

Instituto Tecnológico de Costa Rica

Curso: IC4302

Bases de Datos II

II Semestre 2015

Investigación

Motor de base de datos orientado a grafos

Neo4j

Profesor

Erick Hernández Bonilla

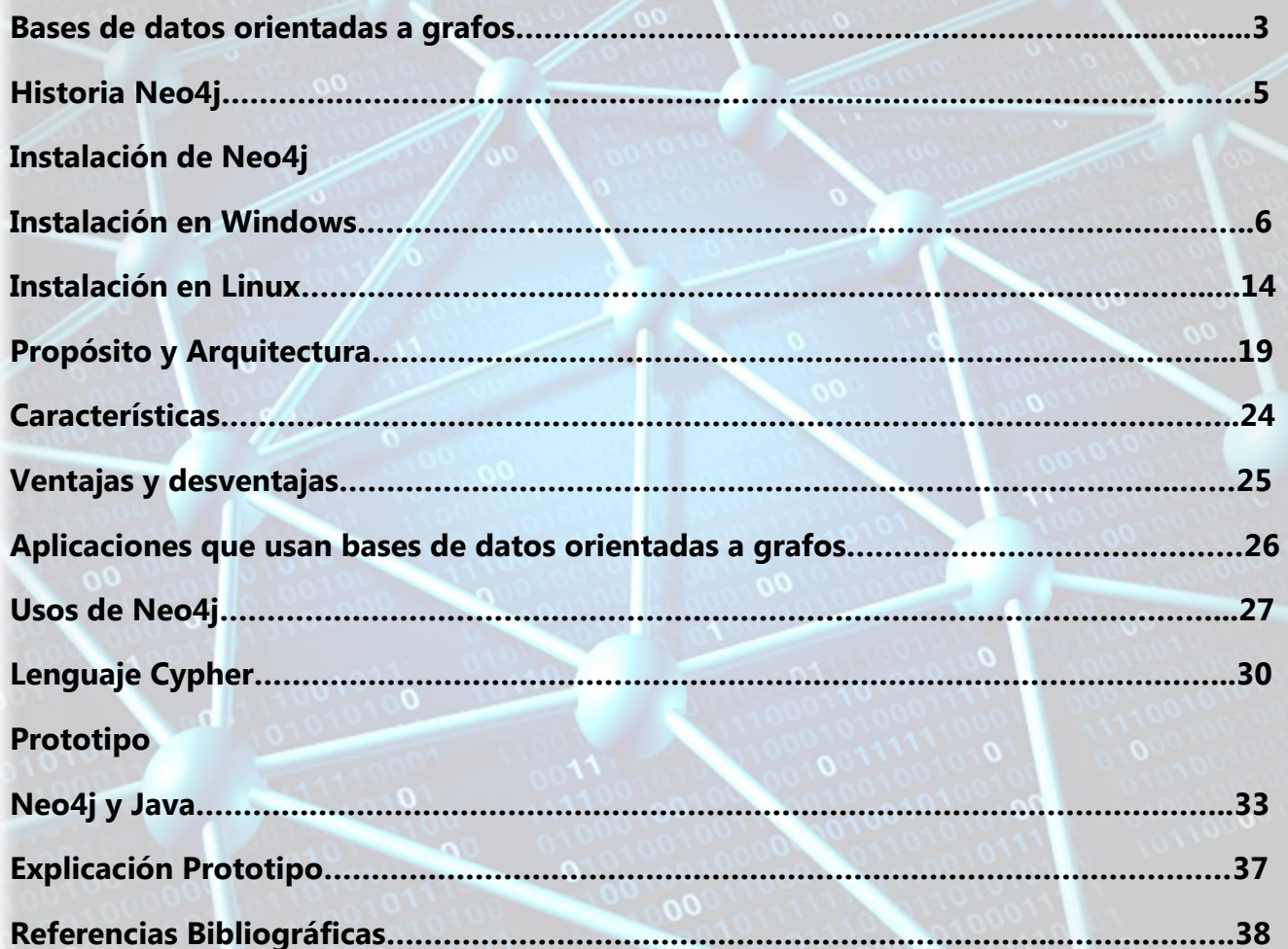
Estudiantes:

Nelson Abarca Quirós

Melissa Molina Corrales

Amanda Solano Astorga

Índice



Bases de datos orientadas a grafos.....	3
Historia Neo4j.....	5
Instalación de Neo4j	
Instalación en Windows.....	6
Instalación en Linux.....	14
Propósito y Arquitectura.....	19
Características.....	24
Ventajas y desventajas.....	25
Aplicaciones que usan bases de datos orientadas a grafos.....	26
Usos de Neo4j.....	27
Lenguaje Cypher.....	30
Prototipo	
Neo4j y Java.....	33
Explicación Prototipo.....	37
Referencias Bibliográficas.....	38

Base de datos orientada a grafos

Es una base de datos que basa su modelo de distribución de datos en grafos con nodos como vértices y relaciones como aristas y propiedades, utilizada para almacenar y representar datos.

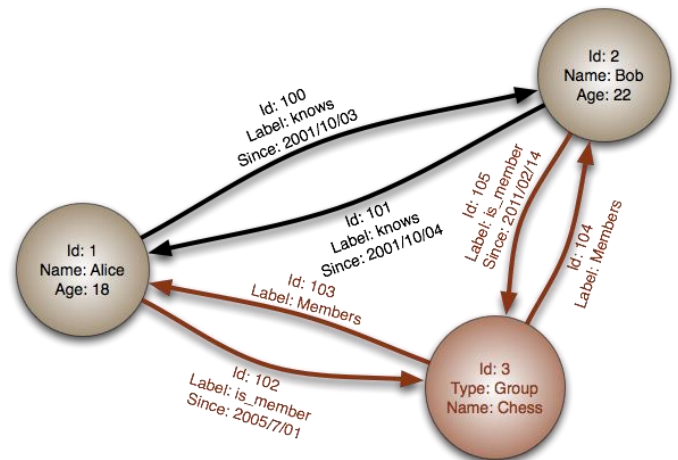


Figura 1. Grafo de una base de datos

Ventajas de una base de datos orientada a grafos

- **Eficiencia y eficacia**

Existe un aumento transparente o evidenciable de rendimiento cuando se trata de consultas de datos relacionados frente a bases de datos relacionales y soluciones No SQL. Con una base de datos orientada a grafos el rendimiento tiende a permanecer relativamente constante, lineal o directamente proporcional a la magnitud del conjunto de datos. Esto es porque las consultas están localizadas en un segmento del grafo que empieza su recorrido a partir de un nodo y continúa a través de sus vértices sin necesidad de recorrer toda una tabla o lista de índices. Por lo tanto el tiempo de ejecución para cada consulta es proporcional solo al tamaño de la parte del grafo recorrido para satisfacer esa consulta.

- **Agilidad**

Se desea evolucionar el modelo de datos, a medida que cambia la aplicación. Las bases de datos orientadas a grafos nos equipan para llevar a cabo el desarrollo y mantenimiento de sistemas sin un esquema rígido de datos. Precisamente porque son libres de esquema, las bases de datos orientadas a grafos no tienen el tipo de mecanismo de restricción de datos orientado a esquemas con el que estamos familiarizados en el mundo relacional, lo cual no significa un riesgo, sino que llama a una especie de libertad que permite hacer mucho más visible y viable la administración de los datos.

- **Flexibilidad**

Las bases de datos orientadas a grafos permiten un modelado iterativo de los datos. El modelo de datos orientado a grafos se expresa y se acomoda a las necesidades del negocio

de una manera que permite moverse a la velocidad de la situación. Los grafos son aditivos, lo que significa que podemos añadir nuevos tipos de relaciones, nuevos nodos, e incluso nuevos subgrafos a una estructura existente sin alterar las consultas existentes y la funcionalidad de la aplicación. Debido a la flexibilidad del modelo de grafos, no se tiene que modelar el dominio del problema en detalle antes de tiempo. La naturaleza aditiva de los grafos también significa que tienden a realizar menos migraciones lo que reduce gastos de mantenimiento y riesgo.

Componentes de una base de datos orientada a grafos

- Grafos
- Nodos
- Relaciones
- Propiedades

Las bases de datos orientadas a grafos acogen relaciones

Las relaciones en una base de datos forman rutas, así como los vértices en un grafo, es así que la consulta o el recorrido de un grafo implica el seguimiento de estas rutas, a diferencia de una base de datos relacional no se hace un recorrido de todos los datos, es por eso que las operaciones sobre esta base de datos están alineados exactamente al despliegue de las entidades y sus relaciones haciéndolas muy efectivas y eficaces.

RDMS VS Graph Database

RDBMS	GRAPH DATABASE
Tablas	Grafos
Filas	Nodos
Columnas y datos	Propiedades y sus valores
Constraints	Relaciones
Joins	Transversal

Neo4j

Historia

Neo4j fue desarrollado por Neo Technology como una base de datos orientada a grafos de código abierto implementada en Java y Scala.

En el 2000, los fundadores de Neo encuentran problemas de rendimiento con RDBMS y comenzaron a construir el primer prototipo Neo4j.

2002 se desarrolló la primera versión de Neo4j.

2003 se inició el desarrollo, primera producción de 24 x 7, despliegue de Neo4j.

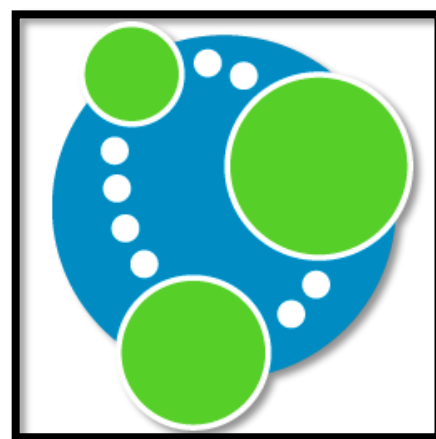
2007 se puso a disposición del público y se crea una empresa con sede en Suecia, además de código abierto el primer gráfico de base de datos Neo4j, bajo la GPL.

En el 2010 se da el Lanzamiento Neo4j versión 1.0.

El código de Neo4j está disponible en gitHub permitiendo a usuarios y entidades colaborar con ellos. Los desarrolladores describen a Neo4j como un motor de persistencia embebido, basado en disco que almacena datos estructurados en grafos más que en tablas.

Empresas como eBay, Walmart, Telenor, UBS, Cisco, Hewlett-Packard o Lufthansa han confiado en las cualidades de Neo4j para mejorar sus servicios.

Su página oficial es <http://neo4j.com/>



Neo4j Database	Fecha de lanzamiento
Neo4j 1.0	Febrero 2010
Neo4j 2.0	Diciembre 2013
Neo4j 2.1.3	Abril 2014

Instalación de Neo4j

Instalación en Windows

Lo primero que se debe hacer es ir a la página de Neo4j <http://neo4j.com/download/> y descargar el instalador tal y como se muestra en la siguiente imagen:

Luego dar clic en **Download Community Edition** y a continuación esperar a que este se descargue.

