

DESARROLLO DE UNA APLICACIÓN DE OBTENCIÓN Y ANÁLISIS DE DATOS MODELADOS COMO GRAFOS

Trabajo Fin de Máster

Universidad Rey Juan Carlos



Alumna: Candela Vidal Rodríguez

Directores: Belén Vela Sánchez y
Jesús Sánchez-Oro Calvo

RESUMEN

En esta memoria se describe en detalle el proceso llevado a cabo para extraer datos abiertos sobre asilo humanitario de la plataforma Humanitarian Data Exchange (en adelante referida como HDX), la transformación de los mismos utilizando python, su carga en Neo4j y el posterior análisis utilizando consultas Cypher y algoritmos implementados como procedimientos de la base de datos orientada a grafos. Finalmente, los resultados serán presentados utilizando neovis.js como soporte para la visualización.

La estructura del trabajo consta de los siguientes apartados:

- Introducción. Se aclara la motivación detrás de este trabajo y se precisan los hitos de los que se conforma el proyecto.
- Análisis del Proceso. Se explican cada uno de los pasos que se han seguido en el desarrollo de la aplicación.
- Arquitectura. Se describe de manera pormenorizada todas las tecnologías utilizadas.
- Datos. Se define el modelo de la base de datos y se interpretan los datos almacenados.
- Algoritmos. Se justifican los algoritmos utilizados sobre el grafo.
- Resultados. Se muestra el producto de los algoritmos aplicados a los datos.
- Visualización. Se presenta un evolución temporal del grafo utilizando neovis.js como herramienta de visualización.
- Mejoras Futuras. Se proponen líneas de trabajo futuras que puedan mejorar la aplicación.
- Conclusiones. Se evalúa la consecución de los hitos definidos previamente.

RESUMEN	1
2 INTRODUCCIÓN	3
2.1 Contexto	3
2.2 Objetivos	5
3 ANÁLISIS DEL PROCESO	7
3.1 Extracción de datos	7
3.2 Procesamiento de datos	8
3.3 Ingesta en base de datos	10
3.4 Análisis de datos	12
3.5 Algoritmos	13
3.6 Visualización	14
4 ARQUITECTURA	15
4.1 Contenedores Docker orquestados con Docker Compose	15
4.1.1 Imagen Docker con Neo4j	17
4.1.2 Imagen Docker con Jupyter Notebook	18
4.2 Control de Versiones	20
4.2.1 Código	20
4.2.2 Datos	21
4.3 Ejecución en la Nube	22
4.4 Automatización de comandos	24
5 DATOS	25
5.1 Datos Originales	25
5.1.1 UNHCR's Population of concern	25
5.1.2 World Bank Indicators	26
5.2 Modelo de Base de Datos	27
5.3 Análisis exploratorio de Datos	28
6 ALGORITMOS	35
6.1 Centralidad	35
6.2 Detección de Comunidades	36
7 RESULTADOS	37
7.1 Degree Centrality	37
7.2 Betweenness Centrality	38
7.3 Strongly Connected Components	38
7.4 Louvain	39
8 VISUALIZACIÓN	41
9 MEJORAS FUTURAS	42
10 CONCLUSIONES	43
11 ANEXO I. Servidor nginx para Neovis.js	44
12 ANEXO II. Pasos para Ejecutar la Aplicación en Google Cloud	45
13 REFERENCIAS	46
14 RELACIÓN DE FIGURAS	48



2 INTRODUCCIÓN

2.1 Contexto

El presente proyecto se desarrolla con el fin de mostrar los conocimientos adquiridos en el Máster en Data Science y el conjunto de habilidades requeridas para todo científico de datos. Dichas habilidades son, según la certificación en Data Science de Microsoft Azure [1]:

- Definir y preparar el entorno de desarrollo.
- Preparar los datos para su modelado.
- Seleccionar adecuadamente las variables a tener en cuenta.
- Desarrollar modelos.

Como tal, se pretende obtener, transformar y analizar un conjunto de datos de manera programática, así como evaluar las tecnologías actuales que más se ajusten al mismo, implementando finalmente una arquitectura robusta que permita desarrollar de manera sencilla las tareas del Pipeline de datos [2].



Fig. 1 PIPELINE de DATOS

Se opta por elegir un conjunto de datos abierto, tanto por la finalidad educativa de este trabajo, como por los tres principios fundamentales que definen **Open Data** [3]:

- Disponibilidad y acceso.
- Reutilización y redistribución.
- Participación universal.

Se facilita por tanto el acceso y la disponibilidad a dichos datos, además de permitir explotarlos de forma libre sin restricciones pues cualquier persona interesada tiene derecho a consumirlos.

Además, para otorgar de un mayor sentido a la aplicación a desarrollar se decide escoger un tema enfocado al **Data Science for Social Good** [4]. Es decir, se elige un problema real con impacto social sin estar motivado necesariamente por la obtención de beneficios económicos.

Por lo general este tipo de proyectos van de la mano de gobiernos u ONGs y es por ello que se determina extraer los datos de una (o varias) fuentes cuya procedencia sea un organismo oficial gubernamental o sin ánimo de lucro.

Tras explorar varias plataformas de datos que reúnen las características fijadas previamente (datos abiertos con fines sociales), entra en juego **Humanitarian Data Exchange** [5], una plataforma creada en 2014 para compartir datos a través de organizaciones y crisis.

HDX es parte de la oficina de las Naciones Unidas para la Coordinación de Asuntos Humanitarios y tiene como meta reunir a los diferentes actores humanitarios para asegurar una respuesta coherente ante emergencias, además de asegurar un marco común donde todas las partes pueden contribuir al esfuerzo global.

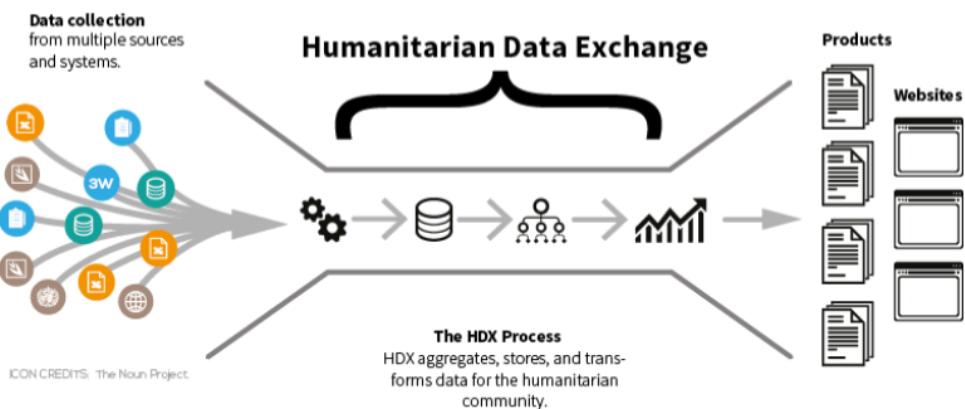


Fig. 2 Proceso HDX
<http://www.kalisch.biz/projects/humanitarian-data-exchange-hdx/>

De entre la amplia colección de datos de innumerables fuentes que ofrece el portal, se acota la búsqueda a datasets relacionados con **asilo humanitario**. Por ser éste un tema actual y muy mediático del cual se habla, en la mayoría de los casos, sin conocer los datos reales o la situación objetiva [6].

El conjunto de datos vinculado a esta temática que ofrece HDX proviene de la agencia de la ONU para los refugiados (ACNUR). Ofrece dos conjuntos diferenciados de datos con las cifras del número de población en riesgo para cada país y año:

- Población en riesgo que reside en <PAÍS> ⇒ Con un fichero de datos por cada país en el que residen las personas afectadas.
- Población en riesgo originaria de <PAÍS> ⇒ Con un fichero de datos por cada país del que son originarias las personas afectadas.

En los datos propuestos, las cifras del número de personas afectadas se segregan en distintas categorías según la situación del desplazado [7]:

Solicitante de Asilo	Busca protección internacional. Son aquellos casos pendientes de resolución.
Refugiado	Huyen del conflicto o de la persecución. Su condición y su protección están definidas por el derecho internacional.

Retornados	Personas que han regresado a su hogar después de haberse visto obligadas a dejarlo previamente.
Personas desplazadas internas	Aquellos que buscan seguridad en otras partes de su país sin cruzar fronteras internacionales.
Apátridas	Personas que no tienen una nacionalidad y pueden tener dificultades para acceder a derechos humanos básicos

Existen por tanto diferencias clave entre la situación de un **solicitante de asilo**, cuya solicitud de refugio todavía no se ha procesado y un **refugiado**, que ya cuenta con el estatus establecido por la ONU que ofrece protección y le asegura la subsistencia.

Dentro del grupo de personas afectadas también se encuentran los ya **retornados** a su país de origen, las personas que por no cruzar fronteras internacionales son consideradas **desplazadas internas** y los **apátridas**, que carecen de nacionalidad con todo lo que ello conlleva.

Independientemente de la terminología, según ACNUR, en promedio cada año cerca de un millón de personas solicitan asilo por verse obligados a abandonar su país de origen debido a conflictos bélicos, desastres naturales o persecuciones sociales, raciales y/o ideológicas.

Por la naturaleza de los datos seleccionados, se estima oportuno orientar la preparación y análisis de los mismos hacia la Teoría de Grafos. Tratando a los países como nodos y siendo las relaciones el número de personas desplazadas desde su país de origen al país de acogida.

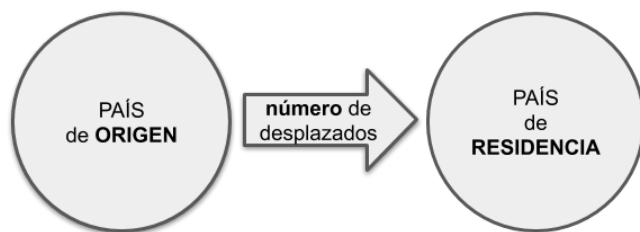


Fig. 3 Modelo Inicial del Grafo

Se pretende, por tanto, extraer y procesar con Python un conjunto de datos sobre asilo humanitario de la plataforma HDX, cargarlos en Neo4j, analizarlos utilizando consultas Cypher y algoritmos ofrecidos por la base de datos y finalmente, visualizar los resultados en neovis.js. Además de estas tecnologías mencionadas, el stack se completa con el uso de Docker, Docker Compose, Git, DVC, Google Cloud y Make (explicados en detalle en el apartado de [arquitectura](#))

2.2 Objetivos

En su conjunto, este proyecto tiene como objetivo principal la extracción de datos abiertos con fin social utilizando una API, el procesamiento y la ingesta de los mismo en una

base de datos apropiada para finalmente analizarlos y aplicar sobre ellos los algoritmos adecuados.

Así como se hace en gestión de proyectos en los que se utiliza la técnica Scrum, para el desarrollo de esta aplicación también se definen una serie de hitos [\[8\]](#) que la desglosan en un conjunto de bloques.

Entre las ventajas de trabajar concretando hitos previamente se encuentran: el poder realizar un seguimiento y evaluación periódica del funcionamiento, la facilidad que aporta para realizar modificaciones en los requisitos, además de un mejor control de los plazos y (aunque en este caso no aplica) de los costes.

Los bloques o hitos en los que se desgrana este proyecto son:

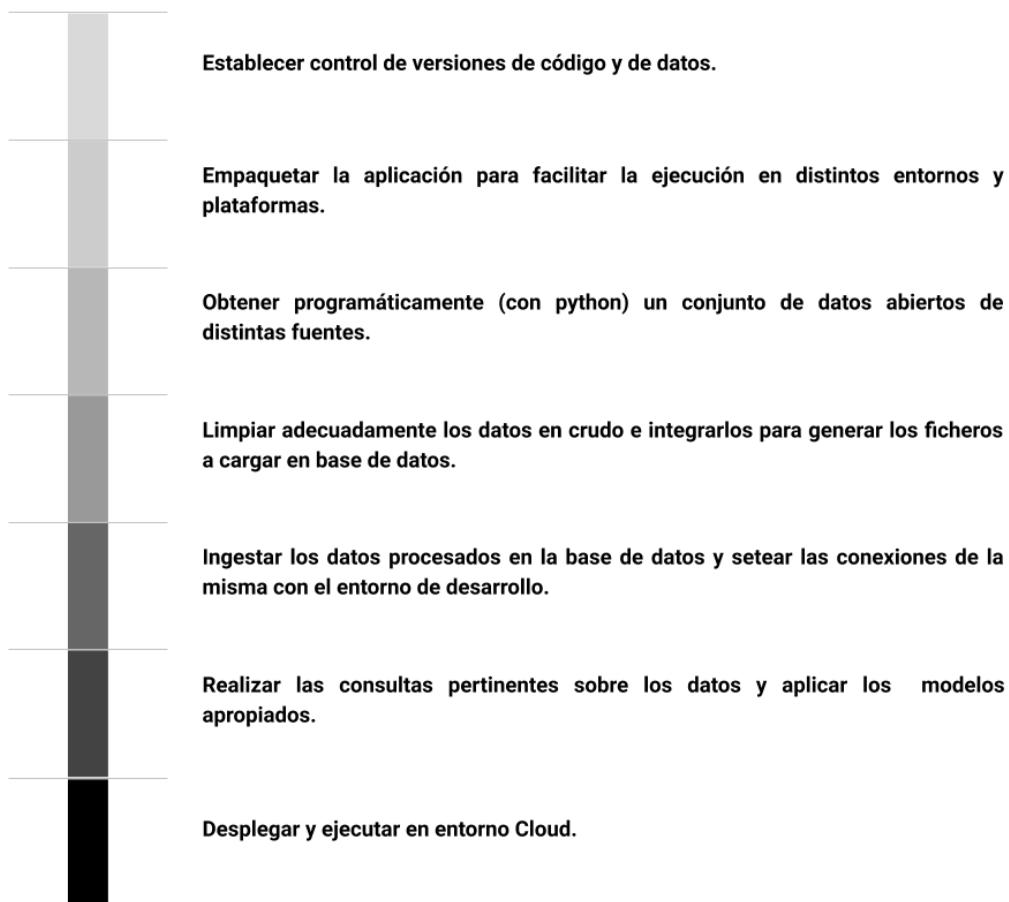


Fig. 4 Hitos del Proyecto



3 ANÁLISIS DEL PROCESO

En esta sección se detalla en profundidad el proceso desarrollado explicando, en los casos en los que se considere oportuno, el código de la aplicación.

3.1 Extracción de datos

Según la página de HDX para developers, la forma recomendada de desarrollar aplicaciones que hagan uso de los datos que ofrecen es con el HDX Python API. Se trata de una librería con soporte para python 2.7 y 3 que hace que publicar datos a, y descargar datos de HDX sea muy fácil. Su uso es simple y la curva de aprendizaje muy suave gracias a la sencilla interfaz y al hecho de que todos los objetos HDX están implementados como clases de Python.

Para conectarse al servidor de datos es necesario definir una configuración. Si solo se busca leer y descargar datos, como es éste caso, se puede hacer de manera muy sencilla así:

```
Configuration.create(hdx_site='prod',user_agent='population_of_concern',  
hdx_read_only=True)
```

Donde los argumentos especificados indican que solo se va a acceder a la versión productiva de datos de HDX en modo lectura. De querer hacer inserciones o modificaciones en los datasets existentes, es necesario disponer de un API key con la que identificarse.

Con un simple `get_all_dataset_names()` de un objeto `Dataset()` se genera un listado de todos los datasets disponibles en forma de lista. Obteniendo la longitud de dicha lista se observa que en HDX hay más de 13000 datasets disponibles.

