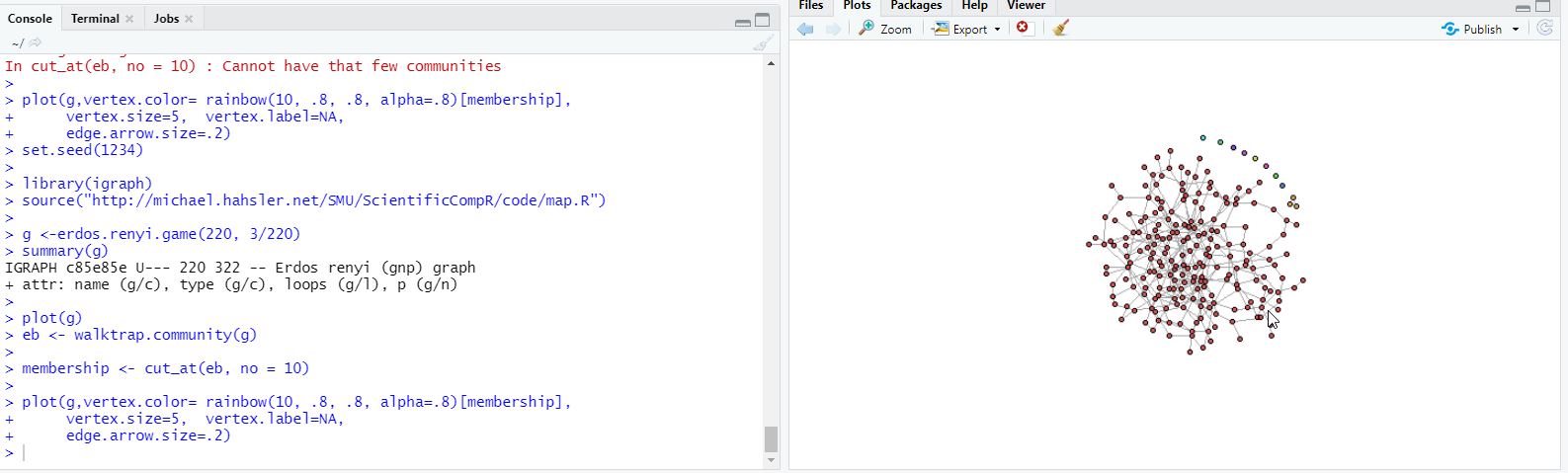
**

**Componentes y lógica de implementación.**

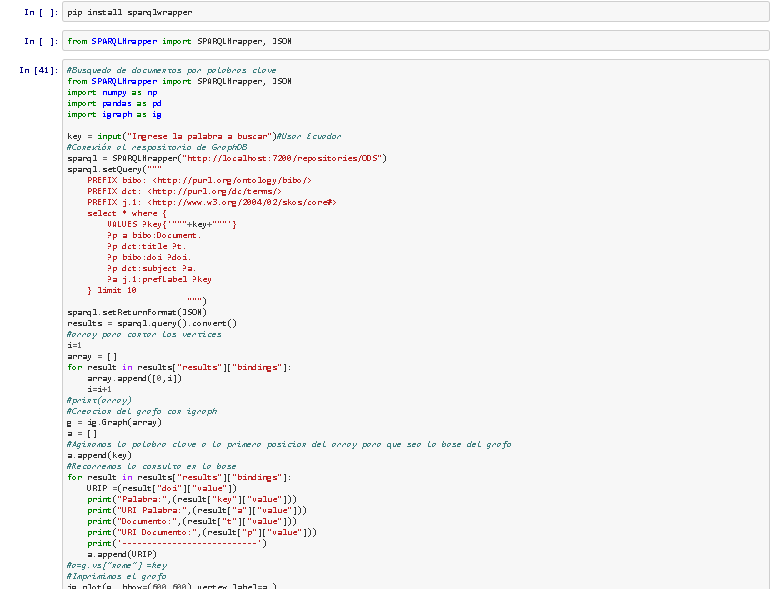
Para el desarrollo de la aplicación se trabaja con la librería de iGraph en R Studio donde se cargan los datos de la base de grafo y se implementan las funciones de la librería para dibujar los grafos y encontrar patrones