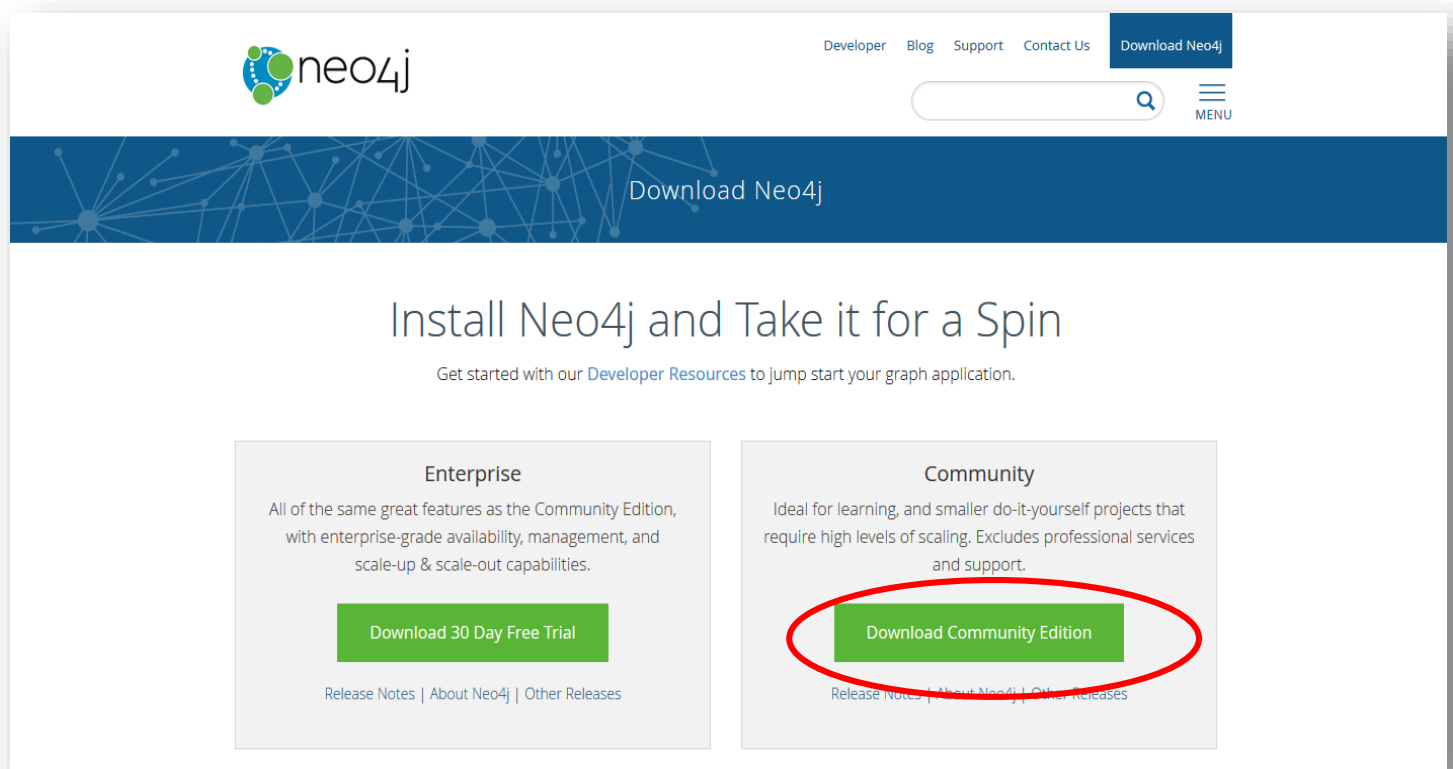


Figura 2. Página de descarga de Neo4j

Una vez se haya descargado el instalador de Neo4j, se deberá ejecutar y a continuación aparecerá la siguiente ventana:



Figura 3. Instalador Neo4j

Dar clic en **Next** para comenzar con la instalación.

A continuación aceptar los términos de la licencia

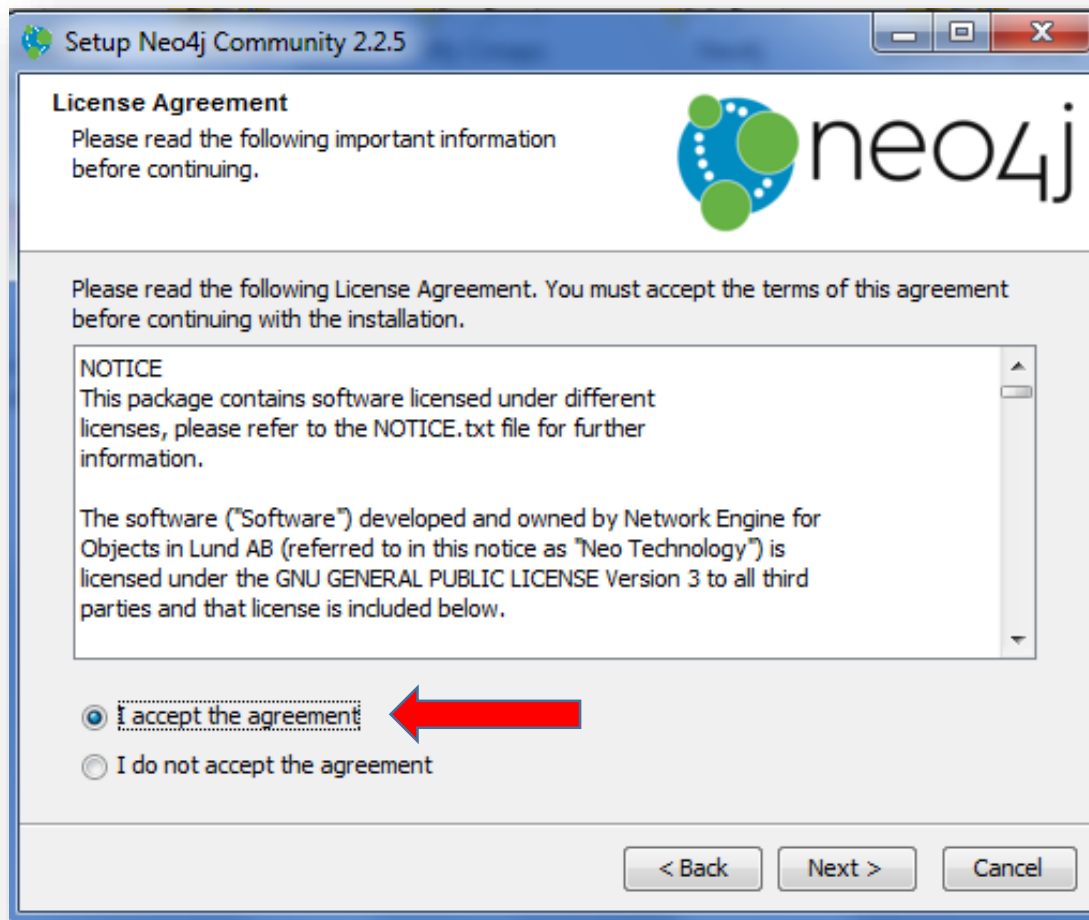


Figura 4. Términos de licencia de Neo4j

Elegir la ubicación en donde Neo4j será instalado en su computadora, una vez elegida la ubicación dar clic en **Next**.

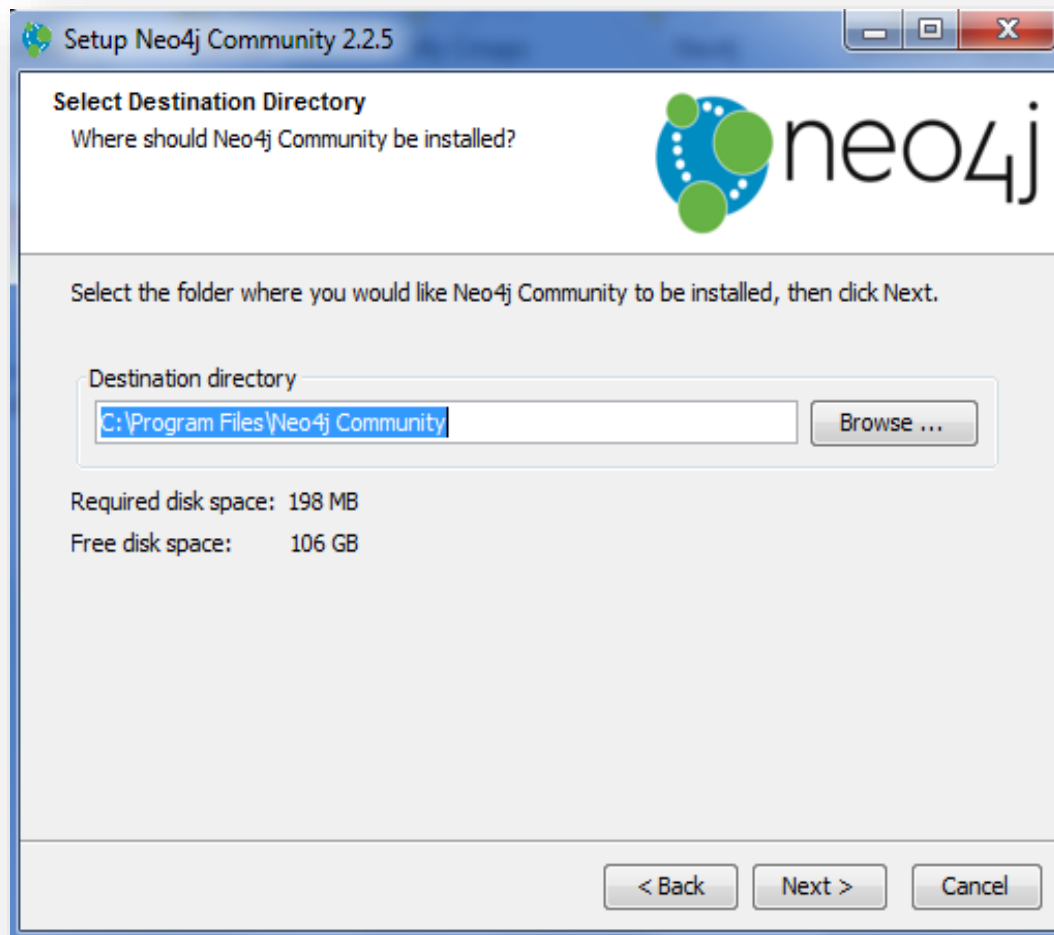


Figura 5. Ubicación para instalar Neo4j

Ahora se comenzará a instalar Neo4j en su computadora.

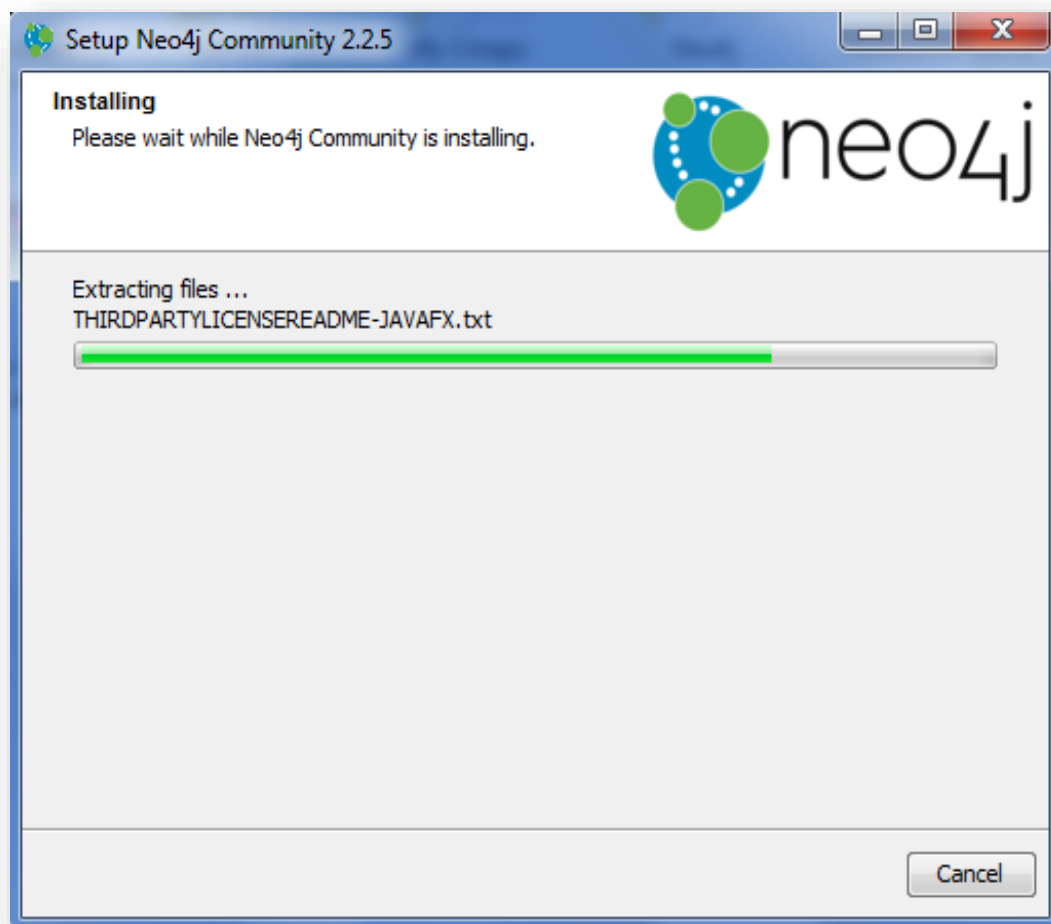


Figura 6. Instalación Neo4j

Finalmente se mostrará una ventana indicando que la instalación de Neo4j ha terminado correctamente.