Lo que resta del código de obtención de datos es una función propia que hace uso de la clase `Dataset()` que ofrece la librería y de los métodos `get_resources()` y `download()` para obtener metadatos y descargar los ficheros de datos. Con esta función y el listado de todos los datasets posibles se confeccionan tres listas con los tres conjuntos de datos a descargar:

- Población de riesgo residiendo en * ("refugees-residing" en el nombre del fichero)
- Población de riesgo originaria de * ('refugees-originating' en el nombre del fichero)
- Indicadores socioeconómicos del Banco Mundial de * ('world-bank-indicators-for' en el nombre del fichero)

*países con datos disponibles

Una experiencia de uso muy satisfactoria y un conjunto de datos adecuadamente curado y mantenido, hacen de HDX una muy buena plataforma de la que obtener datos abiertos de ámbito internacional con carácter social.



3.2 Procesamiento de datos

Una parte importante del tiempo de trabajo de todo Data Scientist se invierte en la fase de preprocesado de datos. En una encuesta publicada en 2016 [9] se constataba que el porcentaje de tiempo dedicado a limpieza y organización de datos asciende a un 60%, seguido en un 20% por el paso ya definido en el apartado anterior de obtención de datos.

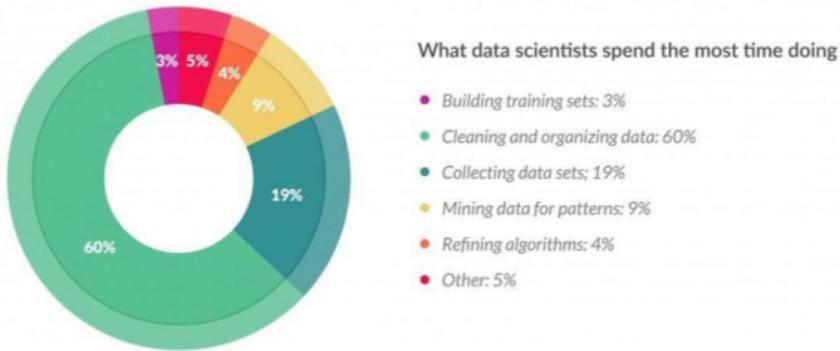


Fig. 5 Reparto del Tiempo de un Data Scientist

<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#efaf0fc76f637>

Al tratarse de una tarea (también conocida como *Data Wrangling* o *Data Mugging*) que consume tantos recursos de tiempo es bueno desgranarla en una serie de subtareas [10] como las que se presentan a continuación:

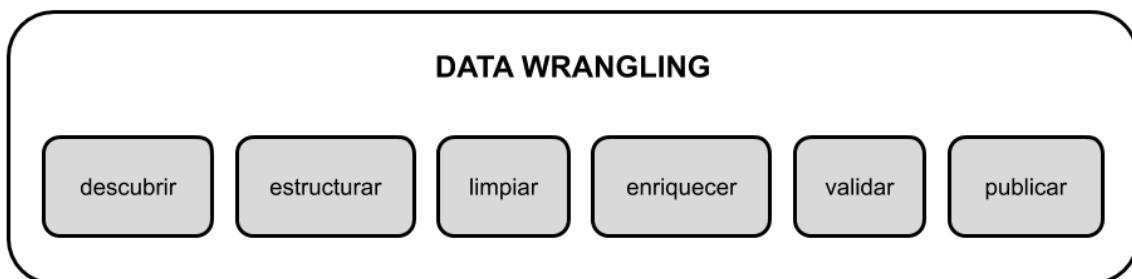


Fig. 6 Subtareas del proceso de Data Wrangling

La fase de **descubrimiento** se ha llevado a cabo analizando las cabeceras, el número de registros y la estructura general de los ficheros *raw*.

En base a esa exploración previa y a la definición del [modelo de base de datos](#) que se tratará más adelante, se lleva a cabo una **estructuración** de los datos. En concreto con los ficheros de 'World Bank Indicators' en los que ha sido necesario transponer los indicadores de filas a columnas.

A continuación se procede a la etapa de **limpieza** propiamente dicha, en la cual lo que se hace pasar como dato faltante difiere según que tipo de fichero se esté tratando:

- *Population of concern*. En este caso serán considerados datos faltantes los asteriscos (valor de N/A por defecto del fichero), cuando el número total de población de ese registro sea cero y/o cuando el nombre del país sea desconocido.



```
na_values=[ "*", "0.0", "Various/Unknown" ]
```

Para eliminar los registros con datos faltantes en este tipo de fichero se dan dos casuísticas:

- Si solo se quiere obtener el número de afectados total (sin tener en cuenta el resto de columnas en las que se desglosa el total).

```
df = df.dropna(axis=0, how='any')
```

- O si se prefiere mantener todas las categorías cuya suma conforma la cifra total. En este caso solo se eliminará un registro si y sólo si alguno de los valores del total de afectados, la fecha y/o los países de origen y destino sean nulos.

```
df = df.dropna(axis=0, subset=['#country+residence', '#country+origin',
 '#date+year', '#affected+total'], how='any',)
```

- *World Bank Indicators*. Más sencillo ya que serán N/A todas las celdas vacías o no informadas y serán eliminados todos los registros que contengan al menos un dato faltante en alguna de las columnas.

```
na_values=[ "" ]
```

En lo que respecta a **enriquecer** el conjunto de datos, se determina que los ficheros con las cifras de población en riesgo son el dataset principal y los ficheros con los indicadores socioeconómicos cumplen la función de enriquecer a los primeros aportando información complementaria.

No se realiza una **validación** de datos en el sentido estricto de la palabra. Lo cual no quita de que haya procesos posteriores que adviertan en el caso de que los datos no se hayan procesado correctamente. En este proceso ese paso crítico es la carga en base de datos que se verá en el siguiente apartado. Concretamente ha sido especialmente delicado el tratamiento y formateado de los tipos de datos para adaptarlo a lo que la base de datos admitía como válido. Se propondrá como mejora futura el incluir reglas de Data Quality que garanticen la consistencia, integridad y exactitud de la información manejada.

Por último y, aunque se hablará de ello más en detalle en el apartado de Arquitectura, los datos se **publican** en un repositorio remoto utilizando un software de versionado de datos (DVC) que facilita la compartición y reproducibilidad de los mismo en los proyectos de Data Science.



3.3 Ingesta en base de datos

Tal y como se menciona en el manual de referencia en el que se basa mayoritariamente este proyecto, Graph Databases de O'Reilly [28], a las bases de datos relacionales les faltan las relaciones. Este tipo de base de datos fue concebido para replicar formatos de datos tabulares, pero tienen serios problemas a la hora de modelar el tipo de relaciones que se dan en el mundo real.

Las bases de datos orientadas a grafos, en cambio, poseen el poder de modelado de las relacionales, combinado con la rapidez y flexibilidad para ajustarse a situaciones reales. Haciendo posible equiparar en importancia las relaciones con las entidades representadas.

A la hora de seleccionar una base de datos orientada a grafos hay que fijarse principalmente en el rendimiento de dos de sus propiedades:

- El almacenamiento subyacente. Se busca que el grafo se almacene de manera nativa y no como en algunos casos en los que parte del almacenamiento se realiza en bases de datos relacionales u orientadas a objetos.
- El motor de procesamiento. Se precisa que la adyacencia sea sin índice. Toda base de datos influenciada por esta característica se considerará que procesa grafos de manera nativa.

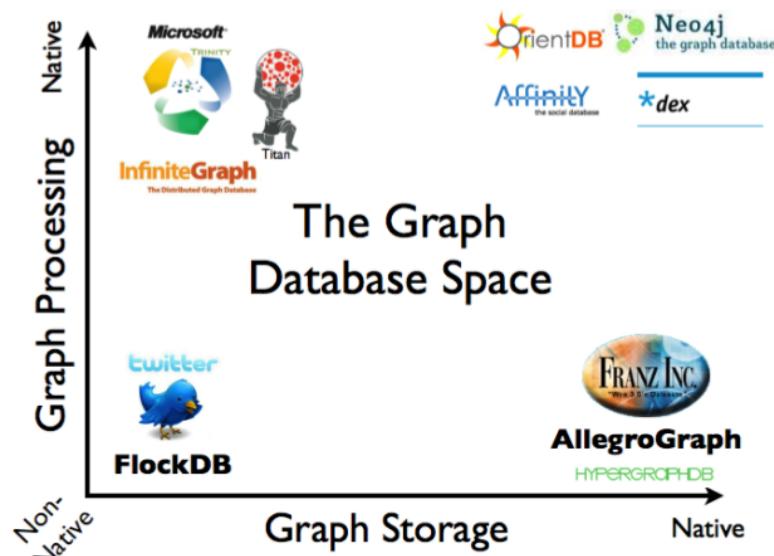


Fig. 7 Comparativa de Bases de Datos orientadas a Grafos
<https://www.oreilly.com/library/view/graph-databases-2nd/9781491930885/ch01.html>

Como se observa en el gráfico superior, Neo4j se encuentra a la cabeza del resto de tecnologías existentes en lo que a estas dos propiedades respecta y ello la convierte en la base de datos orientada a grafos de referencia.

Una vez pre-procesados los datos adecuadamente, lo que sigue es insertarlos en base de datos para su almacenamiento y consulta. Neo4j presenta dos formas de realizar esta operación:

- Importación inicial. Neo4j ofrece una herramienta especial llamada neo4j-import que alcanza velocidades de ingesta de alrededor de 1.000.000 registros por segundo.
- Importación por lotes. En un grafo ya existente se pueden cargar datos (nodos y relaciones) utilizando el comando LOAD de Cypher. Está diseñado para soportar cargas intermedias de cerca de un millón de ítems.

Se opta por la primera de ellas ya que no es necesario hacer inserciones en tiempo real y además reduce el tiempo de carga en base de datos a unos pocos segundos. El input de la herramienta neo4j-import son un conjunto de ficheros csv que proporcionan los datos de los nodos y las relaciones.

Se presentan a continuación las cabeceras de cuatro de los ficheros generados en el preprocesamiento con el fin de importarlos en neo4j:

- Dos ficheros con nodos. Uno por cada tipo de label : Country Year y Country

`countries_nodes_residing.csv`

```
countryYearId:ID(CountryYear-ID),country,year:int,population:float,int_migrant_st
ock:float,pop_growth_percentage:float,urban_pop_percentage:float,:LABEL
Netherlands1992,Netherlands,1992,15184166.0,0.7560565933967871,70.223,CountryYe
ar
```

`countries_residing.csv`

```
countryId:ID(Country-ID),:LABEL
Dominican Rep.,Country
```

- Y dos ficheros con las relaciones entre los nodos especificados:

`relationships_all_residing.csv`

```
:START_ID(CountryYear-ID),:END_ID(CountryYear-ID),affected_number:int,:TYPE
Tibetan2000,Liechtenstein2000,32,RESIDE_IN
```

`countries_years_relationship_residing.csv`

```
:START_ID(Country-ID),:END_ID(CountryYear-ID),:TYPE
Ukraine,Ukraine2002,2002
```

Con esos ficheros se puede ejecutar el siguiente comando en el contenedor docker que esté corriendo la imagen de neo4j.



```
bin/neo4j-import -into data/databases/graphHDX.db --nodes
import/countries_nodes_residing.csv --nodes import/countries_residing.csv
--relationships import/relationships_all_residing.csv --relationships
import/countries_years_relationship_residing.csv
```

Una vez realizada la carga, es necesario reiniciar el servicio del contenedor con la base de datos para que ésta sea accesible y poder efectuar consultas.

3.4 Análisis de datos

Con los datos ya cargados en base de datos se procede a realizar un análisis exploratorio de los mismos. En este proyecto se opta por la rama de la **analítica descriptiva** [\[11\]](#) utilizando agregaciones y minería de datos para ofrecer una visión de lo que ya ha pasado. Si lo que se busca es responder a la pregunta de qué pasará, habría que recurrir a técnicas de analítica predictiva y si por el contrario lo que se quiere es dar solución a problemas de optimización o elección entre posibles resultados, se utilizará analítica prescriptiva.

Cypher [\[12\]](#) es el lenguaje de consulta de Neo4j, un lenguaje simple y lógico diseñado para reconocer patrones en los datos. Está construido para ser comprensible para los humanos con prosa e iconografía en inglés, que hacen que la sintaxis sea visual y fácil de entender.

En Cypher los nodos se representan entre paréntesis (n:TypeNode) y las relaciones entre corchetes con las flechas indicando el sentido de la misma -[r:TypeRel]->. Al igual que en la mayoría de lenguajes de programación, existen una serie de palabras reservadas para acciones específicas de consulta. Dos de las más utilizadas son MATCH (busca los nodos, propiedades, relaciones, etiquetas o patrones que existan en la base de datos) y RETURN (especifica qué valores o resultados se quieren devolver en la consulta Cypher).

```
MATCH p=()-[r:RESIDE_IN]->(n) WHERE n.year=2017 AND n.country="Spain" RETURN p
```

El código sobre estas líneas muestra una consulta muy sencilla en lenguaje Cypher que se puede realizar sobre el grafo de este proyecto cargado en Neo4j. En este caso se buscan todos los nodos y relaciones que cumplan la condición de que el país de residencia sea España en el año 2017. Introduciendo así el filtrado de consultas en Cypher, que en este caso se realiza con la palabra reservada WHERE pero que también se puede hacer utilizando el control de igualdad en la cláusula MATCH. Es decir, la consulta ya definida equivale a :

```
MATCH p=()-[r:RESIDE_IN]->(n {country:"Spain", year:2017}) RETURN p
```

En esta aplicación, las consultas sobre el grafo cargado en base de datos se realizan utilizando el driver de Neo4j para python, el cual permite interactuar con Neo4j desde un programa python manteniendo el léxico de Cypher.

El detalle de las consultas realizadas y el resultado de las mismas se reserva para el apartado [Análisis exploratorio de Datos](#) de esta memoria.

3.5 Algoritmos

Así como se explica en el capítulo uno del libro [Graph Algorithms](#) de O'Reilly, los algoritmos de grafos son un subconjunto de las herramientas que se pueden utilizar en análisis de grafos. Más concretamente, se entiende por algoritmos de grafos al análisis computacional que se realiza sobre grafos de manera global e iterativa. No se incluiría por tanto en esta categoría a las consultas basadas en patrones, utilizadas por lo general para realizar análisis de datos de manera local.

Los algoritmos de grafos se basan en las matemáticas definidas en la teoría de grafos, que aprovechan las relaciones entre nodos para inferir la organización y dinámicas de sistemas complejos.

Los algoritmos existentes se pueden dividir en tres áreas específicas: Pathfinding (búsqueda de rutas), Centralidad y Detección de comunidades. Por la naturaleza del grafo definido en este proyecto, los algoritmos que se aplicarán pertenecen a las dos últimas categorías.

- **Centrality** [\[29\]](#) ⇒ Sirven para entender los roles de nodos específicos en un grafo y su impacto en la red. Son particularmente útiles para identificar los nodos más importantes y permiten entender dinámicas entre los mismos. Fueron creados para análisis de redes sociales pero desde entonces se han extendido a otros muchos campos donde la teoría de grafos es aplicable.

PageRank	Estima la importancia de un nodo en relación a sus vecinos.
ArticleRank	Variante del algoritmo PageRank.
Betweenness Centrality	Mide el número de caminos más cortos que pasan por un nodo.
Closeness Centrality	Calcula qué nodos tienen el camino más corto a otros nodos.
Harmonic Centrality	Variante de Closeness Centrality.
Eigenvector Centrality	Mide la influencia transitiva o conectividad de los nodos.
Degree Centrality	Mide el número de relaciones que tiene un nodo.

- Community Detection [\[29\]](#) ⇒ Valen para detectar grupos de nodos semejantes y evaluar su comportamiento, así como grupos aislados que en conjunto configuran la estructura de la red. Útiles para visualizaciones generales de los grafos.