****Mediante algunos casos de uso, ilustrar el funcionamiento de la app y de los datos enlazados.**

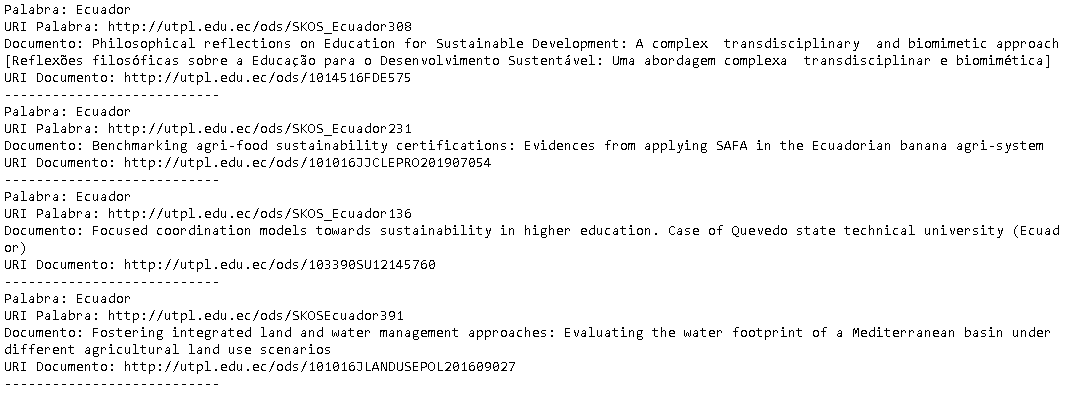
Nuestra aplicación funcionará dentro del entorno de Rstudio el cual nos permitirá trabajar con las librerías de iGraph y presentar resultados mediante la consola y el visualizador.

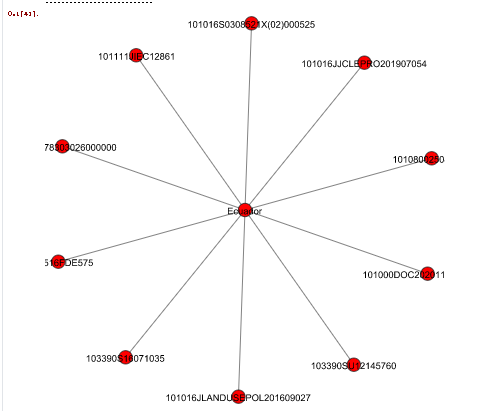


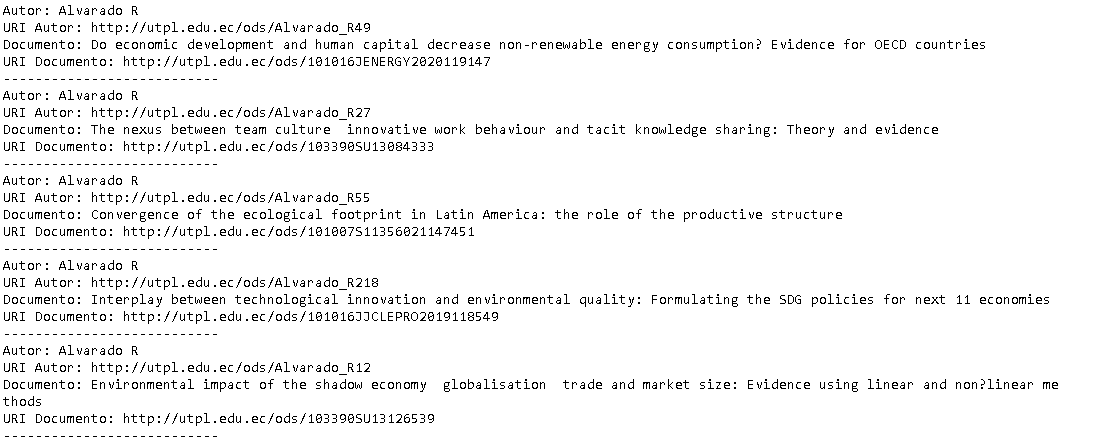
**Aplicación implementada en Python**

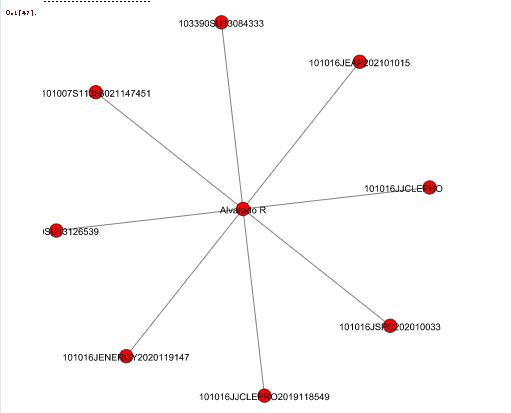
****

**Grafos Resultantes**

****

****

****

****