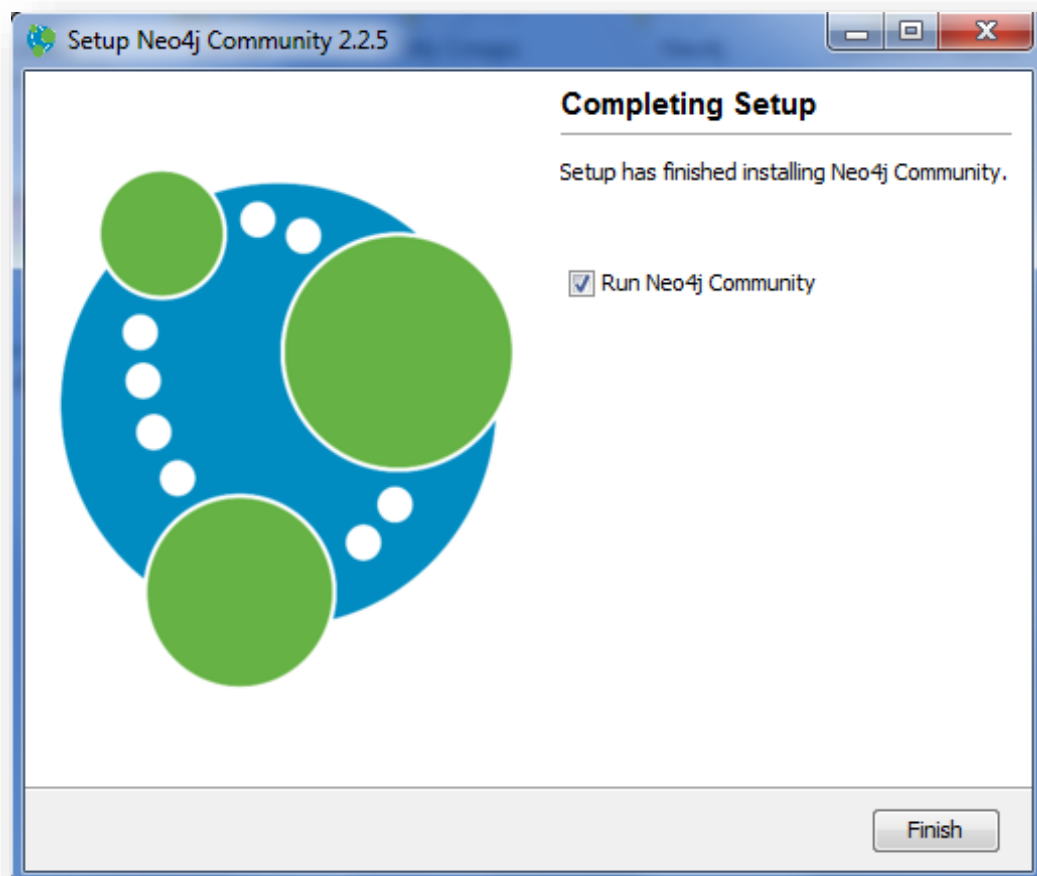


Figura 7. Instalación completada.

Con esto ya tenemos instalado Neo4j y se ha creado un servicio de Windows.

Este servicio de Windows es la parte back-end, la parte front-end se compone de un ejecutable inicial desde el que vamos a poder arrancar el servicio de neo4j y manejar la configuración de la base de datos y de un browser web desde donde vamos a poder realizar consultas, acceder a ayuda y visualizar los nodos, índices y relaciones que tenemos en la base de datos.

Para arrancar Neo4j podemos arrancarlo desde la ruta donde se ha instalado, dentro de la carpeta bin se encuentra **neo4j-community.exe**.

Al iniciar este ejecutable aparecerá una ventana donde se podrá ver la ruta de la base de datos la cual es **C:\Users\usuario\Documents\Neo4j\default.graphdb** como se muestra a continuación:

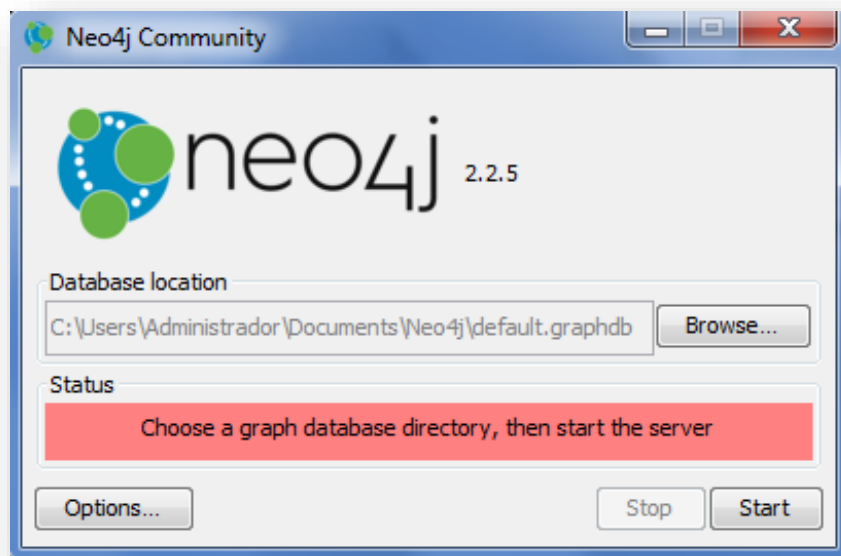


Figura 8. Servicio para iniciar Neo4j

En el botón **Start** es donde se inicia el servicio cuando pulsamos el botón, el servicio empezará a preparar Neo4j.

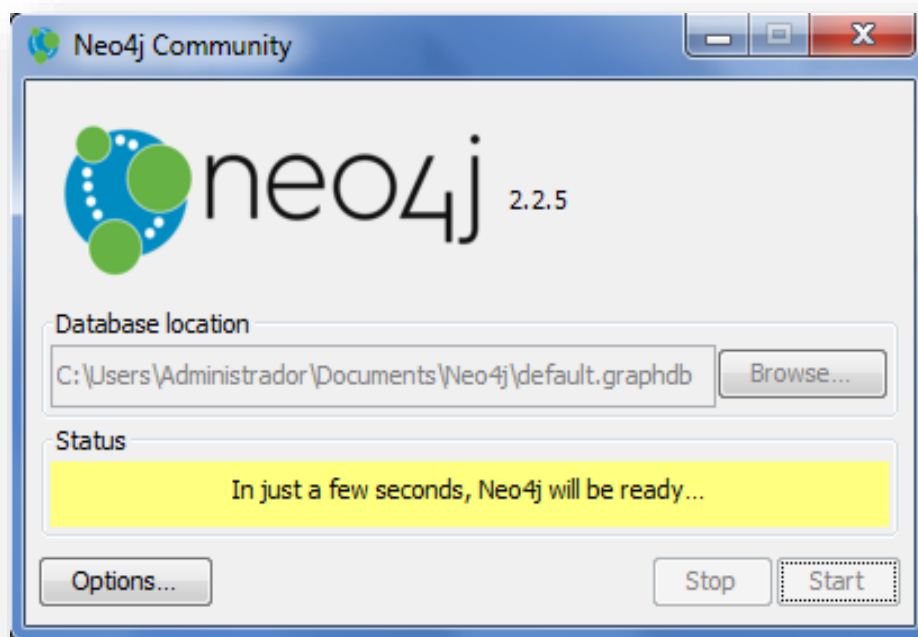


Figura 9. Servicio para iniciar Neo4j

Cuando Neo4j está listo aparecerá el link por defecto para acceder al browser de Neo4j
<http://localhost:7474/>.

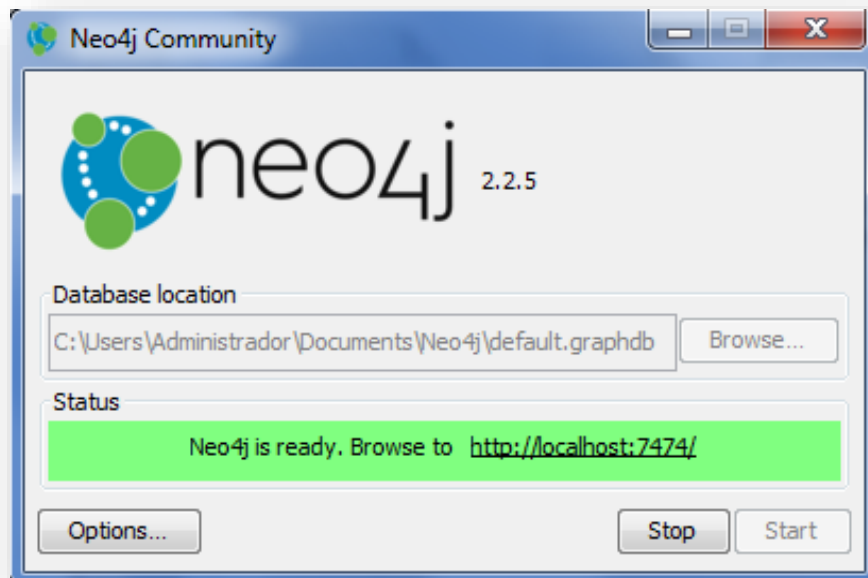


Figura 10. Link para acceder al browser de Neo4j

Ingresamos al link y entraremos al browser de Neo4j tal como se muestra en la siguiente imagen:

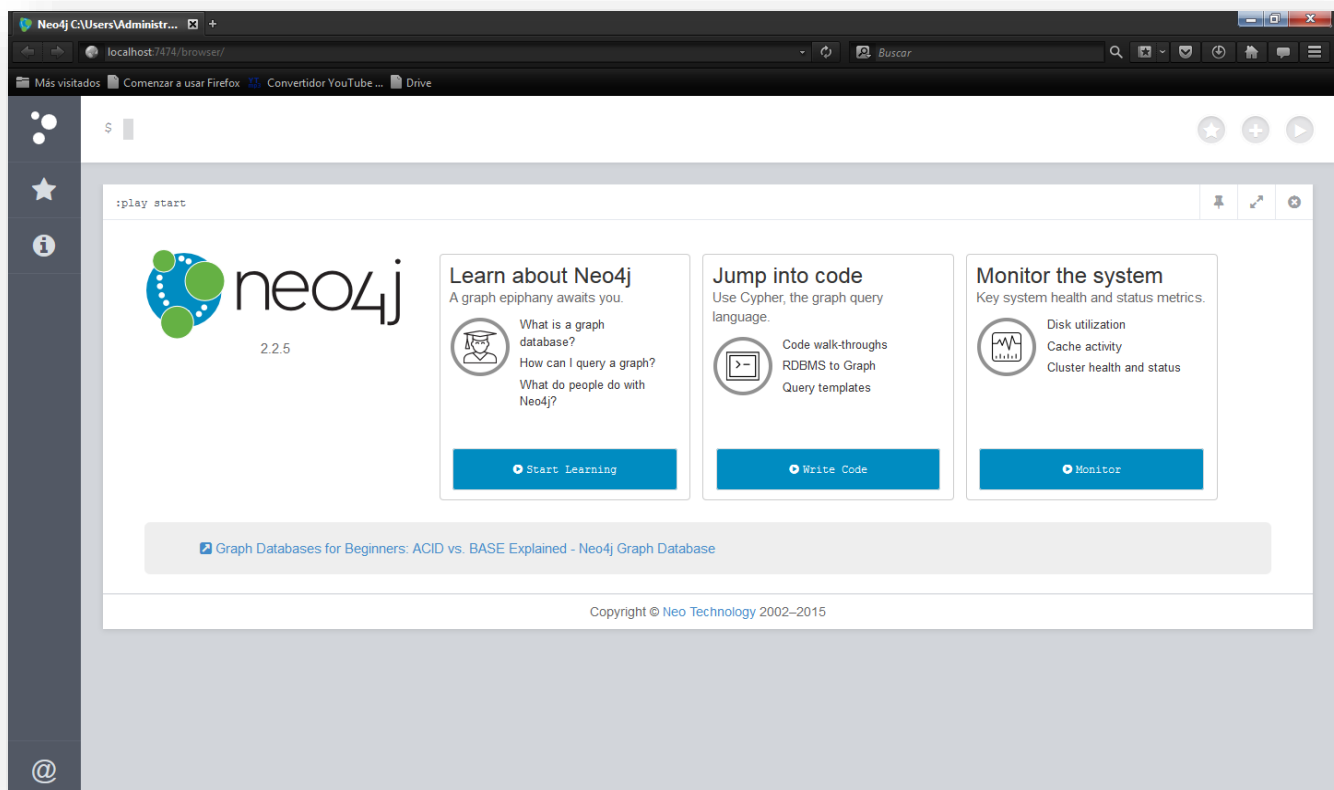


Figura 11. Neo4j Browser

Instalar Neo4j en GNU/Linux

Aclaración: Esta instalación fue hecha sobre Manjaro que a su vez está basada en ArchLinux.

Primero debemos dirigirnos a **<http://neo4j.com/>** a la pestaña **Download Neo4j** como veremos en la siguiente imagen:

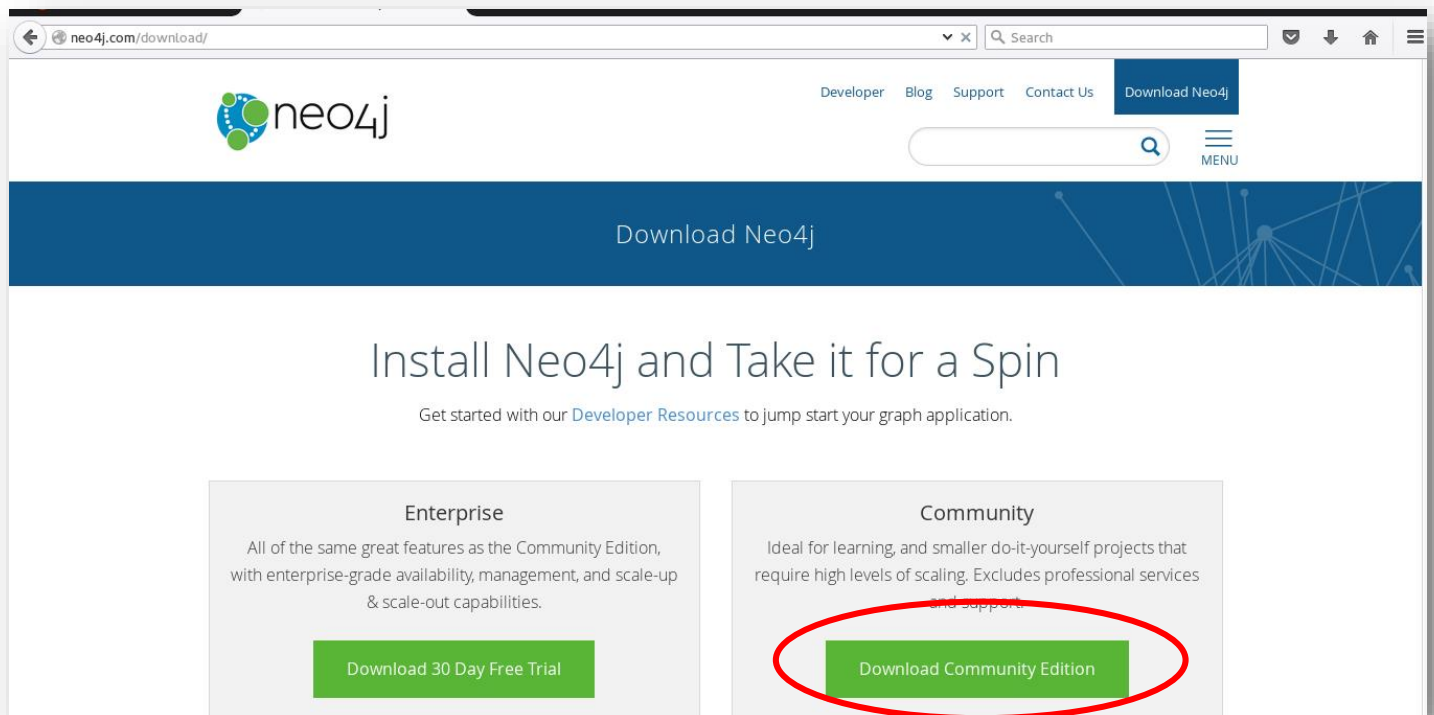


Figura 12. Página de descarga de Neo4j

Seguidamente daremos clic en **Download Community Edition** para descargar la versión gratuita.

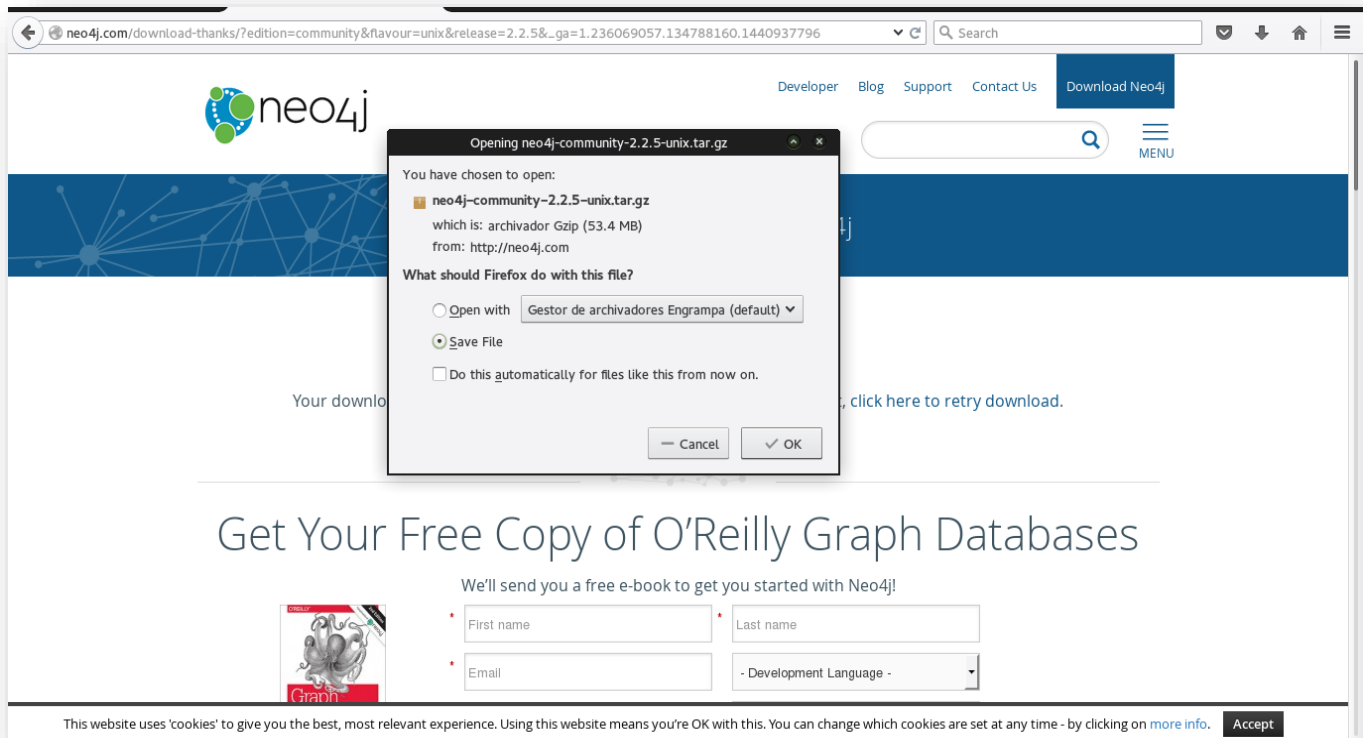


Figura 13. Descarga de Neo4j

Una vez descargado el archivo comprimido abrimos nuestra terminal y nos dirigimos a la carpeta donde este guardado el comprimido que descargamos, nuestro caso se descargó en la carpeta home como veremos en la imagen:

```
[nelson@nelss-9000 ~]$ ls -l
total 54708
drwxr-xr-x 2 nelson users 4096 ago 15 20:04 Biblioteca de calibre
drwxr-xr-x 4 nelson users 4096 ago 30 06:30 Calibre Library
drwxr-xr-x 4 nelson users 4096 ago 20 08:20 Descargas
drwxr-xr-x 8 nelson users 4096 ago 21 16:20 Documentos
drwxr-xr-x 2 nelson users 4096 ago 15 09:39 Escritorio
drwxr-xr-x 3 nelson users 4096 ago 15 09:07 IdeaProjects
drwxr-xr-x 2 nelson users 4096 ago 30 06:46 Imágenes
-rw-r--r-- 1 nelson users 0 ago 17 14:19 Lic.txt
drwxr-xr-x 2 nelson users 4096 ago 9 18:19 Música
-rw-r--r-- 1 nelson users 55962634 ago 30 06:38 neo4j-community-2.2.5-unix.tar.gz
drwxr-xr-x 2 nelson users 4096 ago 9 18:19 Plantillas
drwxr-xr-x 2 nelson users 4096 ago 9 18:19 Público
drwxr-xr-x 3 nelson users 4096 ago 15 09:25 PycharmProjects
-rw-r--r-- 1 nelson users 240 ago 17 14:40 user_space_report.txt
drwxr-xr-x 2 nelson users 4096 ago 9 18:19 Vídeos
```

Figura 14. Terminal

Seguidamente con el comando **tar -xf <nombre del comprimido>** descomprimiremos lo que descargamos:

```
[nelson@nelss-9000 ~]$ tar -xf neo4j-community-2.2.5-unix.tar.gz
[nelson@nelss-9000 ~]$
```