Louvain	Maximiza precisión de agrupaciones al comparar pesos de las relaciones con un promedio definido.
Label Propagation	Infiere los clusters al propagar etiquetas basadas en mayorías de grupos de nodos.
Connected Components	Independiente de dirección en relaciones, busca grupos en los que cada nodo sea accesible por el resto de nodos del mismo grupo.
Strongly Connected Components	Teniendo en cuenta dirección en relaciones, busca grupos en los que cada nodo sea accesible por el resto de nodos del mismo grupo.
Triangle Counting / Clustering Coefficient	Mide cuántos nodos forman triángulos y el grado de los nodos tienden a agruparse.
Balanced Triads	Utilizado para evaluar el equilibrio estructural de la red.

Los algoritmos implementados en la aplicación y el resultado de los mismos se describen en los apartados [ALGORITMOS](#) y [RESULTADOS](#) de la memoria.

3.6 Visualización

Para visualización de grafos en este proyecto se utiliza [Neovis.js](#) [\[13\]](#). Una herramienta que se emplea para crear visualizaciones de grafos basadas en JavaScript que pueden ser embebidas en una aplicación web.

Neovis.js se sirve del conector JavaScript de Neo4j para conectarse y extraer datos de la base de datos y de la librería vis.js para renderizar las visualizaciones de los grafos. También permite aprovecharse de los algoritmos de grafos para diseñar la visualización vinculando los valores de las propiedades con componentes visuales.

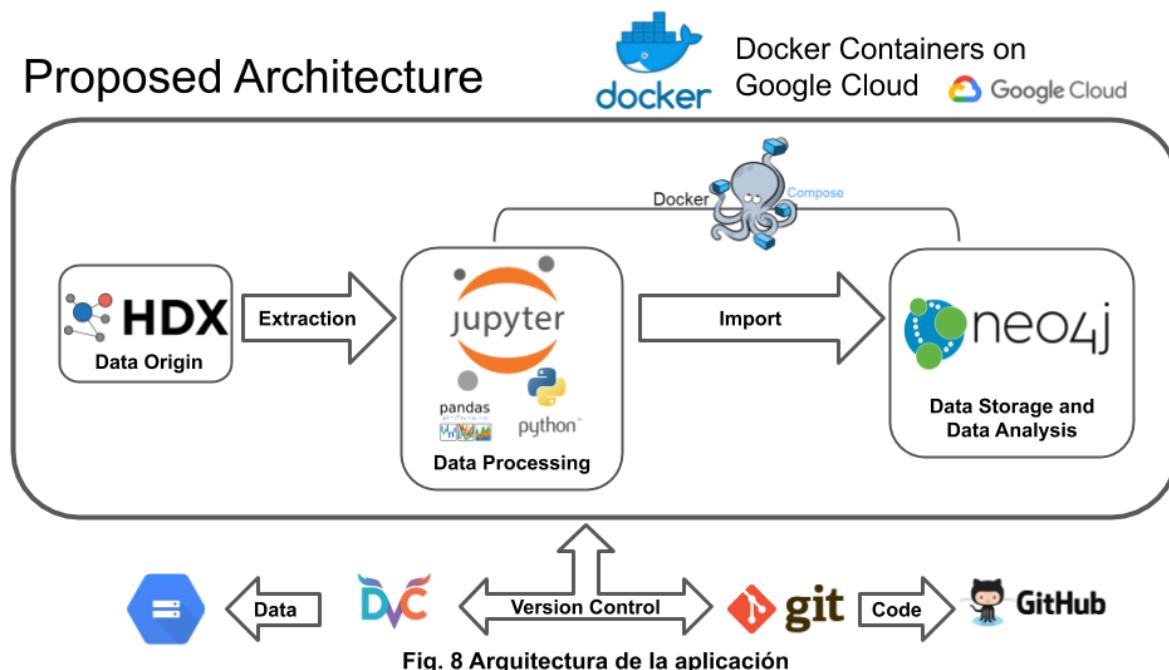
Las características más destacables de la versión de neovis.js utilizada en esta aplicación son :

- Conexión a instancias de Neo4j para obtener datos en tiempo real.
- Posibilidad de mostrar etiquetas y propiedades especificadas por el usuario.
- Delimitar los datos a mostrar con una consulta Cypher por defecto.
- Posibilidad de establecer grosor de relaciones, tamaño de nodos y color de los mismos en función de resultados de algoritmos de comunidades/clustering.



4 ARQUITECTURA

La arquitectura propuesta está basada en contenedores Docker orquestados con Docker Compose. Se construyen dos imágenes docker, una con Jupyter Notebooks con kernels de Python y otra con Neo4j. Para facilitar la reproducibilidad y realizar control de versiones se utiliza Git para código (con repositorio remoto en Github) y DVC para almacenamiento remoto en Google Cloud para datos.



4.1 Contenedores Docker orquestados con Docker Compose

Según la documentación oficial de Docker [14]: “Un **contenedor** es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y confiable en cualquier entorno informático.”

Lo que se construyen son **imágenes** Docker, incluyendo en ellas todo lo necesario para ejecutar una aplicación. Las imágenes se convierten en contenedores Docker cuando se ejecutan en Docker Engine.

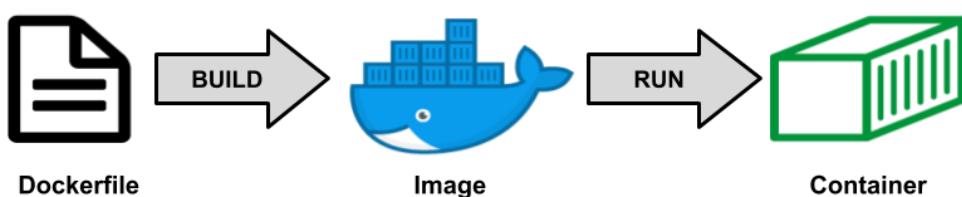


Fig. 9 Evolución desde Dockerfile hasta Contenedor Docker

Los contenedores Docker ofrecen tres ventajas reseñables:

- Estandarización. Favoreciendo la portabilidad de las aplicaciones.

- Ligereza. Utilizan el OS del host por lo que no requieren un OS por aplicación.
- Seguridad. El contenedor proporciona una capacidad de aislamiento que confiere a la aplicación una mayor seguridad.

En lo que respecta a proyectos de Data Science hay cuatro motivos principales [\[15\]](#) por los que elegir contenedores Docker como el soporte sobre el que desarrollar el trabajo:

- I. Reproducibilidad ⇒ Asegura que la aplicación se podrá ejecutar sobre cualquier entorno sobre el que corra Docker engine ya que la imagen incluye todo lo necesario. No se distribuye solo código si no también el entorno de desarrollo/ejecución.
- II. Consistencia ⇒ Ofrece un entorno de ejecución uniforme y consistente para todo artefacto que se haya desarrollado reduciendo el tiempo de administración del sistema.
- III. Trazabilidad ⇒ No sólo es posible versionar los scripts con los que se construyen las imágenes docker, si no que también se pueden almacenar las propias imágenes en un repositorio con control de versiones.
- IV. Portabilidad ⇒ Es muy sencillo trasladar imágenes docker entre distintos entornos e incluso existen plataformas que permiten escalar las aplicaciones de manera muy sencilla.

Entre los frameworks existentes que permiten automatizar la implementación, el escalado y la administración de contenedores destacan principalmente Swarm y Kubernetes. No obstante, debido a las reducida dimensiones de este proyecto en concreto, se ha optado por la herramienta Docker Compose.

Docker Compose [\[16\]](#) sirve para definir y ejecutar aplicaciones Docker que hagan uso de más de un contenedor a la vez. Los servicios se configuran con YAML y se pueden levantar de manera muy sencilla con un simple comando. Compose se puede utilizar en todos los entornos en los que pueda correr Docker desde desarrollo o testing, así como producción o flujos de CI.

Después de definir los Dockerfile de las distintas imágenes implicadas (a detallar en el siguiente subapartado), se elabora un archivo docker-compose.yml donde se especifican los servicios que se levantarán con el comando docker-compose up.

A continuación se explica el código del fichero YAML que corresponde a cada uno de los servicios de la aplicación:

- neo4j-hdx

```
build: ./docker/neo4j
image: neo4j-hdx
container_name: neo4j-hdx-container
ports:
  - "7687:7687"
```

```

- "7474:7474"
volumes:
- ./neo4j/data:/data
- ./neo4j/logs:/logs
- ./neo4j/import:/var/lib/neo4j/import
environment:
- NEO4J_AUTH=neo4j/test
- NEO4J_dbms_active_database=graphHDX.db
  
```

A parte de indicar dónde se encuentra el Dockerfile de la imagen a construir y cuál será el nombre (tanto de la imagen como del contenedor que se ejecute), se exponen dos puertos para acceder a neo4j y se montan 3 volúmenes (data, logs e import). Por último se setean dos variables de entorno: el usuario/contraseña de la base de datos y el nombre de la base de datos que se activará al levantar el servicio.

- jupyter-hdx

```

build: ./docker/jupyter
image: jupyter-hdx
container_name: jupyter-hdx-container
ports:
- "8888:8888"
volumes:
- ./home/jovyan/work
working_dir: /home/jovyan/work
  
```

Al igual que en el anterior, se apunta al path del Dockerfile y se da nombre a la imagen y contenedor, pero en este caso solo se expone un puerto y se monta un volumen. También se fija el directorio del Workspace de Jupyter.

4.1.1 Imagen Docker con Neo4j

En lo que respecta a la arquitectura de neo4j se decide utilizar la base de datos en modo *server* (en contraposición a modo *embedded*). No solo porque es el modo en el que se despliega en gran medida la base de datos hoy en día, sino también porque:

- Ofrece un API REST que permite enviar peticiones HTTP en formato JSON y además soporta la ejecución de consultas Cypher.
- El que se acceda a los datos con documentos JSON otorga independencia de la plataforma en la que se ejecute el cliente.
- Permite el escalado indistintamente del clúster del servidor de la aplicación

Se toma como imagen docker de partida la ofrecida por Neo4j [\[17\]](#) con la edición community y se le añaden dos extensiones de librerías:

- graph-algorithms-algo ⇒ Proporciona para Neo4j 3.x versiones de algoritmos de grafos eficientemente implementadas expuestos como procedimientos de Cypher. Ofrece, entre otros, algoritmos de centralidad y de detección de comunidades.
- apoc (Awesome Procedures on Cypher) ⇒ Librería estándar de utilidades para procedimientos y funciones comunes. Es, posiblemente, la mayor y más ampliamente usada librería de Neo4j. Para esta aplicación se hará uso del módulo que posibilita exportar subgrafos de la base de datos a otros formatos.

Ambas librerías se instalan de la misma manera. Con `wget` se descargan los jars en la versión correspondiente, dejando ambos en la carpeta `/var/lib/neo4j/plugins/` del contenedor. Los ejecutables que se han descargado para este caso son:

```
graph-algorithms-algo-3.5.4.0.jar  
apoc-3.5.0.3-all.jar
```

Por último, se necesita dar acceso sin restricciones a los paquetes porque los algoritmos usan la API del Kernel de nivel inferior para leer y escribir en Neo4j. Esto se consigue seteando la siguiente variable de entorno que incluirá la propiedad indicada en el fichero de configuración de la base de datos.

```
ENV NEO4J_dbms_security_procedures_unrestricted=algo.\\*\*,apoc.\\*\*
```

4.1.2 Imagen Docker con Jupyter Notebook

La imagen de partida sobre la que se construye el entorno de desarrollo con Jupyter y kernels de python pertenece al **Jupyter Docker Stacks** [18], un set de imágenes Docker listas para ser ejecutadas que contienen las aplicaciones y herramientas interactivas de Jupyter.

Las imágenes del stack de Jupyter se crean de manera incremental. Es decir, partiendo de una imagen *core* con lo más básico, se van generando imágenes que, además de incluir las funcionalidades de la imagen previa, añaden algo nuevo, aumentando el tamaño en consecuencia.

En función de las necesidades del proyecto se puede seleccionar una imagen u otra. En este caso y en vista de [posibles mejoras futuras](#) que requieran procesamiento con Spark, se ha optado por la imagen de *pyspark-notebooks*. De caja con esta imagen (además de *pyspark*), se encuentran los paquetes más importantes del ecosistema científico de Python: *pandas*, *scipy*, *matplotlib*, *scikit* o *bokeh*, entre otros.

Para llevar a cabo las tareas definidas en este proyecto es necesario instalar con `pip` una serie de paquetes de python adicionales. Estas librerías se listan en el fichero `requirements.txt`:

```
hdx-python-api
```



```
py2neo  
neo4j  
nbdime  
tqdm  
networkx  
graphframes
```

- HDX Python. Como ya se definió en el apartado de [Extracción de datos](#), se trata de una librería de python diseñada para facilitar el desarrollo de código que interacciona con la plataforma de Humanitarian Data Exchange (HDX). Presenta un API sencillo que permite descargar y subir datos a HDX.
- Py2neo. Es una librería cliente y un kit de herramientas [\[19\]](#) para trabajar con Neo4j desde aplicaciones Python y desde la línea de comandos. Soporta tanto Bolt como HTTP y ofrece un API de alto nivel que permite interaccionar de manera intuitiva con la base de datos.
- Neo4j Python Driver. Controlador oficialmente compatible con Neo4j que se conecta a la base de datos utilizando el protocolo binario. Toda la actividad de la base de datos se coordina a través de Transacciones y Sesiones [\[20\]](#). Siendo una Transacción una unidad de trabajo que se confirma en su totalidad o se retrotrae en caso de fallo. Y una Sesión, un contenedor lógico para cualquier número de unidades de trabajo transaccionales relacionadas con la causa.
- nbdime. Proporciona herramientas de *diffing* y *merging* de Jupyter Notebooks [\[21\]](#). Especialmente útil cuando se desea incluir notebooks (guardados en texto plano en formato JSON) en un flujo de versionado de código. No solo ofrece la posibilidad de usarlo en línea de comandos si no que dispone también de una extensión, en este caso para jupyter lab, que permite, de una manera muy sencilla y visual, examinar los cambios que se hayan producido en el código.
- tqdm. Se instala para hacer que los loops exhiban un medidor de progreso con información acerca de la progresión de la tarea.
- Networkx. Aunque el análisis se hará con las herramientas de consulta de la base de datos, se podrá usar esta librería para visualización de grafos, en conjunto con matplotlib, desde la parte python.
- Graphframes. De cara a ampliar y mejorar el proyecto en el futuro se incluye este paquete que aporta y extiende la funcionalidad de GraphX en forma de API de alto nivel para Pyspark.

Ya que también se propondrá como [mejora futura](#) el realizar tests de la aplicación, se busca dejar el entorno de desarrollo lo más preparado para ello posible. Por tanto se incluye un segundo fichero `requirements-test.txt` con un listado de librerías que

también se instalarán con pip. Las dos librerías principales a tener en cuenta son pytest (framework que facilita el desarrollo y ejecución de tests en python) y nbval (plugin de pytest para validar Jupyter Notebooks).

Finalmente cabe destacar en el Dockerfile de la imagen de jupyter que se cambia el CMD (comando por defecto) para que el contenedor al arrancar lance un servidor de Jupyter Lab en lugar del estándar de Jupyter Notebook. Además se le indica que no requiera token al conectarse a través del navegador web.

```
CMD start.sh jupyter lab --LabApp.token=''
```

4.2 Control de Versiones

Los sistemas de control de versiones son un tipo de herramientas de software que ayudan a los equipos a gestionar los cambios en el código a lo largo del tiempo guardando cada modificación en un tipo de base de datos especial. En caso de errores, siempre es posible volver a estados anteriores del código minimizando las interrupciones al resto del equipo.

4.2.1 Código

Como herramienta de versionado de código se ha elegido Git, por tratarse de la más extendida y de fácil uso, con repositorio remoto en [GitHub](#).

Para este proyecto se establece un sencillo Git Flow [\[22\]](#) en el que el repositorio central consiste de dos ramas principales con vida infinita:

- master. Rama principal donde lo que allí se encuentra será siempre código productivo.
- develop. Paralela a master, es la rama donde el código siempre refleja un estado con los últimos cambios de desarrollo liberados para la próxima versión.