Figura 15. Comando para descomprimir

Una vez descomprimido entramos a la carpeta que traía el comprimido y en ella buscaremos y entraremos a la carpeta bin como se muestra en la siguiente imagen:

```
[nelson@nelss-9000 ~]$ cd neo4j-community-2.2.5
[nelson@nelss-9000 neo4j-community-2.2.5]$ ls -l
total 284
drwxr-xr-x 2 nelson users 4096 ago 30 06:52 bin
-rw-r--r-- 1 nelson users 81831 ago 27 06:42 CHANGES.txt
drwxr-xr-x 2 nelson users 4096 ago 30 06:52 conf
drwxr-xr-x 3 nelson users 4096 ago 30 06:52 data
drwxr-xr-x 2 nelson users 4096 ago 30 06:52 lib
-rw-r--r-- 1 nelson users 125051 ago 27 06:42 LICENSES.txt
-rw-r--r-- 1 nelson users 36045 ago 27 06:42 LICENSE.txt
-rw-r--r-- 1 nelson users 6005 ago 27 06:42 NOTICE.txt
drwxr-xr-x 2 nelson users 4096 ago 30 06:52 plugins
-rw-r--r-- 1 nelson users 1567 ago 27 06:42 README.txt
drwxr-xr-x 4 nelson users 4096 ago 30 06:52 system
-rw-r--r-- 1 nelson users 4318 ago 27 06:40 UPGRADE.txt
[nelson@nelss-9000 neo4j-community-2.2.5]$ cd bin/
[nelson@nelss-9000 bin]$ ls -l
total 44
-rwxr-xr-x 1 nelson users 10454 ago 27 06:42 neo4j
-rwxr-xr-x 1 nelson users 3575 ago 27 06:42 neo4j-import
-rwxr-xr-x 1 nelson users 4444 ago 27 06:42 neo4j-installer
-rwxr-xr-x 1 nelson users 3443 ago 27 06:42 neo4j-shell
-rw-r--r-- 1 nelson users 136 ago 27 06:42 README.txt
-rwxr-xr-x 1 nelson users 10483 ago 27 06:42 utils
```

Figura 16. Acceso a la carpeta bin

Finalmente para poder trabajar en neo4j debemos correr el siguiente comando:

“./neo4j console”, una vez hecho esto el servicio se empezara a ejecutar y ya podremos acceder al **localhost: 7474**

```
[nelson@nelss-9000 bin]$ ./neo4j console
WARNING: Max 1024 open files allowed, minimum of 40 000 recommended. See the Neo
4j manual.
Starting Neo4j Server console-mode...
Using additional JVM arguments: -server -XX:+DisableExplicitGC -Dorg.neo4j.serv
er.properties=conf/neo4j-server.properties -Djava.util.logging.config.file=conf/
logging.properties -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -XX:-Om
itStackTraceInFastThrow -XX:hashCode=5 -Dneo4j.ext.udc.source=tarball
2015-08-30 13:03:08.545+0000 INFO [API] Setting startup timeout to: 120000ms ba
sed on 120000
2015-08-30 13:03:10.531+0000 INFO [API] Successfully started database
2015-08-30 13:03:10.605+0000 INFO [API] Starting HTTP on port :7474 with 4 thre
ads available
2015-08-30 13:03:10.821+0000 INFO [API] Enabling HTTPS on port :7473
07:03:11.021 [main] INFO org.eclipse.jetty.util.log - Logging initialized @3173
ms
2015-08-30 13:03:11.087+0000 INFO [API] Mounting static content at [/webadmin]
from [webadmin.html]
2015-08-30 13:03:11.156+0000 INFO [API] Mounting static content at [/browser] f
rom [browser]
07:03:11.162 [main] INFO org.eclipse.jetty.server.Server - jetty-9.2.4.v2014110
3
07:03:11.208 [main] INFO o.e.j.server.handler.ContextHandler - Started o.e.j.s.
h.MovedContextHandler@77134f4c{/,null,AVAILABLE}
07:03:11.292 [main] INFO o.e.j.w.StandardDescriptorProcessor - NO JSP Support f
or /webadmin, did not find org.apache.jasper.servlet.JspServlet
07:03:11.317 [main] INFO o.e.j.server.handler.ContextHandler - Started o.e.j.w.
WebAppContext@77f758ca{/webadmin,jar:file:/home/nelson/neo4j-community-2.2.5/syst
em/lib/neo4j-server-2.2.5-static-web.jar!/webadmin.html,AVAILABLE}
07:03:11.807 [main] INFO o.e.j.server.handler.ContextHandler - Started o.e.j.s.
ServletContextHandler@735c81c3{/db/manage,null,AVAILABLE}
07:03:12.060 [main] INFO o.e.j.server.handler.ContextHandler - Started o.e.j.s.
ServletContextHandler@5e01fdc0{/db/data,null,AVAILABLE}
07:03:12.076 [main] INFO o.e.j.w.StandardDescriptorProcessor - NO JSP Support f
or /browser, did not find org.apache.jasper.servlet.JspServlet
07:03:12.077 [main] INFO o.e.j.server.handler.ContextHandler - Started o.e.j.w.
WebAppContext@3164c650{/browser,jar:file:/home/nelson/neo4j-community-2.2.5/syst
```

Figura 17. Ejecución de comando. `/neo4j console` para iniciar el servicio y acceder al `localhost`.

Finalmente podremos ingresar al Neo4j Browser tal y como se muestra a continuación:

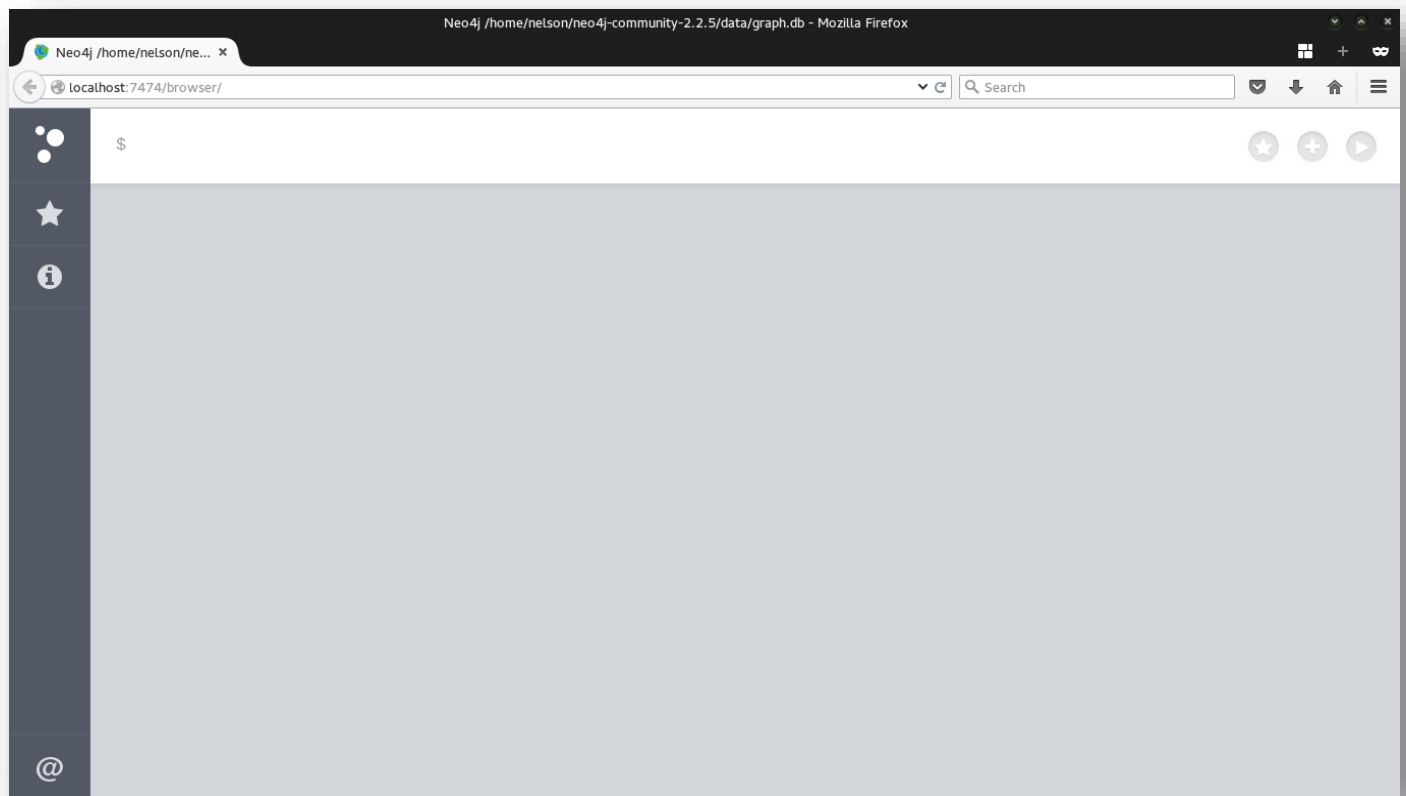


Figura 18. Neo4j Browser

Propósito

Las bases de datos orientadas a grafos ayudan a encontrar relaciones entre los datos y extraer su verdadero valor.

Arquitectura

- Neo4j usa grafos para representar datos y las relaciones entre ellos.
- Una colección de nodos (cosas) y aristas (relaciones) que conectan pares de nodos.
- Se asocian propiedades (clave-valor) sobre nodos y relaciones.
- Las relaciones conectan dos nodos y tanto nodos como relaciones pueden alojar un número arbitrario de pares clave/valor.
- Neo4j utiliza grafos de propiedad para extraer valor añadido de los datos
- No hay esquema, diseño de modelo de datos de abajo a arriba
- No SQL
- Neo4j tiene CQL (Cypher Query Language) Como lenguaje de consulta.

Modelo de datos

Un Grafo → guarda datos en → Nodos → que tienen → Propiedades

-El grafo más sencillo es un nodo con una colección de pares clave-valor o propiedades

-Una propiedad consta de una clave de tipo string que apunta a un valor primitivo o en forma de colección

Los Nodos → se organizan en → Relaciones → que tienen → Propiedades

-Las relaciones hacen que los Nodos se conviertan en estructuras arbitrarias.

Un Transversal → navega → un Grafo, el cual identifica → Caminos → que ordenan → Nodos

-Permite encontrar respuestas a preguntas como “qué música tienen mis amigos que yo no tengo”

Un Índice → mapea de → Propiedades → a → Nodos o Relaciones

-Permite encontrar un nodo o relación en función de una propiedad.

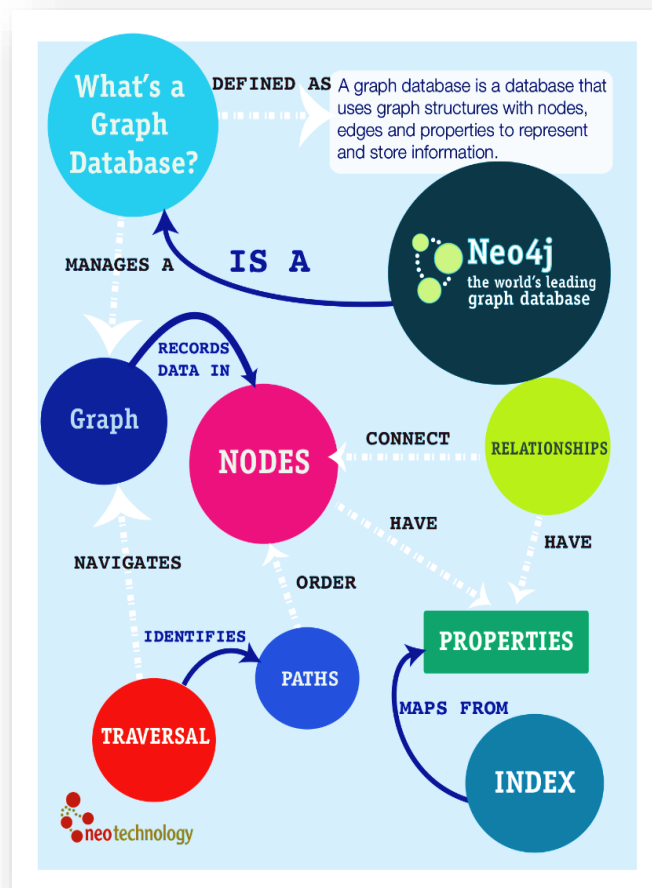


Figura 19. Modelo de datos de Neo4j

La estructura de directorios que vamos a encontrarnos, una vez hayamos descomprimido el paquete, es la siguiente:

Directorio	Significado
/bin	Contiene diferentes ficheros bat que permiten arrancar/parar la base de datos y operar con ella. El que nos permite poner en marcha la base de datos de manera inmediata es "Neo4j.bat" .
/conf	Ficheros básicos de configuración.
/data	Aquí están todos los datos de la base de datos que usan nuestras aplicaciones.
/doc	Directorio con documentación y tutoriales en diferentes formatos.
/lib	Todas las librerías que hacen que Neo4J funcione, ya sea como servidor o en modo embebido, es lo que usaremos en nuestras aplicaciones.
/plugins	Cualquier extensión que añade funcionalidad al servidor, ya sea pensada para obtener nodos, relaciones, caminos o propiedades de los grafos.
/system	Es el core de Neo4J o motor de ejecución, aquí está todo lo que Neo4J usa cuando está en funcionamiento.