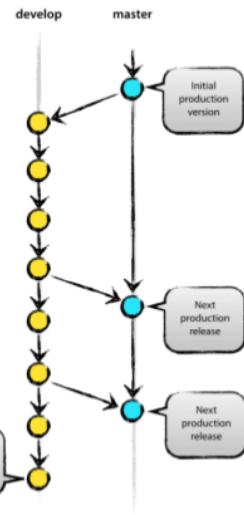


Fig. 10 Git Flow del Proyecto
<https://nvie.com/posts/a-successful-git-branching-model/>

4.2.2 Datos

Para llevar a cabo el visionado de datos y código se propone el uso de DVC o Data Version Control [23], quizá la parte más novedosa en lo que a la arquitectura de esta aplicación se refiere. DVC es una herramienta que permite:

- Guardar y reproducir los ensayos realizados.
- Control de Versiones de modelos y datos.
- Establecer un flujo de trabajo para desarrollo y colaboración entre miembros del equipo y usuarios.

Es por este último punto por el que se decide incluir DVC en la aplicación. Para poder alojar los datos en un repositorio remoto ([storage de google cloud](#) en este caso) y favorecer la compartición de los mismos entre usuarios y plataformas de ejecución.

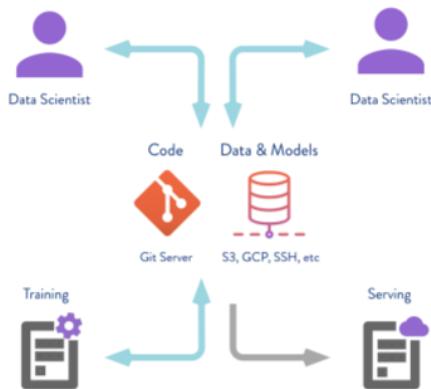


Fig. 11 Caso de uso DVC
<https://dvc.org/doc/use-cases/share-data-and-model-files>

Al ser menos común que Git, por lo general no se encuentra instalado por defecto en ningún sistema, pero éste es un paso muy sencillo ya que el ejecutable está disponible en los repositorios de paquetes de referencia de los principales sistemas operativos.

Tras la instalación y al igual que sucede con Git, si se trata de un repositorio nuevo, se debe inicializar en el directorio correspondiente. Si ya está inicializado (ya existe un subdirectorio `.dvc/`) se podrá interactuar con el repositorio remoto utilizando `dvc add` y `dvc push` para subir datos y `dvc pull` para bajarlos. Los archivos generados de `dvc` pueden ser comiteados con `git` para gestionar su control de versiones.

4.3 Ejecución en la Nube

Se ha considerado necesario trasladar la ejecución de esta aplicación a la Nube por dos motivos:

- En ordenadores personales de 8 GB de RAM, los 4 GB de memoria que limitan al contenedor de neo4j no son suficientes para consultar el grafo generado o aplicar algoritmos sobre el mismo.
- La tendencia en las empresas, aun aquellas con plataformas propias, es a la hibridación. Por tanto, se busca que un Data Scientist también sea capaz de demostrar habilidades para trabajar en entornos de Cloud Pública.

La creciente importancia y demanda del *Cloud Computing* hace que exista un gran número de proveedores de servicios en la Nube. No obstante, los tres principales son AWS, Azure y Google [\[24\]](#). El que posee mayor cuota de mercado, por ser también el más longevo, es AWS con más de un 62%, seguido de Azure con un 20% y Google con un 12%

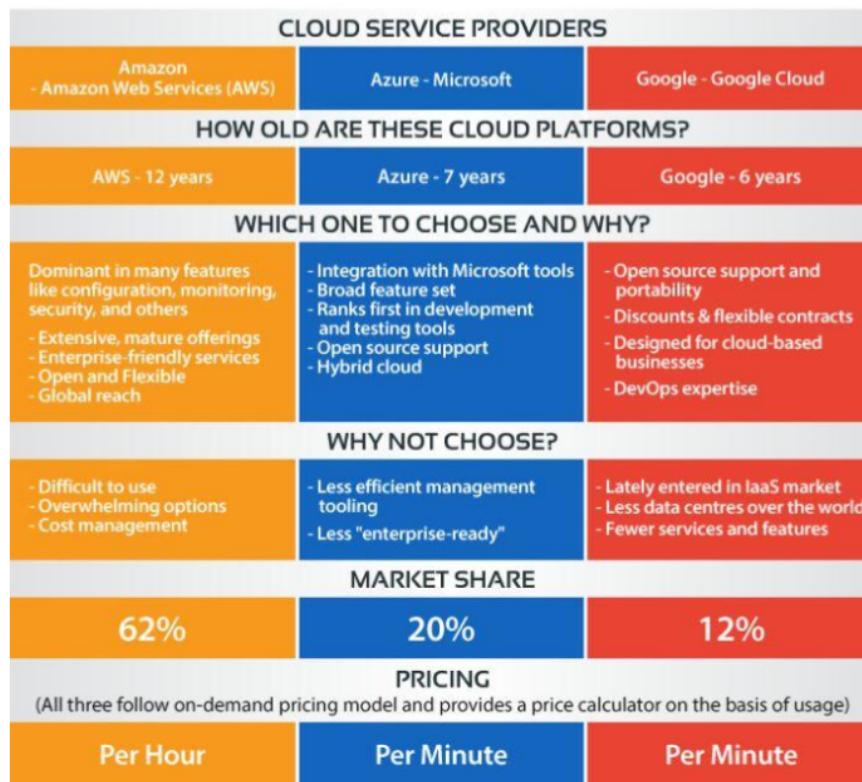


Fig. 12 Comparativa de Proveedores de Servicios Cloud
<https://www.whizlabs.com/blog/wp-content/uploads/2018/05/aws-vs-azure-vs-google-infographic.jpg>

En este caso la balanza se ha decantado por Google Cloud porque no es necesaria ninguna herramienta especial, su uso es más sencillo e intuitivo y además ofrece un plan gratuito durante un año muy competitivo.

Los pasos que se han seguido para configurar el entorno de ejecución en Google Cloud son [\[25\]](#):

1. Crear una cuenta de Google Cloud gratuita asociada a una cuenta de Gmail.
2. Crear un nuevo proyecto.
3. Crear una instancia de Compute Engine. Indicando la zona en la que se va a ejecutar, el tipo de máquina con la memoria necesaria + número de cores y el sistema operativo. Permitiendo además el tráfico HTTP y HTTPS e indicando que no borre el disco cuando se borre la instancia.
4. Hacer que la IP externa sea estática para no tener que cambiar la URL a la que apuntan los servicios cada vez que se levante la instancia.
5. Añadir un par de reglas de Firewall especificando los puertos que exponen los servicios. 8888 para Jupyter y 7474,7687 para Neo4j.

Una vez la instancia se haya configurado correctamente, se podrá ejecutar la aplicación siguiendo los pasos detallados en el [ANEXO I](#).

4.4 Automatización de comandos

Makefile [\[26\]](#) es una herramienta de automatización que ejecuta y compila programas de manera eficiente.

La utilidad `make` requiere un archivo `Makefile` que define una serie de tareas que se vayan a ejecutar. El caso más habitual es utilizar `make` para compilar un programa desde el código fuente. En este proyecto se hará uso de la herramienta para automatizar una serie de tareas y crear alias para otras cuya instrucción sea necesario ejecutar a menudo:

- Comandos de Docker. Para evitar escribir `docker-compose` cuando se construya, levante o paren servicios se definen tres instrucciones: `build`, `up` y `down`
- Comandos de Neo4j. Automatizan la carga de los distintos posibles ficheros de datos en Neo4j.
- Comandos para ejecutar en Google Cloud. Realizan las instalaciones necesarias en las instancias de GC en las que se quiera ejecutar la aplicación.
- Comandos de Tests. Actualmente se encuentra en progreso pero se incluye como mejora futura.

5 DATOS

5.1 Datos Originales

Los datos originales en bruto (o Raw Data) que se extraen de HDX proceden de dos organismos internacionales con bases de datos íntegras, fiables y adecuadamente mantenidas



Fig. 13 Fuentes de Datos en bruto

5.1.1 UNHCR's Population of concern

UNHCR (o ACNUR en castellano) es la Agencia de la ONU para los refugiados y posee una base de datos con datos de población en riesgo desde los años 60 hasta del 2017, incluyendo lugar de origen/residencia de los afectados y el estatus de los mismos(refugiados, solicitantes de asilo, desplazados internos, ...).

En la tabla siguiente se presenta la cabecera de los ficheros de los que se extraen los datos con las cifras de población desplazada, incluyendo el nombre de cada columna, descripción de la misma y el tipo de dato que almacena:

NOMBRE DE COLUMNA	DESCRIPCIÓN	TIPO
#date+year	Año	Integer
#country+residence	País / territorio de asilo/residencia	String
#country+origin	Origen	String
#affected+refugees	Refugiados	Integer
#affected+asylum	Solicitantes de asilo (casos pendientes)	Integer
#affected+returned_refugees	Refugiados retornados	Integer
#affected+idps	Desplazados Internos (IDPs)	Integer
#affected+returned_idps	IDPs Retornados	Integer
#affected+stateless	Apátridas	Integer
#affected+others	Otros en riesgo	Integer
#affected+total	Población Total	Integer

Aunque para cada país se descargan y procesan dos tipos de ficheros (residing in y originating from), en este estudio se utilizarán únicamente los ficheros con la relación

definida de RESIDING_IN. Se considera que ambos ficheros debería arrojar los mismos resultados y se propone como [mejora futura](#) corroborar dicha suposición.

5.1.2 World Bank Indicators

El Banco Mundial (**The World Bank**) es una organización multinacional cuyo propósito declarado es reducir la pobreza en el mundo apoyando a las naciones en desarrollo. Con dicho fin recopilan anualmente (desde hace más de 50) en su base de datos, un conjunto de indicadores socioeconómicos de 217 economías.

Los ficheros con los indicadores del Banco Mundial tienen las siguientes columnas:

NOMBRE DE COLUMNA	DESCRIPCIÓN	TIPO
#country+name	Country Name	String
#country+code	Country ISO3	String
#date+year	Year	Integer
#indicator+name	Indicator Name	String
#indicator+code	Indicator Code	String
#indicator+num	Value	Float

En el fichero de cada país, la columna #indicator+name (o #indicator+code) contiene un registro por cada combinación de indicador disponible y año. La tabla inferior lista todos los indicadores ofrecidos por el World Bank para cada país:

#indicator+name	IN
Population, total	✓
Age dependency ratio, old (% of working-age population)	□
Age dependency ratio, young (% of working-age population)	□
Number of infant deaths	□
International migrant stock (% of population)	✓
Land area (sq. km)	□
Population growth (annual %)	✓
Population density (people per sq. km of land area)	□
Population in urban agglomerations of more than 1 million	□
GNI (current US\$)	□
Poverty headcount ratio at \$1.90 a day (2011 PPP) (% of population)	□
Mortality rate, adult, female (per 1,000 female adults)	□
Mortality rate, adult, male (per 1,000 male adults)	□
Logistics performance index: Overall (1=low to 5=high)	□
Rail lines (total route-km)	□

GNI, PPP (current international \$)	<input type="checkbox"/>
GDP per capita, PPP (current international \$)	<input type="checkbox"/>
GNI per capita, PPP (current international \$)	<input type="checkbox"/>
Net ODA received per capita (current US\$)	<input type="checkbox"/>
Mobile cellular subscriptions (per 100 people)	<input type="checkbox"/>
Households and NPISHs Final consumption expenditure, PPP (constant 2011 international \$)	<input type="checkbox"/>
Access to electricity (% of population)	<input type="checkbox"/>
GINI index (World Bank estimate)	<input type="checkbox"/>
Inflation, consumer prices (annual %)	<input type="checkbox"/>
Prevalence of undernourishment (% of population)	<input type="checkbox"/>
Fixed telephone subscriptions (per 100 people)	<input type="checkbox"/>
GDP per capita, PPP (constant 2011 international \$)	<input type="checkbox"/>
Prevalence of HIV, total (% of population ages 15-49)	<input type="checkbox"/>
Urban population (% of total)	<input checked="" type="checkbox"/>

Para este trabajo se han seleccionado los indicadores marcados con en la columna **IN**, pues se considera que complementan al estudio de los datos de Asilo Humanitario de UNHCR.

- Population, total ⇒ La población total incluye todos los residentes de un país independientemente de estatus legal o ciudadanía. Los valores ofrecidos son estimaciones realizadas a mitad de año.
- International migrant stock (% of population) ⇒ Número de personas nacidas en un país distinto al país en el que residen. Incluye refugiados. Los datos se obtienen principalmente de los censos de población. Se utiliza un modelo para estimar los migrantes de países de los que no se poseen datos.
- Population growth (annual %) ⇒ La tasa de crecimiento anual de la población para el año t es la tasa exponencial de crecimiento de la población de medio año desde el año $t-1$ hasta el año t , expresada como porcentaje.
- Urban population (% of total) ⇒ Se refiere a las personas que viven en áreas urbanas según lo definen las oficinas nacionales de estadística.

5.2 Modelo de Base de Datos

Así como definen los autores del manual [Graph Databases](#), los modelos sirven para abstraer características de un dominio ingobernable a un espacio donde se puedan manipular. En concreto, lo único que distingue el modelado de grafos de otras técnicas más tradicionales es la mayor afinidad de los primeros entre los modelos lógicos y físicos.

El modelado de ésta base de datos en particular se puede considerar uno de los puntos más críticos del proyecto. Esto es debido a la complejidad añadida de modelar el

tiempo como parte del grafo manteniendo la consistencia entre los nodos y sin perder información necesaria, pero a su vez adaptándolo a las consultas que se van a realizar.

La solución propuesta se extrae de un ejemplo de modelado de un conjunto de datos de vuelos de aerolíneas [27], el cual extrapolado al caso de datos de asilo humanitario a lo largo de los años, se representa así:

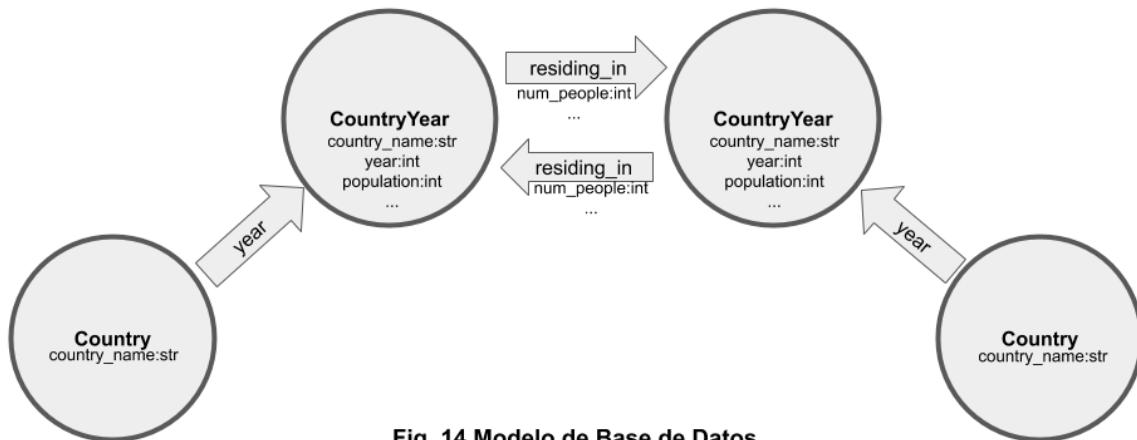


Fig. 14 Modelo de Base de Datos

En lugar de que exista un único tipo de nodo `Country` como inicialmente se proponía, ahora se suma un segundo `CountryYear`, representando a un país en un año concreto. Los tipos de relaciones también aumentan, de existir un único tipo de relación `RESIDE_IN` que conectaba a los países, aparecen tantos tipos nuevos de relación como años `year` con datos disponibles.

5.3 Análisis exploratorio de Datos

Utilizando el API de Py2neo se realizan una serie de consultas sencillas sobre el grafo obteniendo los siguientes resultados:

- Total del número de nodos $\Rightarrow 7376$
- Tipos de nodos \Rightarrow 'Country', 'CountryYear'
- Total del número de relaciones $\Rightarrow 124501$
- Total de tipos de relaciones $\Rightarrow 56 =$ 'RESIDE_IN' + todos los años con datos disponibles
- Total de años con datos disponibles $\Rightarrow 58$
- Total del número de combinaciones entre todos los países y años con datos disponibles $\Rightarrow 7154$

El resto de consultas dejan de lado Py2neo para utilizar únicamente el díver de Python de Neo4j, el cual permite mayor flexibilidad a la hora de diseñar las sentencias de Cypher.

El primer gráfico con datos agregados que se presenta muestra la tendencia al alza con el paso de los años del total de población en riesgo. Cifra total, que se ha duplicado en los últimos 4 años, llegando en 2017 a rozar los 70 Millones de personas.

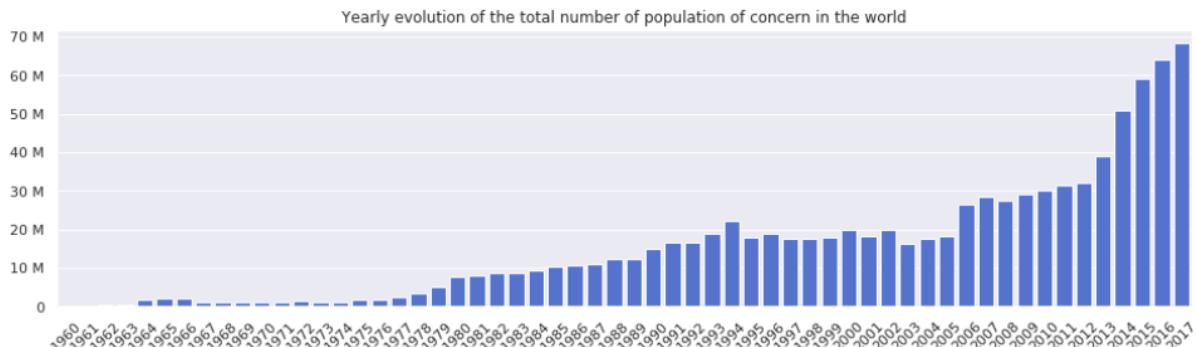


Fig. 15 Evolución Anual del total de desplazados en el mundo

Desglosando el total de población en riesgo, se observa la clara diferencia entre el número de desplazados internos, refugiados y casos pendientes a lo largo de los años:

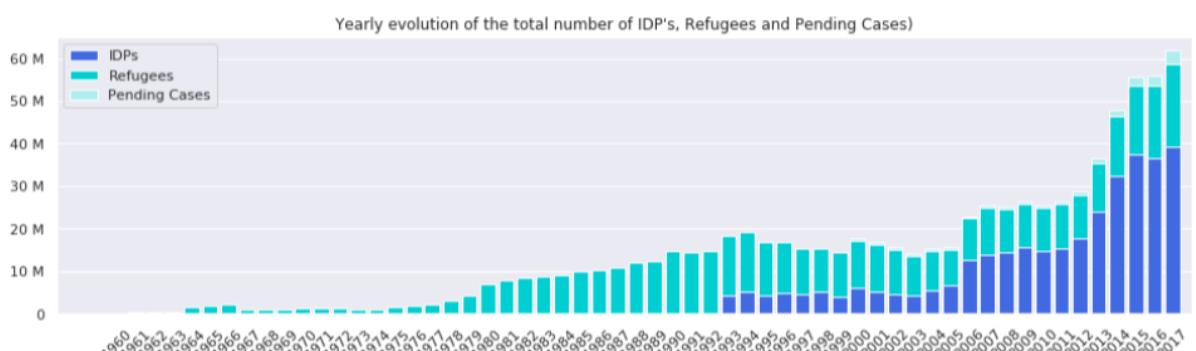


Fig. 16 Evolución Anual del número de IDP, refugiados y casos pendientes en el mundo

El análisis continúa obteniendo un listado para cada año con el agregado del total de afectados para cada país (de residencia y de origen).

Ordenando los resultados de manera descendente y limitando los resultados de la consulta se obtiene el top (para cada año) de los países con mayor número de afectados.

- Top 3 países de residencia para cada año:

Top 3 countries where the highest number of population of concern have resided in through the years.
Size of bubble represents the number of refugees

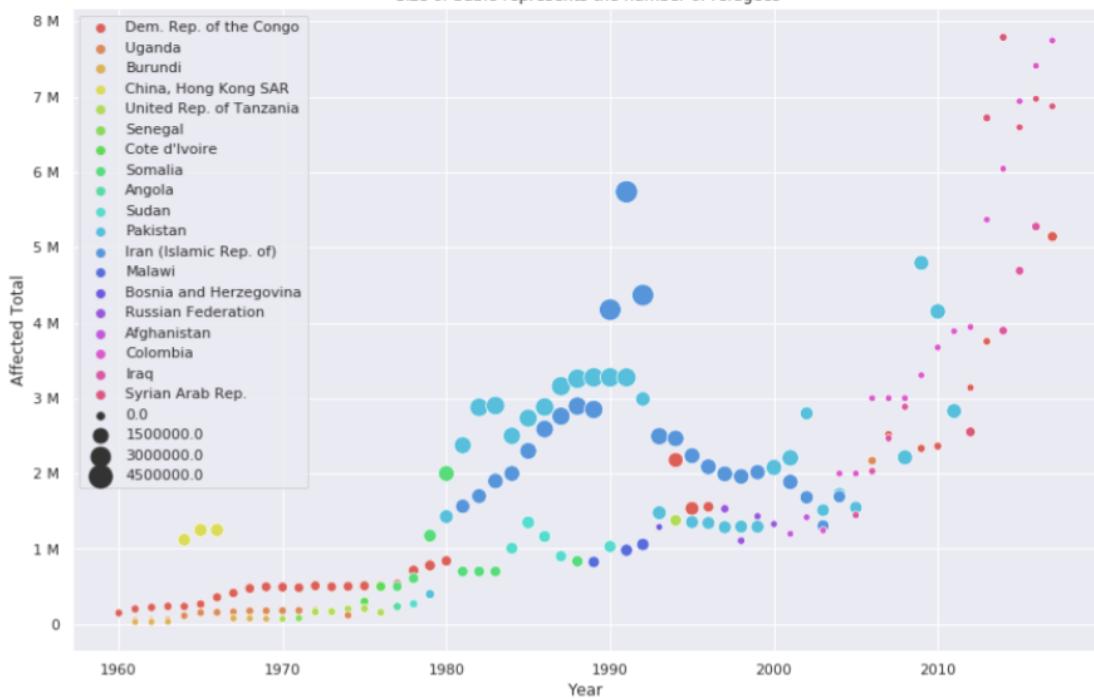


Fig. 17 Top 3 países de residencia con el mayor número de desplazados

- Top 3 países de origen para cada año:

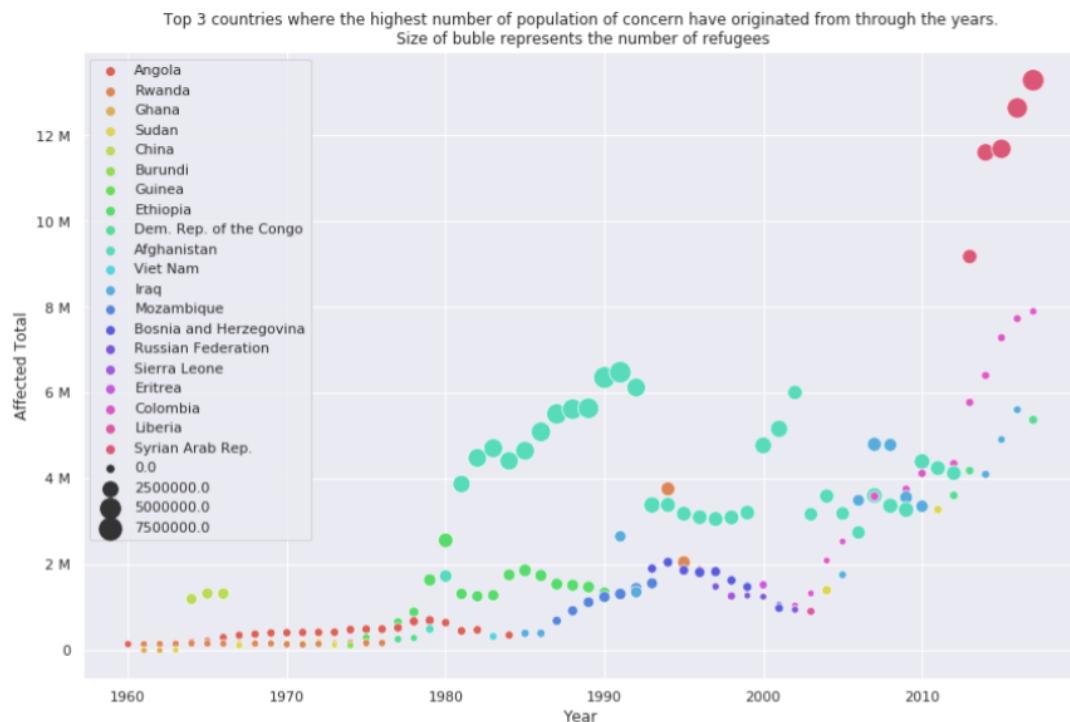


Fig. 18 Top 3 países de origen con el mayor número de desplazados

Merece la pena listar los 10 países de origen con mayor número de población en riesgo y ver la evolución de la cifra total durante los tres últimos años:

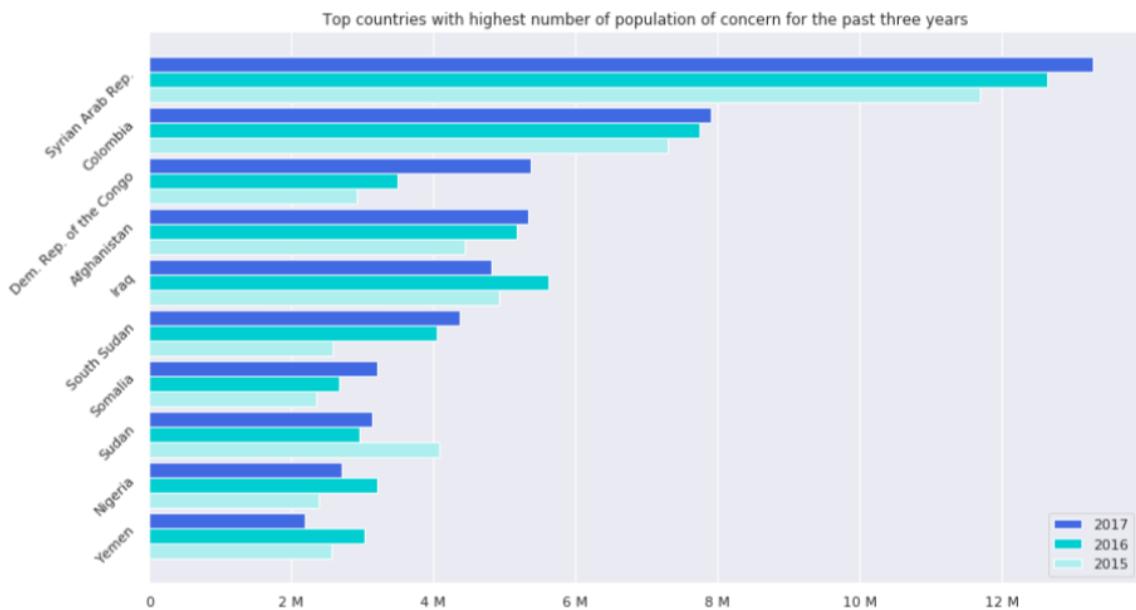


Fig. 19 Los 10 países de origen con el mayor número de desplazados durante los últimos 3 años

Si en cambio se ordena el listado con el total de afectados para cada año de manera ascendente y se limita el número de resultados por año a un único valor, lo que se saca es el país de residencia/origen con el menor número de población en riesgo por cada año:

- País de residencia

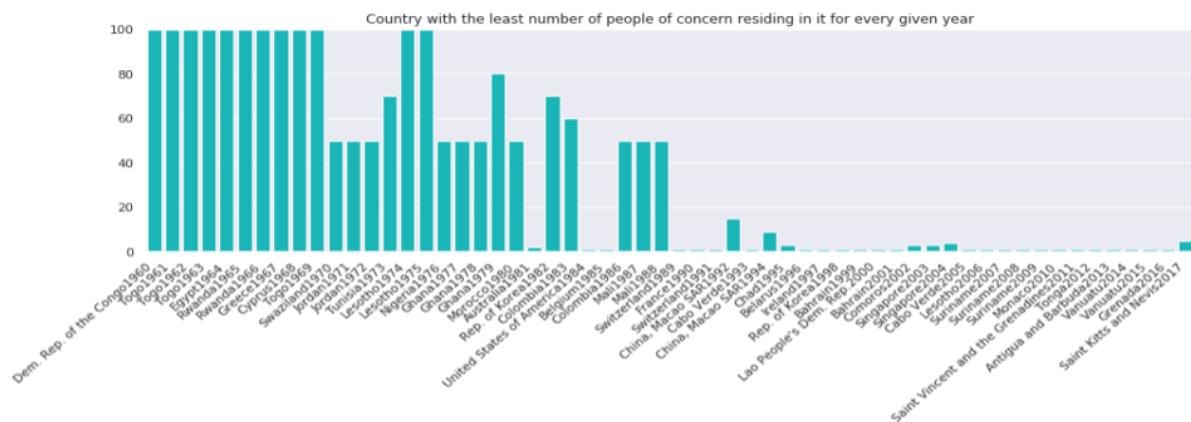


Fig. 20 Países de residencia con el menor número de desplazados

- País de origen

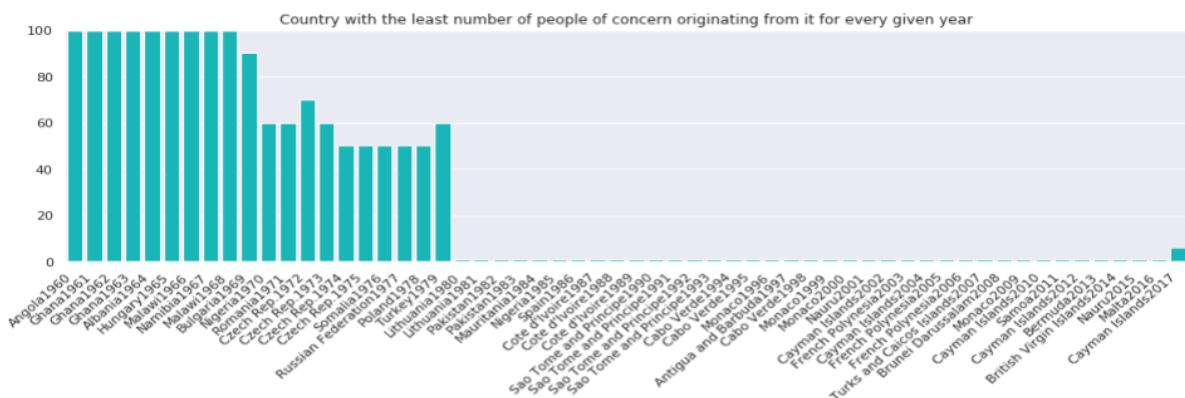


Fig. 21 Países de origen con el menor número de desplazados



Lo siguiente es poder conocer más a fondo la situación de un país específico para un año en concreto. En los resultados se muestra el listado de países con población en riesgo con origen/residencia en España en 2017.