Todos los parámetros de inicialización de Neo4J, están en el fichero **"neo4j.properties"** y se aplican al volver a arrancar la base de datos, por tanto cualquier actualización de la misma implica parar la instancia (o instancias si está funcionando en un clúster) que queramos actualizar.

Escenarios de uso

Neo4j puede usarse en dos escenarios típicos:

Puede estar embebida en una aplicación:

Podemos incluir las librerías necesarias de Neo4J, dentro de nuestra aplicación, de manera que seamos capaces de hacer uso de las ventajas de esta base de datos sin necesidad de tener que depender de un servidor externo, y equivale a trabajar con la base de datos en memoria.

Embeber la base de datos en la aplicación puede hacerse también de dos maneras diferentes y que depende de nuestras necesidades de rendimiento y de los recursos de los que dispongamos:

1. Si estamos en un entorno donde no tenemos muchos recursos (memoria, CPU, etc.) o las necesidades de la aplicación no son muy exigentes, podemos usar una instancia simple de Neo4J. A esta instancia simple se accede mediante llamadas al **API GraphDatabaseService**.
2. En caso de necesitar un entorno de alta disponibilidad con múltiples instancias, y en donde los recursos de los que disponemos son menos escasos, podemos acceder al API de **HighlyAvailableGraphDatabase**.

Funcionar como un servidor:

Es la forma más habitual de acceder a la base de datos, y en este caso funcionaría como un servidor típico de bases de datos, corriendo sobre una o varias máquinas (en función de nuestras necesidades de rendimiento), que atiende las peticiones que le lleguen por parte de los clientes.

Aquí trabajaremos con peticiones de tipo **REST (Representational State Transfer)** al servidor para que nos resuelva nuestra consulta. Las peticiones siguen el esquema estándar de REST.

Un servicio **REST** no tiene estado, lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos. Esto es, que no se puede llamar a un servicio REST y pasarle unos datos (por ejemplo: un usuario y una contraseña) y esperar que nos recuerde en la siguiente petición. De ahí el nombre: el estado lo mantiene el cliente y por lo tanto es el cliente quien debe pasar el estado en cada llamada.

Si quiero que un servicio REST me recuerde, debo pasarle quien soy en cada llamada. Eso puede ser un usuario y una contraseña, un token o cualquier otro tipo de credenciales, pero debo pasarlas en cada llamada. Y lo mismo aplica para el resto de información.

El modo de operar que elijamos de Neo4J, depende por tanto, del problema que estemos resolviendo, y del entorno en donde la aplicación se ejecutará y esto implica que tenemos que tener en cuenta dos cosas:

- La primera es que las restricciones de memoria limitan el tamaño del grafo, por tanto cuanto más memoria, mayor será el grafo que podamos manipular sobre la misma.
- La segunda es que las restricciones de entrada/salida a disco limitan el rendimiento de acceso a la información, si el soporte donde almacenamos los datos tiene un mal rendimiento, parece claro que las consultas que hagamos al grafo tardaran más en realizarse.

Requerimientos mínimos de Neo4j:

	CPU	Memoria	Disco	Sistemas de Ficheros
Mínimo	Intel 486	1 Gb	SCSI,EIDE	ext3 o similar
Recomendado	Intel Core i7	4-8 Gb	SSD w/SATA	ext4, ZFS

Y funciona sobre los sistemas operativos Windows, Linux y Mac OS X.

Migración entre versiones

Es posible migrar desde una versión inferior a una superior, por ejemplo de la versión 1.1 a la 1.2, no es posible migrar de una 1.1 a una 1.3, es decir, no está permitido la migración entre versiones que impliquen salto de versiones intermedias.

Capacidad

A partir de la versión 2.1.3 Neo4j, es compatible con un montón de nodos, relaciones y propiedades para desarrollar y apoyar las aplicaciones de las empresas.

Bloques de Neo4j	Capacidad
Nodos	Alrededor de 35 billones
Relaciones	Alrededor de 35 billones
Etiquetas	Alrededor de 275 billones

Características

- Licencia abierta: Dual y comercial
- Código abierto
- Base de datos orientada a grafos
- Las transacciones son atómicas, consistentes y duraderas.
- Apropiaada para casos de uso de la web como etiquetado, anotaciones de metadatos, redes sociales, wikis, y otros conjuntos de datos jerárquicos o en forma de red.
- Contiene una interfaz de usuario para ejecutar comandos CQL: Neo4j Data Browser.
- Neo4j implementa un modelo orientado a grafos de manera eficiente a nivel de almacenamiento.
- Ofrece una gran cantidad de características de las bases de datos incluyendo el cumplimiento de transacción ACID (Atomicidad, coherencia, aislamiento y durabilidad).
- Es compatible con la exportación de datos de consulta de JSON y formato XLS.
- Proporciona API REST para ser visitada por cualquier lenguaje de programación como Java, Spring, Scala.
- Tiene una librería Java que te permite empotrarla en tu aplicación Java.
- Es compatible con dos tipos de API de Java: Cypher API y Native Java API para desarrollar aplicaciones Java.
- Neo4j puede almacenar cualquier tipo de información usando los siguientes conceptos:
 - Nodos: Almacena los registros del grafo.
 - Relaciones: Conecta los distintos nodos.
 - Propiedades: Datos con nombre.

Ventajas

- Es muy fácil para representar los datos conectados.
- Es muy fácil y rápido para recuperación, recorrido y navegación de los datos más conectados.
- Representa los datos semi-estructurados con mucha facilidad.
- Los comandos CQL del lenguaje de consulta están en formato legible y es fácil de aprender.
- Utiliza modelo de datos simple y de gran alcance.
- Agilidad en la gestión de datos.
- Las bases de datos orientadas a grafos como Neo4j tienen mejor rendimiento que las relacionales (SQL) y las no relacionales (No SQL). La clave es que, aunque las consultas de datos aumenten exponencialmente, el rendimiento de Neo4j no desciende, frente a lo que sí sucede con las BD relacionales como MySQL.
- Flexibilidad y escalabilidad, las bases de datos orientadas a grafos aportan mucho en este sentido porque cuando aumentan las necesidades, las posibilidades de añadir más nodos y relaciones a un grafo ya existente son enormes.

Desventajas

- Neo4j Community no es escalable para cantidades de datos muy grandes, hay que cambiar la configuración de la memoria en Neo4j acorde con las necesidades del grafo.
- Lo que más ocupa en memoria es el almacenamiento de los nodos, se hacen ineficientes las consultas si no se ha modelado el problema bien y si se almacena demasiada información en los nodos.
- Neo4j no tiene gestión de usuarios, nada de autenticación: hay una extensión para autenticar, o como alternativa, se puede poner un proxy delante de la BD para autenticar.

Aplicaciones que usan bases de datos orientadas a grafos

Podemos decir que las bases de datos de orientadas a grafos son útiles principalmente para almacenar datos más conectados.

Si utilizamos RDBMS para almacenar los datos más conectados, entonces no proporcionan un rendimiento adecuado para recorrer gran cantidad de datos. En estos escenarios, las bases de datos orientadas a grafos mejoran el rendimiento de las aplicaciones de una forma eficiente. Hoy en día, la mayoría de las aplicaciones de redes sociales como **Facebook, Google+, LinkedIn, Twitter, Yammer** entre otras y aplicaciones de alojamiento de video como **Google, YouTube, Flickr, Yahoo Video**, están utilizando los datos más conectados.



Usos de Neo4j

Neo4j tiene distintos usos en la actualidad en empresas internacionales:

- Detección del fraude:

Neo4j ya trabaja con varias corporaciones en la detección del fraude en sectores como la banca, los seguros o el comercio electrónico. Esta base de datos puede descubrir patrones que con otro tipo de BD sería difícil de detectar. Un fraude habitual es la apertura de líneas de crédito con identidades falsas con la idea no pagar.

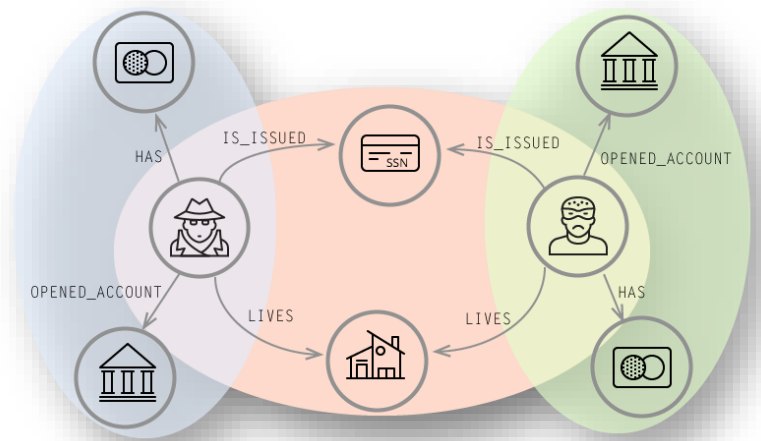


Figura 20. Detección del fraude

- Recomendaciones en tiempo real y redes sociales:

Neo4j permite conectar de forma eficaz a las personas con nuestros productos y servicios, en función de la información personal, sus perfiles en redes sociales y su actividad online reciente. En este sentido, las bases de datos orientadas a grafos son interesantes porque son capaces de conectar personas e intereses. Con esa información, una empresa puede ajustar sus productos y servicios a su público objetivo y personalizar la recomendación en función de los perfiles. Eso es lo que permite que se aumente la precisión comercial y el compromiso del cliente.