- Top 10 de países de residencia con país de origen España

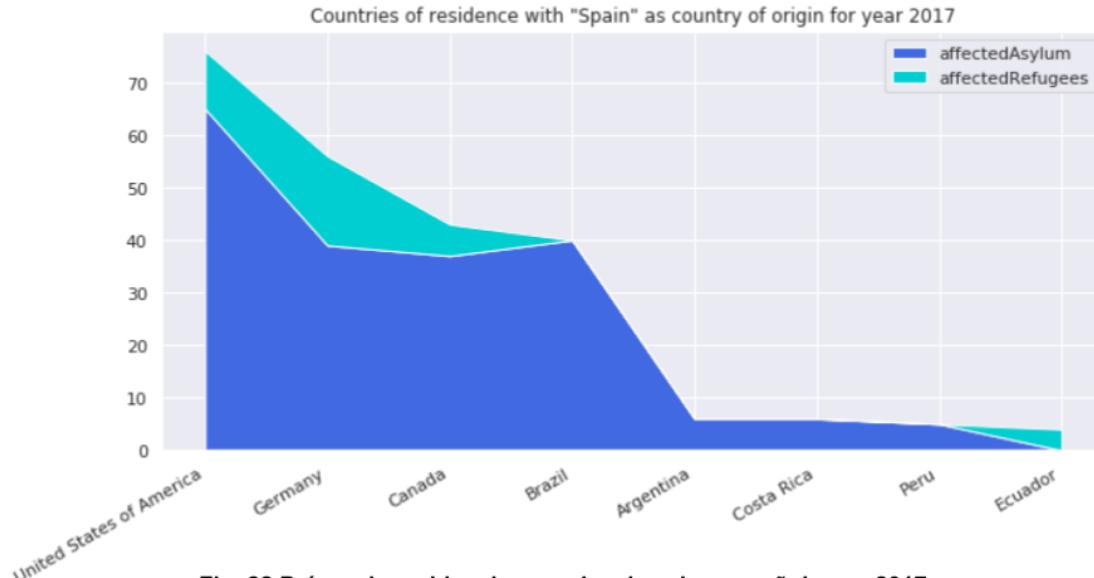


Fig. 22 Países de residencia para desplazados españoles en 2017

- Top 10 de países de origen con país de residencia España

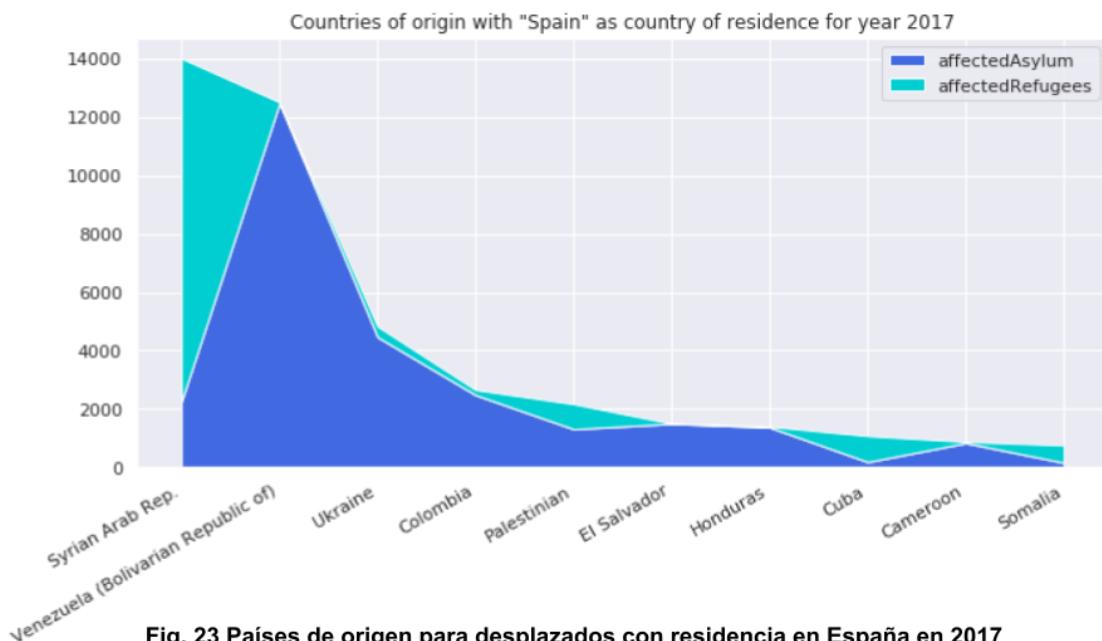


Fig. 23 Países de origen para desplazados con residencia en España en 2017

Resulta interesante conocer la diferencia entre el número de refugiados (solicitudes concedidas) y el número de solicitantes de asilo (solicitudes pendientes) que residen en un

país. Con dicho fin se calcula la media a lo largo de los años de la diferencia entre el total de ambas propiedades de la relación.

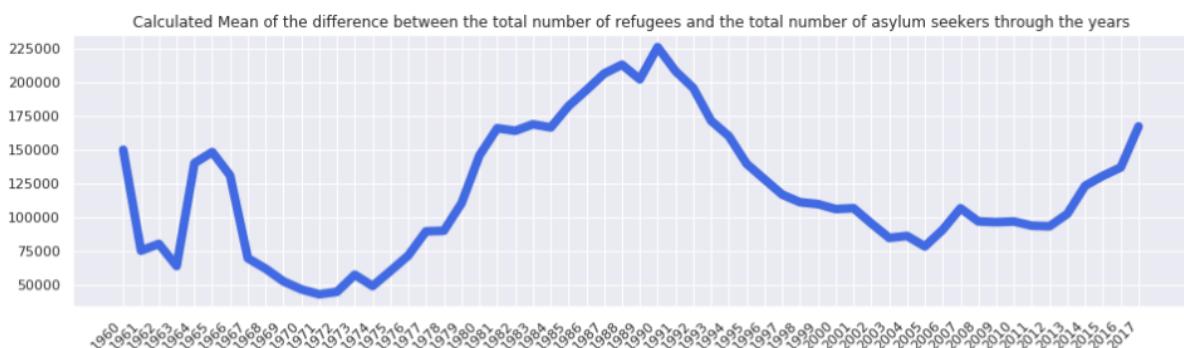


Fig. 24 Media para cada año de la resta entre el número de refugiados y el de casos pendientes

Consecuentemente se obtienen los países con más número de solicitudes pendientes que de concedidas (valor negativo) y los países con menos (valor positivo). Se representan los resultados de los últimos cuatro años.

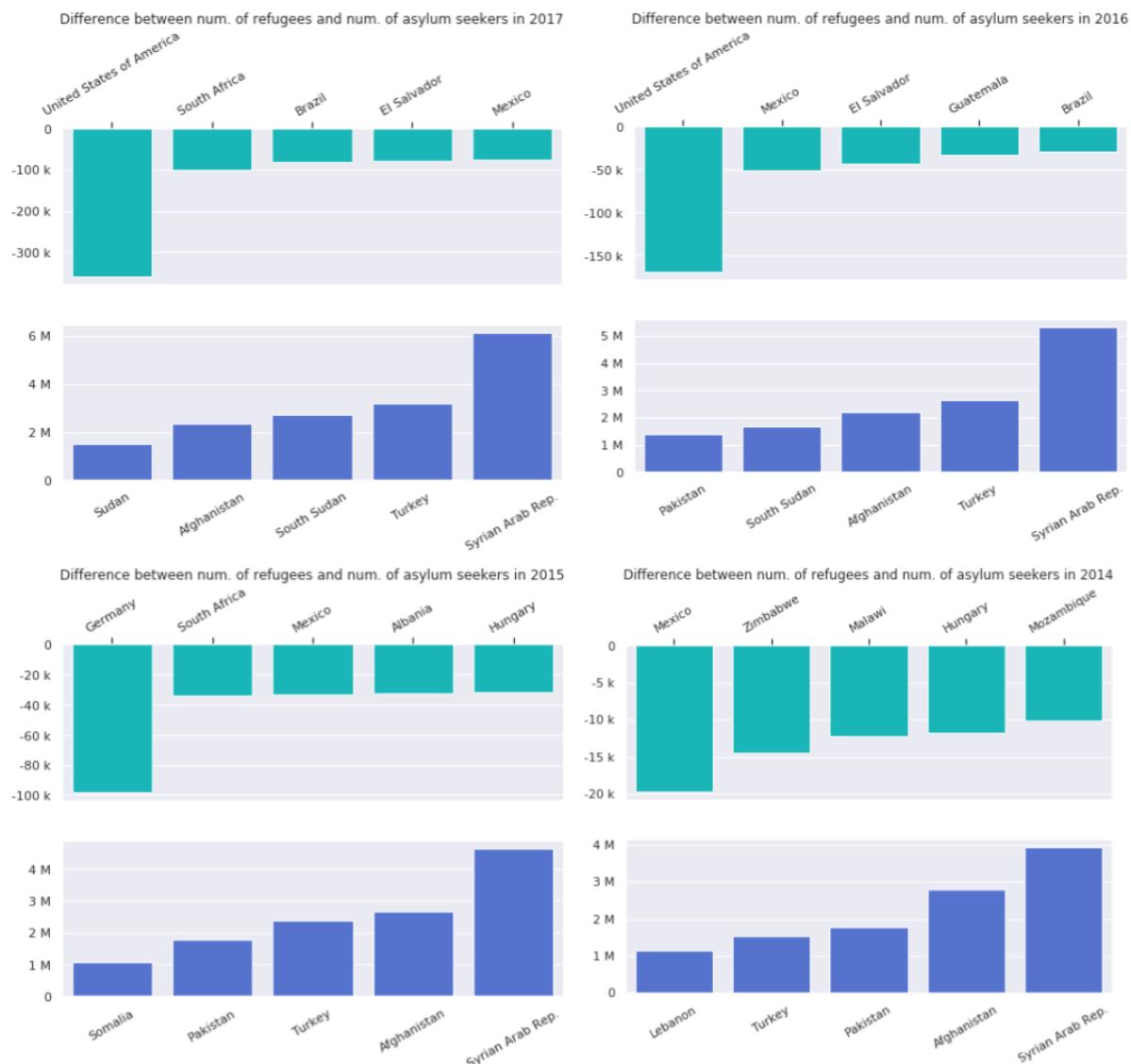


Fig. 25 Países con mayor y menor diferencia entre el número de refugiados y casos pendientes

Por último se estudian el top 3 para cada año de países con el mayor número de desplazados internos.

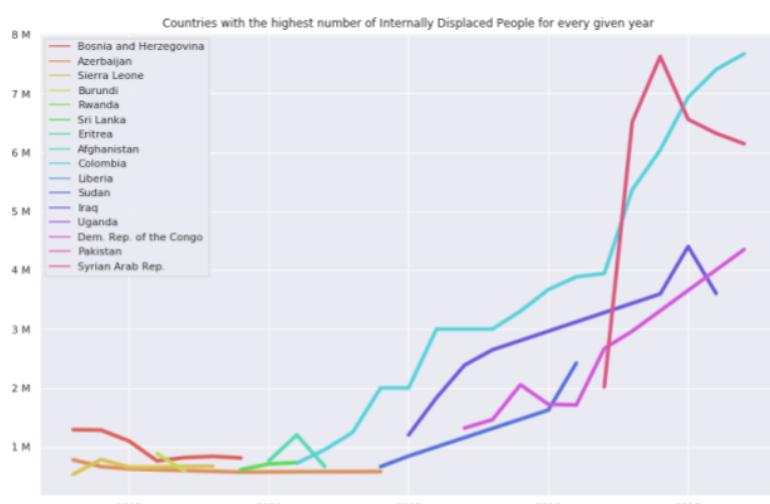


Fig. 26 Top 3 países con mayor número de IDP



6 ALGORITMOS

Por la naturaleza de los datos se considera que los algoritmos a aplicar más adecuados son los de Centralidad y Detección de comunidades.

6.1 Centralidad

Antes de explicar los algoritmos de centralidad ofrecidos por Neo4j se decide calcular el grado de centralidad para todos los nodos de cada subgrafo con una consulta de Cypher.

Lo que se representa a continuación es el histograma y boxplot del grado de centralidad de todos los nodos para el año 2017:

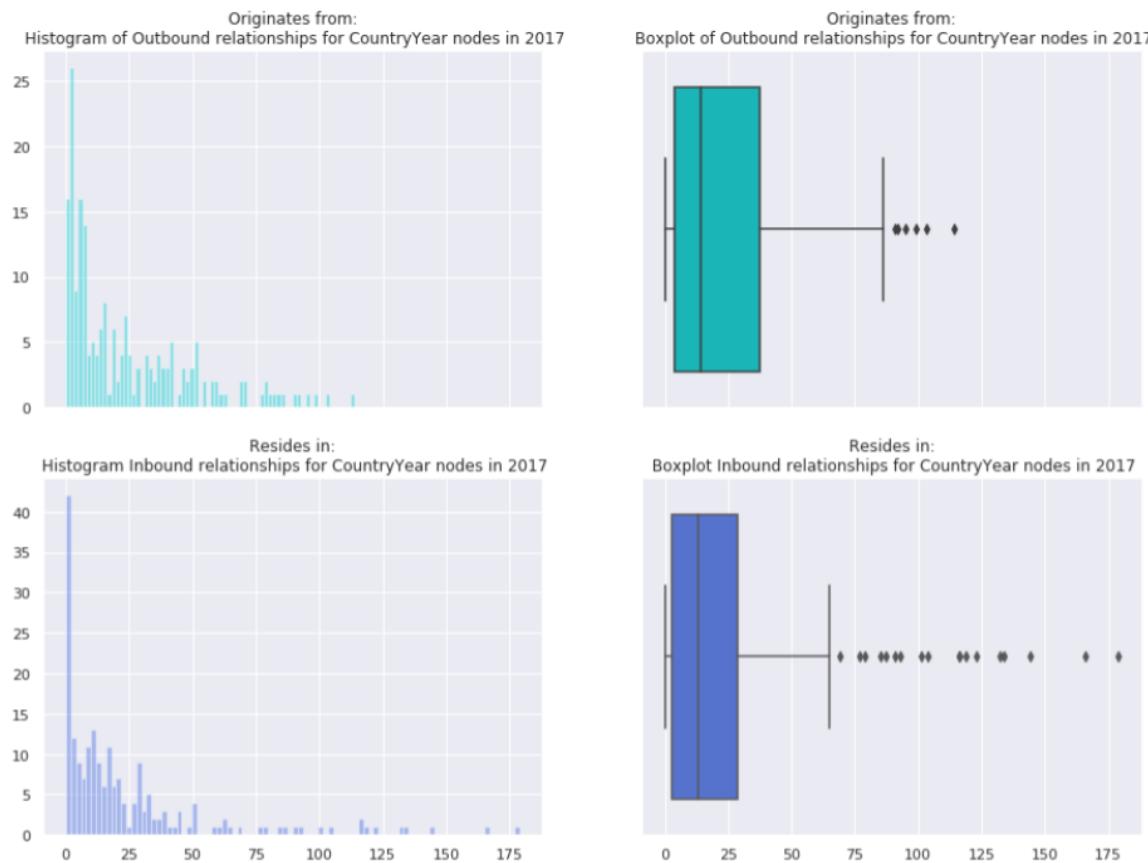


Fig. 27 Histograma y Boxplot del número de relaciones de los nodos (año 2017)

Ejecutando la query anterior para todos los años con datos disponibles se elabora el gráfico siguiente, que muestra la media calculada del grado de centralidad de los nodos a lo largo de los años.

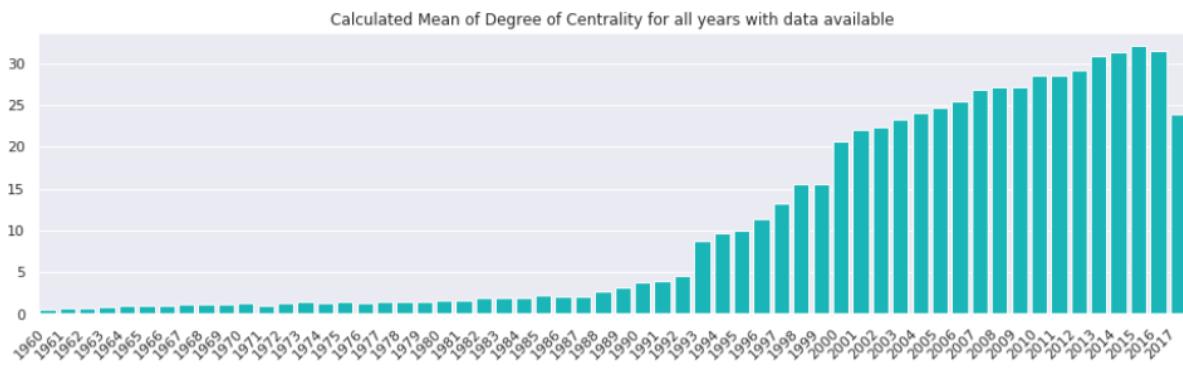


Fig. 28 Media calculada del Grado de Centralidad de los nodos para cada año

De los distintos tipos de algoritmos de centralidad que ofrece Neo4j se decide probar con los siguientes:

- Degree Centrality. Será utilizado para definir los nodos más populares en el grafo ya que se calcula contando el número de relaciones que entran y salen del mismo.
- Betweenness Centrality. Teniendo en cuenta el peso de las relaciones, primero calcula el camino más corto entre cada par de nodos del grafo para finalmente asignar una puntuación a cada nodo en función del número de caminos más cortos que pasan por el. Midiendo así la influencia de un nodo sobre el flujo que circula por el grafo.

6.2 Detección de Comunidades

Para esta aplicación, el objetivo de aplicar algoritmos de detección de comunidades a los subgrafos generados, es ver si es posible realizar una clasificación de países en lo que a asilo humanitario se refiere. Concretamente se hará uso de:

- Strongly Connected Components. Se probará este algoritmo por tratarse de un grafo con relaciones dirigidas buscando obtener clusters que puedan requerir ser tratados de manera independiente.
- Louvain. Este algoritmo encuentra clusters comparando la densidad de comunidades tras asignar nodos a diferentes grupos. Prueba varios agrupamientos con el fin de encontrar un óptimo global.

7 RESULTADOS

Como aclaración, en lugar de ejecutar los algoritmos explicados previamente sobre el grafo entero, lo que se hará es crear subgrafos filtrando por el año. De cada uno de esos subgrafos se obtendrán medidas de centralidad y agrupaciones por comunidad semejantes. En esta memoria se mostrarán de manera visual (utilizando neovis.js) los resultados obtenidos, para cada uno de los algoritmos utilizados, para el año 2017.

En lo que a la visualización respecta, el grosor de las fechas que representan las relaciones RESIDE_IN, se corresponde con el número de desplazados totales entre dichos nodos.

7.1 Degree Centrality

Se calcula y representa el grado de centralidad de cada uno de los nodos del subgrafo correspondiente a los datos de 2017. A mayor tamaño de los nodos, mayor será el número calculado del grado de centralidad.

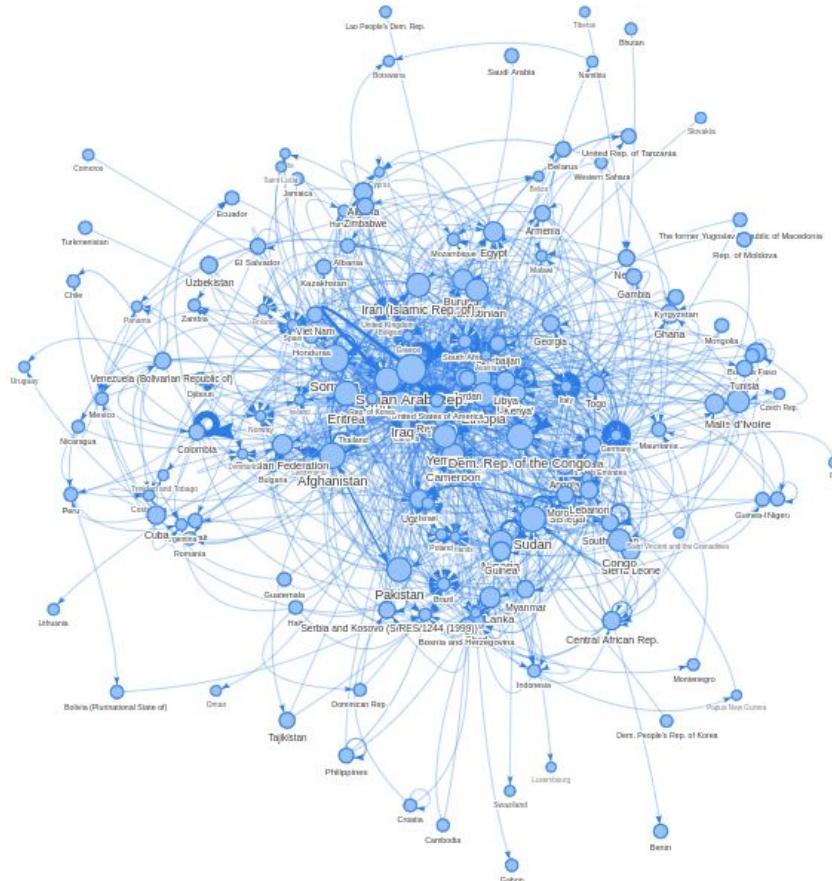


Fig. 29 Centrality 2017

Destacan por tamaño países como Siria, Afganistán, Pakistán, Arabia Saudí, Sudán o Somalia. Todos ellos países de África y Oriente Medio con conflictos en activo en 2017 o vecinos de países en guerra [\[30\]](#).



7.2 Betweenness Centrality

Los resultados obtenidos tras aplicar el algoritmo Betweenness Centrality para el año 2017 son los siguientes.

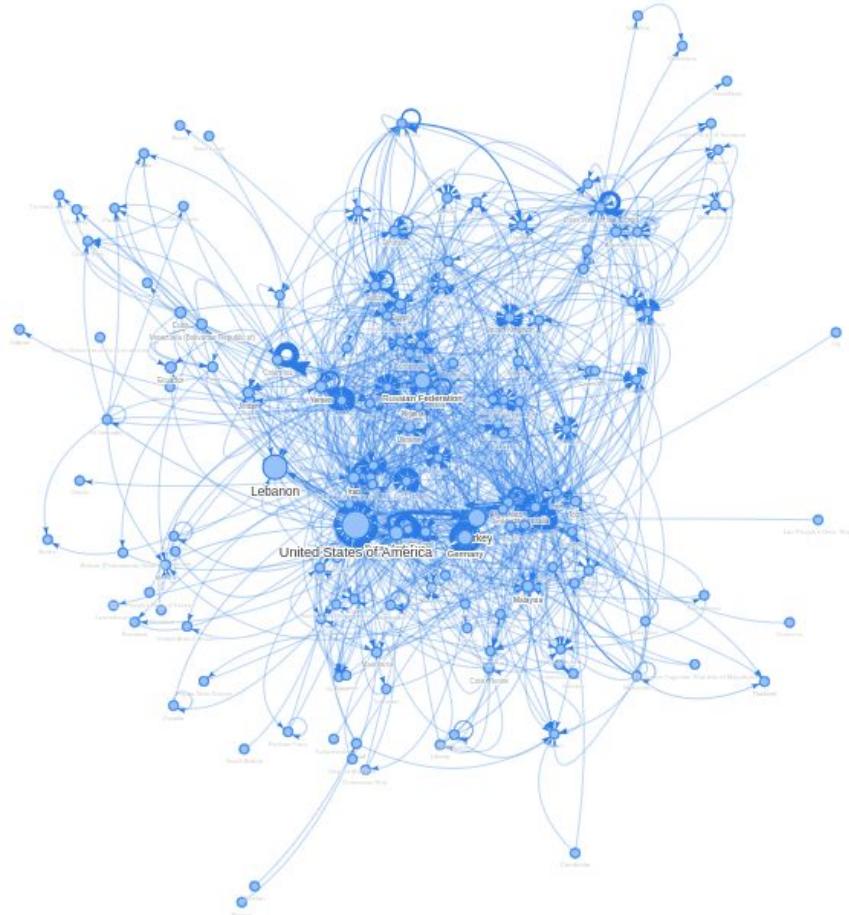


Fig. 30 Betweenness 2017

En este caso se observa una mayor diferencia en tamaños. Con un nodo (Estados Unidos) mucho más grande en proporción al resto.

7.3 Strongly Connected Components

Como se observa en la siguiente figura, no se considera que este algoritmo arroje resultados concluyentes para el subgrafo de 2017.

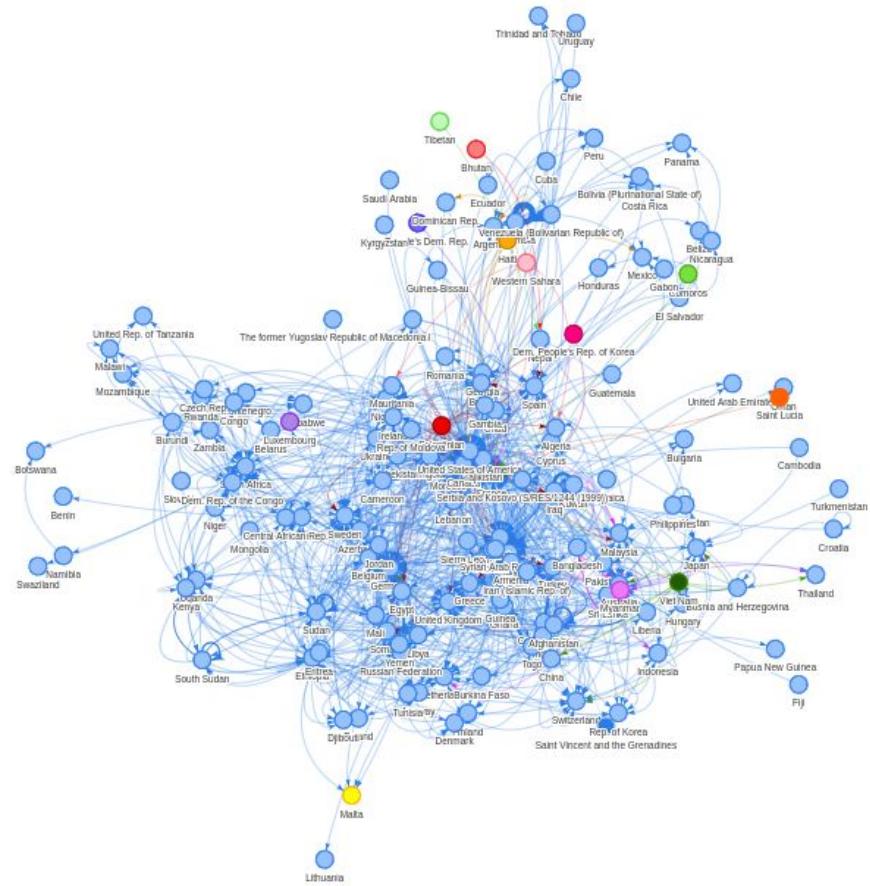


Fig. 31 SCC 2017

7.4 Louvain

La visualización obtenida en este caso parece más adecuada a los datos de partida.

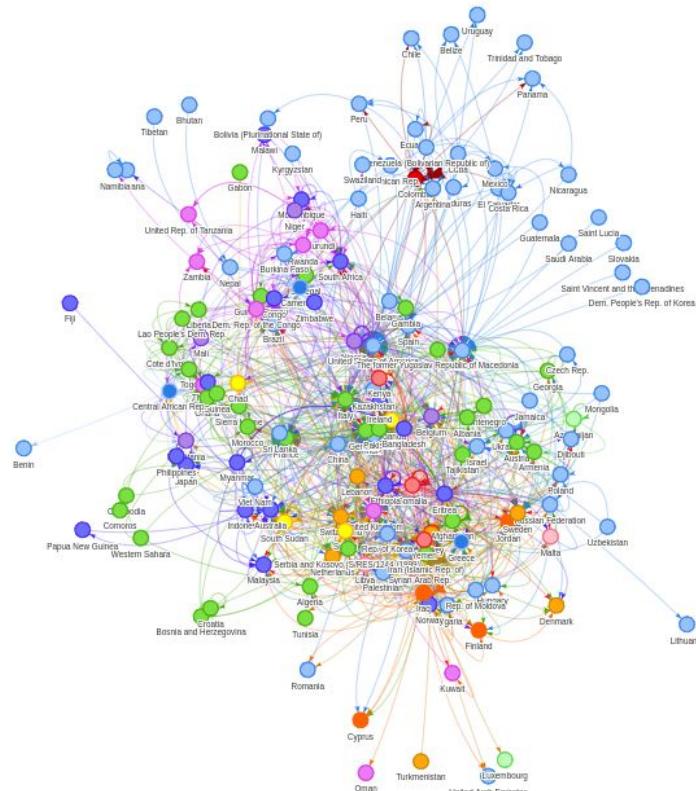


Fig. 32 Louvain 2017

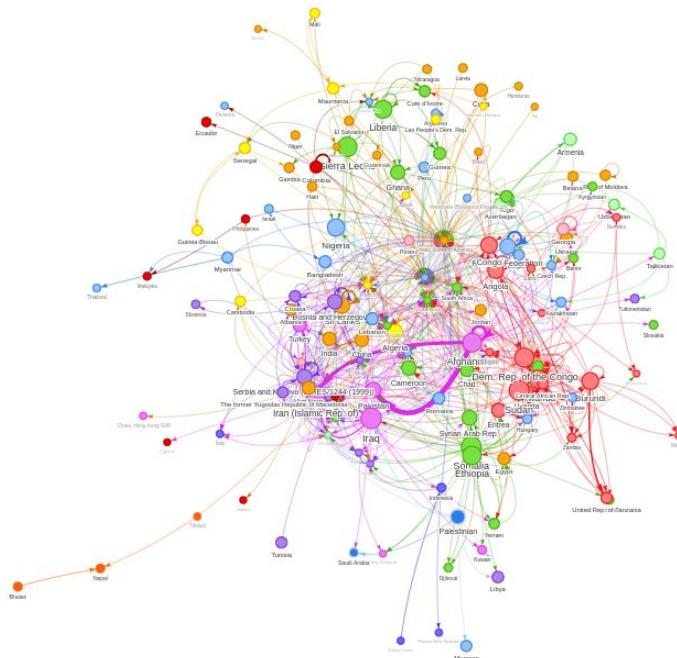
Se observa que alguna de las comunidades detectadas reúne a países próximos geográficamente. Por ejemplo, a la izquierda de la imagen se encuentra un grupo de color azul oscuro que engloba estados del sudeste asiático y oceanía: Papúa Nueva Guinea, Filipinas, Indonesia, Malasia, Australia, Myanmar. También se advierte otro cluster de nodos de color rosa con países del sudeste africano: Tanzania, Ruanda, Burundi, Zambia o República del Congo.