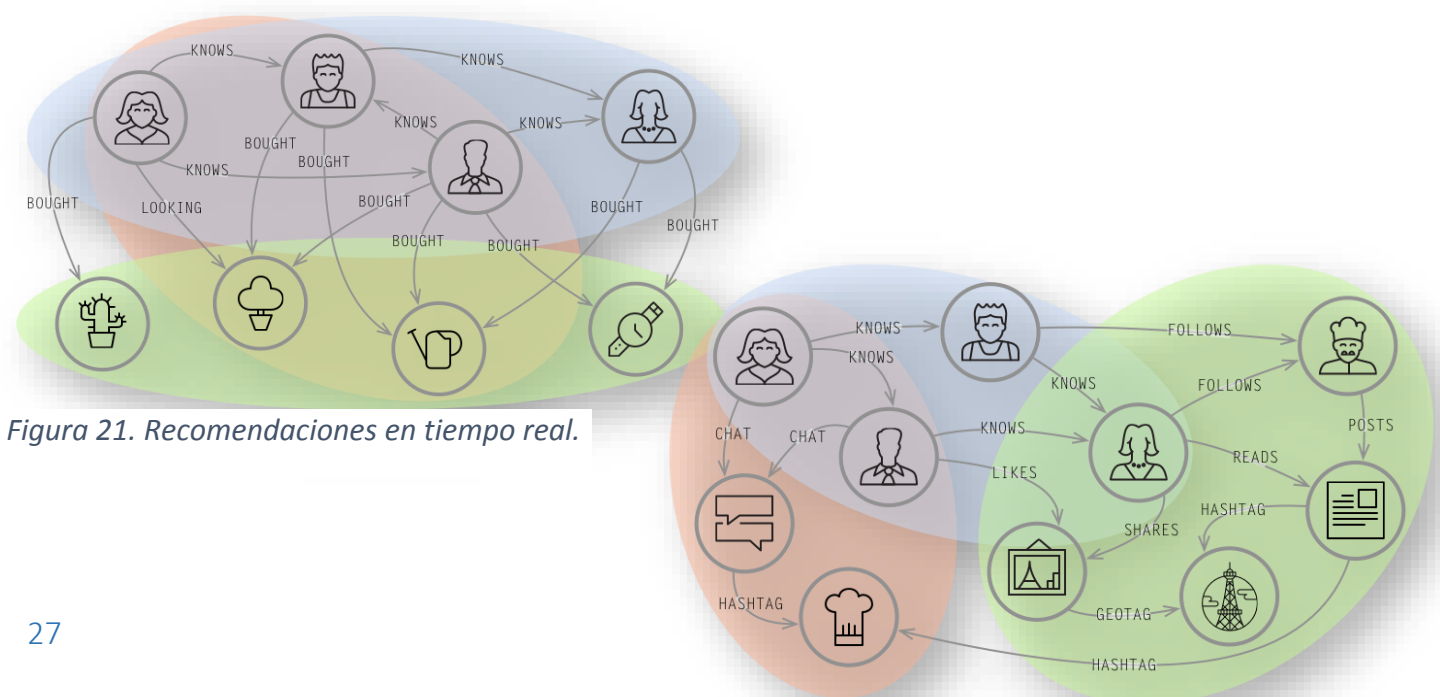


Figura 21. Recomendaciones en tiempo real.

Figura 22. Redes Sociales

- Gestión de centros de datos:

Las bases de datos gráficas son el antídoto perfecto ante el crecimiento desbordante de los datos. La gran cantidad de información, dispositivos y usuarios hacen que las tecnologías tradicionales no puedan gestionar tantos datos. La flexibilidad, rendimiento y escalabilidad de Neo4j permite gestionar, monitorizar y optimizar todo tipo de redes físicas y virtuales pese a la gran cantidad de datos.

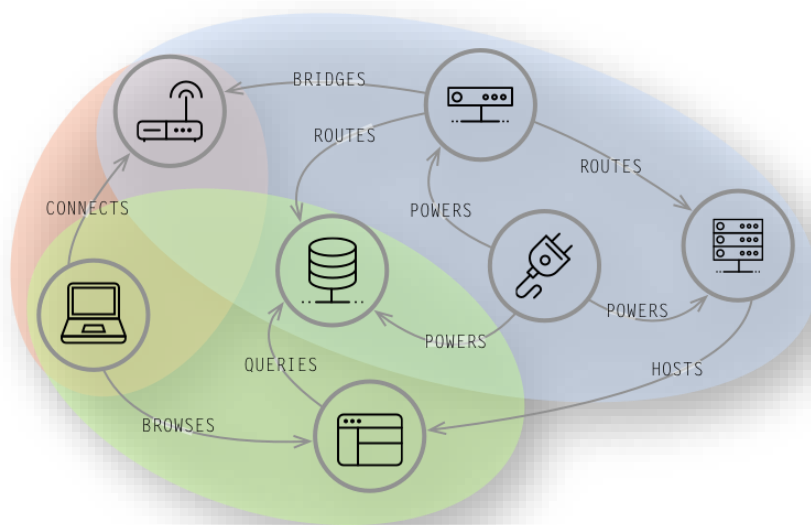


Figura 23. Gestión de centros de datos.

- Gestión de sistemas de datos maestros:

La gestión de datos maestros (Master Data Management) .La creación de un sistema de información centralizado y fiable siempre es una cuestión compleja. El objetivo final es que cada miembro de una organización use los mismos formatos y aplicaciones para los datos. Eso genera un protocolo de trabajo que es aprovechable por el resto.

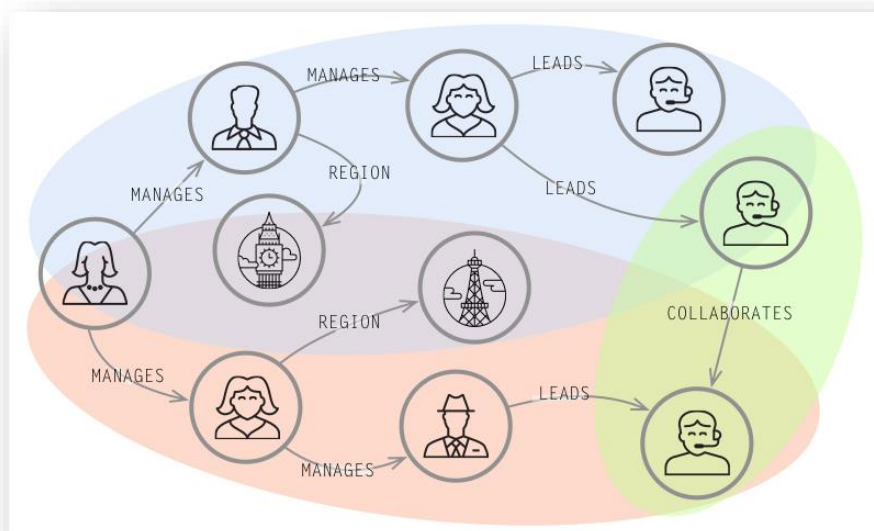


Figura 24. Gestión de sistemas de datos maestros.

- Herramientas de búsqueda:

Con herramientas de búsqueda basadas en grafos Neo4j, sus consultas devuelven resultados más precisos y pertinentes en tiempo real.

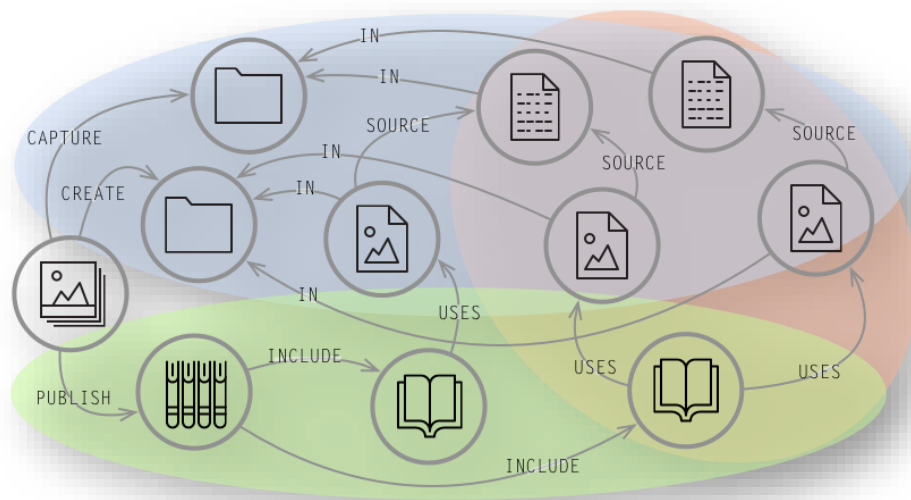


Figura 25. Herramientas de búsqueda.

Lenguaje Cypher

Cypher es el nombre del lenguaje que usa neo4j para crear y manejar la base de datos, es un lenguaje bastante natural y fácil de utilizar como veremos a continuación.

Para poder crear un grafo se utiliza la siguiente sintaxis:

```
Create  
( ) --> ( )
```

Los '**()**' representan los nodos, en estos es donde se almacenan los datos y las '**-->**' representan las relaciones.

Dentro de los nodos puede ir la siguiente sintaxis:

```
(Terror:Clase {nombre:'Terror'})
```

En donde **Terror** antes de los ':' es el identificador Clase es el tipo de nodo (equivalente al nombre de una Tabla en SQL) y para agregar atributos, se debe hacer entre '{ }', como en este caso el atributo nombre que para este caso es **Terror**, en conclusión para crear un nodo se debe hacer de la siguiente manera:

```
(<identificador>:<Tipo de nodo> {<Atributo>:<dato>,...})
```

Ahora para poder crear las relaciones entre los nodos se utiliza la siguiente sintaxis:

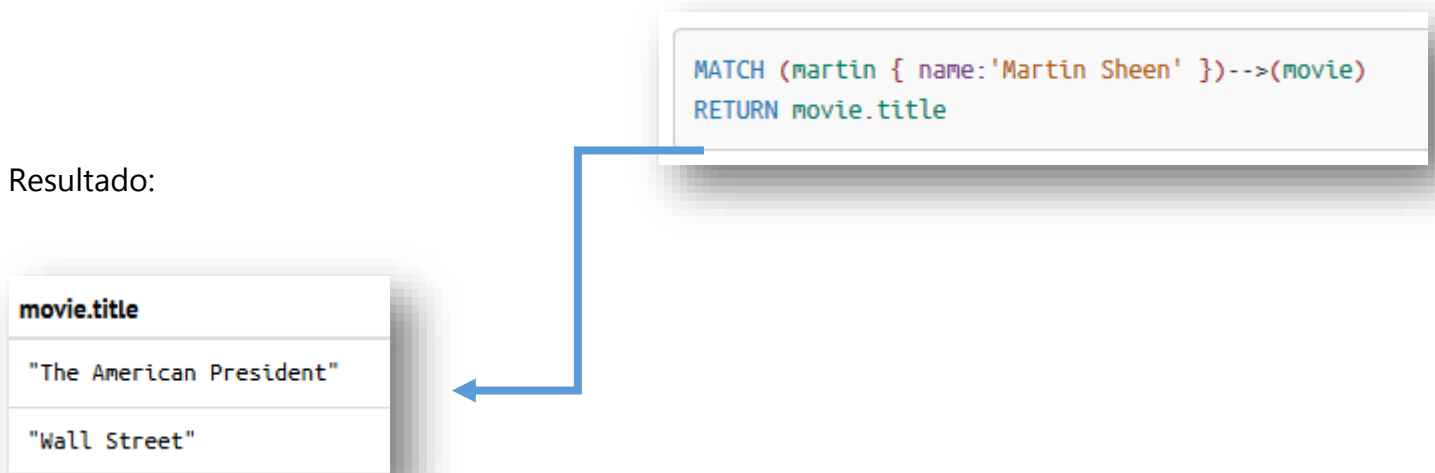
```
CREATE  
(StephenK)-[:Escribio]->(IT)
```

```
CREATE  
(<identificador>)-[:<relacion> {<Atributos>:<datos>,...}]->(identificador)
```

Como se ve en la imagen se escribe entre paréntesis el nombre de los identificadores de los nodos ya creados, y entre ellos se escribe una flecha con dos guiones, entre los guiones abrimos '[']' en donde escribiremos ':' y el nombre de la relación, igual que a un nodo, se puede agregar atributos a las relaciones.

Cabe resaltar que en Cypher existen los strings, ints y bools, operaciones de suma, resta, multiplicación entre otros, comparaciones mayor, menor, igual, operadores booleanos, existen colecciones, funciones como el **avg()** y **count()** como en cualquier otro SQL.

Para poder hacer búsquedas se utiliza la palabra 'Match' como veremos en la siguiente sintaxis:



El query anterior lo que hace es buscar las películas en las que haya actuado Martinsheed

```
MATCH (person)
WHERE person.name = 'Peter' AND person.belt IS NULL RETURN person
```

Al igual que en los lenguajes SQL normal, también existe la sentencia `where` para especificar nuestra búsqueda. También existe la sentencia `order by` para ordenar y `limit` para limitar el número de búsqueda.