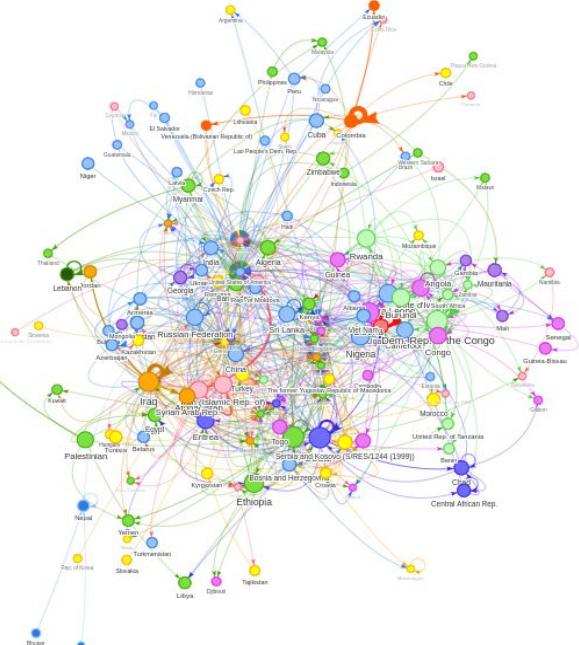
8 VISUALIZACIÓN

Para mostrar la evolución temporal del Grafo de Asilo Humanitario se decide utilizar los algoritmos Degree Centrality (variación en el tamaño de los nodos en función del grado de centralidad) y Louvain (color en los nodos representando cada una de las comunidades detectadas). El grosor de las relaciones representa el número total de desplazados del país de origen al país de residencia.

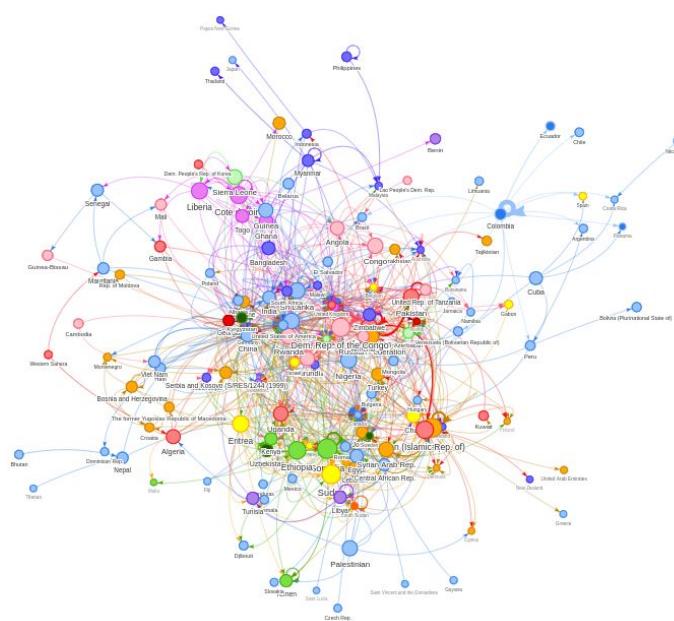
Asylum Seekers in 2001



Asylum Seekers in 2006



Asylum Seekers in 2011



Asylum Seekers in 2017

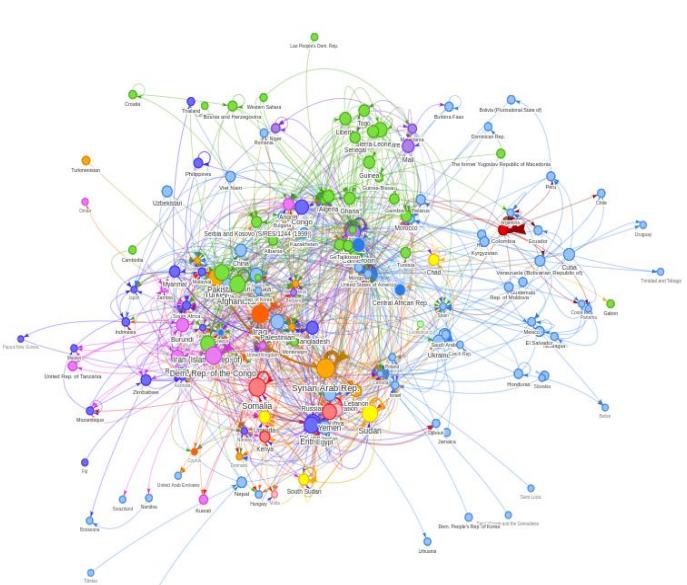


Fig. 33 Evolución Temporal

9 MEJORAS FUTURAS

Son varios los puntos de este trabajo que, a futuro, podrían ser desarrollados en mayor profundidad. Se proponen los siguientes:

- Implementar validaciones de los datos mediante reglas de calidad. Tanto en los datos de origen como en los datos procesados antes de ser cargados en base de datos.
- Arreglar el cruce de datos entre los ficheros de UNHCR y World Bank para que no haya países sin datos de Indicadores del Banco Mundial.
- Analizar los datos de los ficheros con la relación ORIGINATE_FROM y compararlos con los de RESIDE_IN.
- Introducir algoritmos de Graphframes y evaluar las posibles diferencias resultantes con los algoritmos utilizados de Neo4j.
- Mejorar la visualización y presentación de resultados.
- Testear las funciones definidas en el proceso python y la performance las queries Cypher sobre Neo4j.

10 CONCLUSIONES

Finalmente se listan, de nuevo, los Hitos definidos en el apartado de [Objetivos](#). Indicando con un CHECK si se dan por conseguidos en este proyecto:

HITO	CHECK	Comentario
Control de versiones de código y de datos	✓	Uso adecuado de Git y DVC
Empaquetar la aplicación	✓	Imágenes docker para cada servicio orquestadas con docker compose
Obtener programáticamente datos abiertos de distintas fuentes	✓	Datos de UNHCR y World Bank obtenidos con el API de HDX
Limpiar los datos e integrarlos para generar los ficheros a cargar	✓	Elaborados ficheros de carga de nodos con relaciones adecuadas
Importar los datos procesados en la base de datos	✓	Carga en base de datos del grafo completo en pocos segundos
Realizar consultas sobre los datos y aplicar algoritmos	✓	Ánálisis exploratorio extenso y visualización basada en algoritmos
Ejecutar en entorno Cloud	✓	Aplicación ejecutada correctamente en Google Cloud



11 ANEXO I. Servidor nginx para Neovis.js

Aunque es posible visualizar los archivos .html de Neovis.js con un simple navegador, también se ofrece la posibilidad de levantar un contenedor docker con un servidor de nginx.

Para ello se crea un nuevo fichero YAML docker-compose.web.yml que incluye, además de los servicios de neo4j y jupyter notebooks, el servicio de nginx basado en la imagen de alpine:

```
nginx-hdx:
  image: nginx:alpine
  container_name: nginx-hdx-container
  volumes:
    - ./neovis-js:/usr/share/nginx/html/:ro
  ports:
    - "8080:80"
```

En este caso se monta, en el contenedor, el volumen del directorio del proyecto en el que se han guardado los archivos .html con las visualizaciones de neovis.js y se bindean los puertos 8080 y 80.

Por comodidad, también se incluyen en el Makefile los comandos build-web, up-web y down-web que construyen, levantan y paran los servicios definidos en el fichero YAML de docker-compose.web.yml.

12 ANEXO II. Pasos para Ejecutar la Aplicación en Google Cloud

En este apartado se listan los pasos a seguir para ejecutar la aplicación completa en una instancia de Google Cloud:

- Conectarse por ssh a la instancia de google cloud hdx
- Clonar el repositorio de código ⇒

```
git clone  
https://github.com/CandelaV/UNHCRs_population_of_concern_HDX.git
```

- Descargar los datos desde el repositorio remoto ⇒

```
dvc pull
```

- Dar permisos para poder editar notebooks ⇒

```
sudo make chmod 777 -R notebooks/
```

- Levantar los servicios construyendo las imágenes ⇒

```
sudo make build-web
```

- Cargar datos en Neo4j ⇒

```
sudo make load-neo4j-residing-all
```

- Abrir Jupyter Lab ⇒ en el navegador ir a `http://<EXTERNAL_IP>:8888`

- Abrir consola de Neo4j ⇒ en el navegador ir a `http://<EXTERNAL_IP>:7474`

- Ejecutar notebooks ⇒ en la carpeta `notebooks/` abrir `c_data_analysis_with_python_and_neo4j_RESIDE_IN.ipynb` y `d_algorithms_with_python_and_neo4j.ipynb` y ejecutar todas las celdas `Run>Run All Cells`

- Visualización ⇒ en el navegador acceder al servidor web en `http://<EXTERNAL_IP>:8080`

NOTA: También es posible ejecutar la aplicación en un ordenador personal, siempre y cuando éste disponga de las prestaciones adecuadas (p.e. ordenador con 16 GB de RAM y S.O. Ubuntu). En este caso, en lugar de conectarse a una `<EXTERNAL_IP>` habría que hacerlo a `localhost`



13 REFERENCIAS

A continuación se listan todas las referencias a artículos, libros o páginas web que se citan en esta memoria:

- [1] "Microsoft Certified: Azure Data Scientist Associate." <https://www.microsoft.com/en-us/learning/azure-data-scientist.aspx>.
- [2] "A Beginner's Guide to the Data Science Pipeline – Towards Data" 15 ene.. 2018, <https://towardsdatascience.com/a-beginners-guide-to-the-data-science-pipeline-a4904b2d8ad3>.
- [3] Definition, O. (2017). Open definition 2.1.
- [4] Coulton, C. J., Goerge, R., Putnam-Hornstein, E., & de Haan, B. (2015). Harnessing big data for social good: A grand challenge for social work. Cleveland: American Academy of Social Work and Social Welfare, 1-20.
- [5] "Humanitarian Data Exchange: Welcome." <https://data.humdata.org/>.
- [6] Kosho, J. (2016). Media Influence on Public Opinion Attitudes Toward the Migration Crisis. International Journal of Scientific & Technology Research, 5(05), 86-91.
- [7] "A quién ayudamos - ACNUR." <https://www.acnur.org/a-quien-ayudamos.html>.
- [8] "¿Qué es un HITO de un proyecto? | Sinnaps - Project Management." <https://www.sinnaps.com/blog-gestion-proyectos/hito-la-gestion-proyectos>.
- [9] Press, G. (2016). Cleaning big data: Most time-consuming, least enjoyable data science task, survey says. Forbes Magazine.
- [10] "What is Data Wrangling? | 6 Steps in Data Wrangling - Acadgild." 17 dic.. 2018, <https://acadgild.com/blog/6-steps-in-data-wrangling>.
- [11] "Descriptive, Predictive, and Prescriptive Analytics Explained." <https://halobi.com/blog/descriptive-predictive-and-prescriptive-analytics-explained/>.
- [12] "Neo4j's Graph Query Language: An Introduction to Cypher." <https://neo4j.com/developer/cypher-query-language/>.
- [13] "neo4j-contrib/neovis.js: Neo4j + vis.js = neovis.js. Graph ... - GitHub." <https://github.com/neo4j-contrib/neovis.js/>.
- [14] "What is a Container? | Docker." <https://www.docker.com/resources/what-container>.
- [15] "Example Use Cases of Docker in the Data Science Process | Jens" 4 abr.. 2019, <https://jenslaufer.com/data/science/use-cases-of-docker-in-the-data-science-process.html>.
- [16] "Overview of Docker Compose | Docker" <https://docs.docker.com/compose/overview/>.
- [17] "Docker image for Neo4j 2.x - Neo4j Graph Database Platform." <https://neo4j.com/developer/docker-23/>.
- [18] "Jupyter Docker Stacks — docker-stacks latest documentation." <https://jupyter-docker-stacks.readthedocs.io/en/latest/>.
- [19] "The Py2neo v4 Handbook — The Py2neo v4 Handbook." <https://py2neo.org/>.
- [20] "Sessions & Transactions — Neo4j Bolt Driver 1.7 for Python." <https://neo4j.com/docs/api/python-driver/current/transactions.html>.
- [21] "nbdime – diffing and merging of Jupyter Notebooks ... - Read the Docs." <https://nbdime.readthedocs.io/en/latest/>.
- [22] Driessen, V. (2010). A successful Git branching model. URL <http://nvie.com/posts/a-successful-git-branching-model>.
- [23] "Machine Learning Version Control System · DVC." <https://dvc.org/>.
- [24] "AWS Vs Azure Vs Google: Cloud Services Comparison ... - Whizlabs." 17 may.. 2018, <https://www.whizlabs.com/blog/aws-vs-azure-vs-google/>.

[25] "Running Jupyter Notebook on Google Cloud Platform in 15 min." 8 sept.. 2017, <https://towardsdatascience.com/running-jupyter-notebook-in-google-cloud-platform-in-15-min-61e16da34d52>.

[26] "What is a Makefile and how does it work? | Opensource.com." 22 ago.. 2018, <https://opensource.com/article/18/8/what-how-makefile>.

[27] "Modeling Airline Flights in Neo4j | Max De Marzi." 26 ago.. 2015, <https://maxdemarzi.com/2015/08/26/modeling-airline-flights-in-neo4j/>.

[28] Robinson, I., Webber, J., & Eifrem, E. (2015). Graph databases: new opportunities for connected data. " O'Reilly Media, Inc.".

[29] Mark Needham & Amy E.Hodler (2019). Graph Algorithms. Practical Examples in Apache and Neo4j. " O'Reilly Media, Inc.".

[30] List of armed conflicts in 2017

https://en.wikipedia.org/wiki/List_of_armed_conflicts_in_2017

También se incluye un listado de las referencias de las figuras que no han sido elaboradas:

- Fig. 2 : <http://www.kalisch.biz/projects/humanitarian-data-exchange-hdx/>
- Fig. 5 :
<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#efa0fc76f637>
- Fig. 7 :
<https://www.oreilly.com/library/view/graph-databases-2nd/9781491930885/ch01.html>
- Fig. 10 : <https://nvie.com/posts/a-successful-git-branching-model/>
- Fig. 11 : <https://dvc.org/doc/use-cases/share-data-and-model-files>
- Fig. 12 :
<https://www.whizlabs.com/blog/wp-content/uploads/2018/05/aws-vs-azure-vs-google-infographic.jpg>



14 RELACIÓN DE FIGURAS

- Fig. 1 PIPELINE de DATOS
- Fig. 2 Proceso HDX
- Fig. 3 Modelo Inicial del Grafo
- Fig. 4 Hitos del Proyecto
- Fig. 5 Reparto del Tiempo de un Data Scientist
- Fig. 6 Subtareas del proceso de Data Wrangling
- Fig. 7 Comparativa de Bases de Datos orientadas a Grafos
- Fig. 8 Arquitectura de la aplicación
- Fig. 9 Evolución desde Dockerfile hasta Contenedor Docker
- Fig. 10 Git Flow del Proyecto
- Fig. 11 Caso de uso DVC
- Fig. 12 Comparativa de Proveedores de Servicios Cloud
- Fig. 13 Fuentes de Datos en bruto
- Fig. 14 Modelo de Base de Datos
- Fig. 15 Evolución Anual del total de desplazados en el mundo
- Fig. 16 Evolución Anual del número de IDP, refugiados y casos pendientes en el mundo
- Fig. 17 Top 3 países de residencia con el mayor número de desplazados
- Fig. 18 Top 3 países de origen con el mayor número de desplazados
- Fig. 19 Los 10 países de origen con el mayor número de desplazados durante los últimos 3 años
- Fig. 20 Países de residencia con el menor número de desplazados
- Fig. 21 Países de origen con el menor número de desplazados
- Fig. 22 Países de residencia para desplazados españoles en 2017
- Fig. 23 Países de origen para desplazados con residencia en España en 2017
- Fig. 24 Media para cada año de la resta entre el número de refugiados y el de casos pendientes
- Fig. 25 Países con mayor y menor diferencia entre el número de refugiados y casos pendientes
- Fig. 26 Top 3 países con mayor número de IDP
- Fig. 27 Histograma y Boxplot del número de relaciones de los nodos (año 2017)
- Fig. 28 Media calculada del Grado de Centralidad de los nodos para cada año
- Fig. 29 Centrality 2017
- Fig. 30 Betweenness 2017
- Fig. 31 SCC 2017
- Fig. 32 Louvain 2017
- Fig. 33 Evolución Temporal