Y finalmente para poder actualizar datos se utiliza la sentencia '**set**' como en el siguiente caso:

```
MATCH (n { name: 'Andres' })  
SET n.surname = 'Taylor'  
RETURN n
```


Prototipo

Neo4j y Java

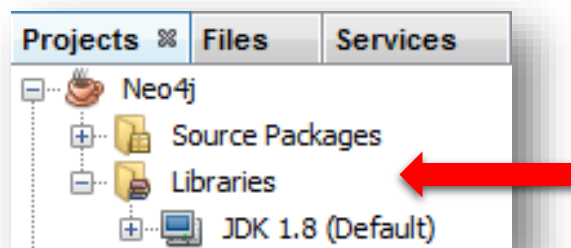
Podemos usar java para trabajar con Neo4j, crear relaciones y nodos y luego poder mostrar en forma de grafo cada una de las estructuras creadas, para la realización de este prototipo se decidió utilizar java, es decir utilizar neo4j embebida en una aplicación.

Requerimientos para empezar a trabajar Neo4j con java

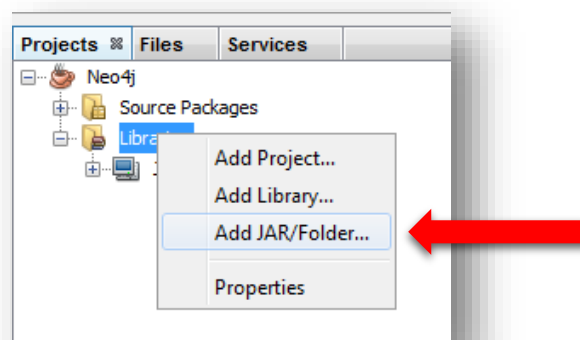
Agregar librería de neo4j a la aplicación de java:

neo4j-desktop-2.2.5.jar tal y como se muestra a continuación:

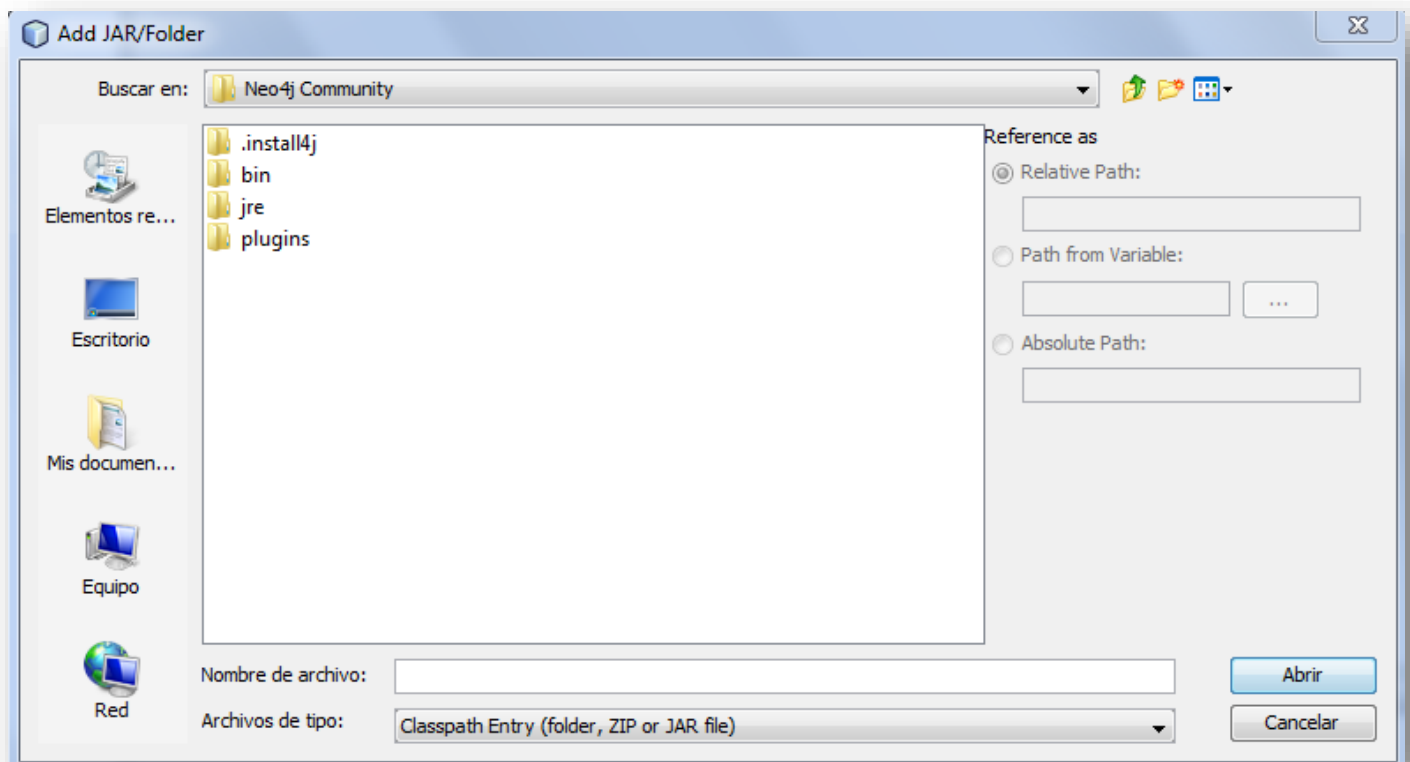
Ir a la carpeta **Libraries** del proyecto:



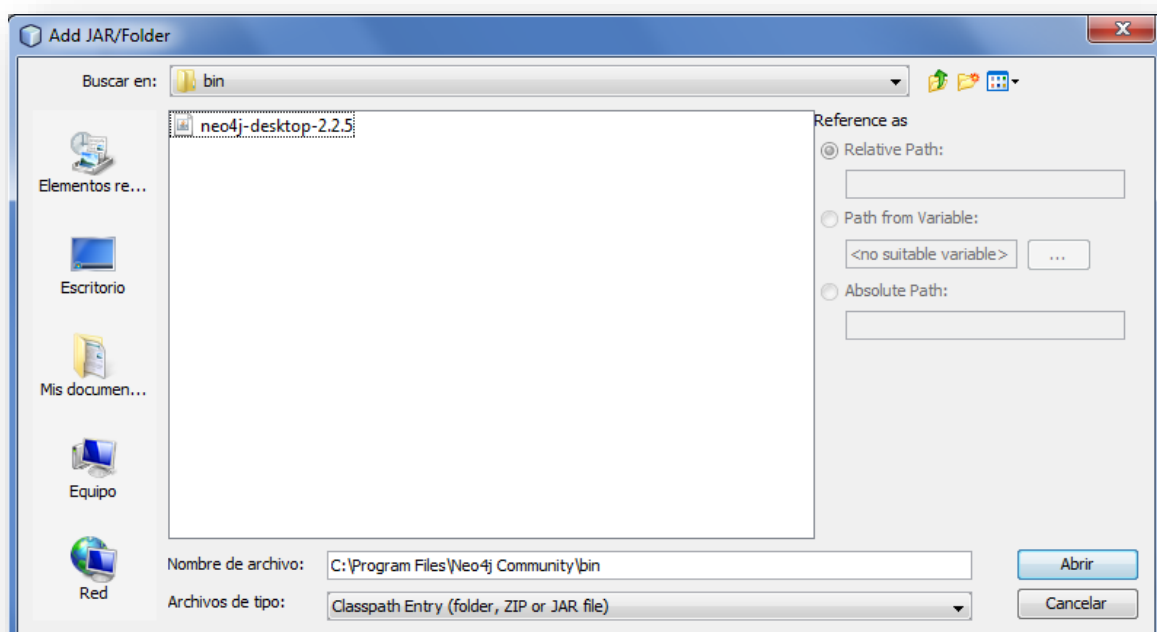
A continuación dar clic derecho en la carpeta **Libraries** y dar clic en **Add JAR/Folder**



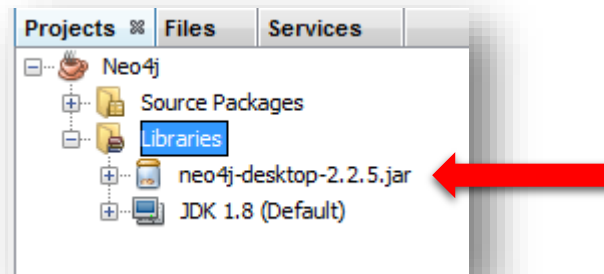
Se mostrará la siguiente ventana:



A continuación ir a la carpeta **Neo4j Community** e ingresar a la carpeta **bin**



Dar clic en la librería y en **Abrir** y ahora ya tendremos agregada la librería al proyecto.



¿Cómo usar Java con Neo4j?

Lo primero es tener una base de datos, a la cual le debemos indicar el directorio donde residen sus archivos (si el directorio no existe, lo crea y es una nueva base, si el directorio ya existe, lee los datos encontrados en ese directorio).

La cual se encuentra en la carpeta de Neo4j: **C:\Users\Administrador\Documents\Neo4j**.

El principal punto de acceso para correr una instancia Neo4j es **GraphDatabaseService**. La aplicación más común es la clase **EmbeddedGraphDatabase**, que se usa para incrustar Neo4j en una aplicación. Normalmente, se crea una **EmbeddedGraphDatabase** como se muestra a continuación:

```
public static void main(String[] args) {  
    GraphDatabaseFactory neodb = new GraphDatabaseFactory();  
    //Directorio de la Base de datos Neo4j  
    GraphDatabaseService graphDb = neodb.newEmbeddedDatabase("C:\\Users\\Administrador\\Documents\\Neo4j\\default.graphdb");  
}
```

Directorio en donde se encuentra la base de datos Neo4j.

GraphDatabaseService proporciona operaciones para **create node**, **get nodes given an id** y en última instancia **shutdown** Neo4j.

Debemos definir los tipos de relación que queremos almacenar. En este caso vamos a crear unos enums de Java para los tipos de relación que queremos manejar. Con eso tenemos la estructura básica. Ahora solamente tenemos que ir creando nodos, y relacionándolos entre ellos. Todo lo que se hace en Neo4j debe ser dentro de una transacción.

```
public class Neo4j_Prueba {  
  
    //Creacion de los tipos de los nodos  
    public enum NodeType implements Label{  
        Person, Course  
    }  
  
    //Creacion de las relaciones entre los nodos  
    public enum RelationType implements RelationshipType{  
        Knows, BelongsTo  
    }  
}
```

The diagram illustrates the execution flow of a Neo4j transaction. It features a code block on the left with several annotations on the right, connected by arrows indicating the sequence of operations.

Code Block:

```
try(Transaction tx = graphDb.beginTx()){  
    //Creacion de nodos con sus propiedades  
    Node bobNode = graphDb.createNode(NodeType.Person); // Tipo del nodo Person  
    bobNode.setProperty("Pid", 5001);  
    bobNode.setProperty("Name", "Bob");  
    bobNode.setProperty("Age", 23);  
  
    Node aliceNode = graphDb.createNode(NodeType.Person);  
    aliceNode.setProperty("Pid", 5002);  
    aliceNode.setProperty("Name", "Eve");  
  
    Node eveNode = graphDb.createNode(NodeType.Person);  
    eveNode.setProperty("Name", "Eve");  
  
    Node itNode = graphDb.createNode(NodeType.Course); //Tipo del nodo Course  
    itNode.setProperty("Id", 1);  
    itNode.setProperty("Name", "IT Beginners");  
    itNode.setProperty("Location", "Room 153");  
  
    Node electronicNode = graphDb.createNode(NodeType.Course);  
    electronicNode.setProperty("Name", "Electronics Advanced");  
  
    //Creacion de relaciones entre los nodos  
    bobNode.createRelationshipTo(aliceNode, RelationType.Knows); //Tipo de relacion Knows  
    tx.success();  
    graphDb.shutdown(); //Cerrar base  
}
```

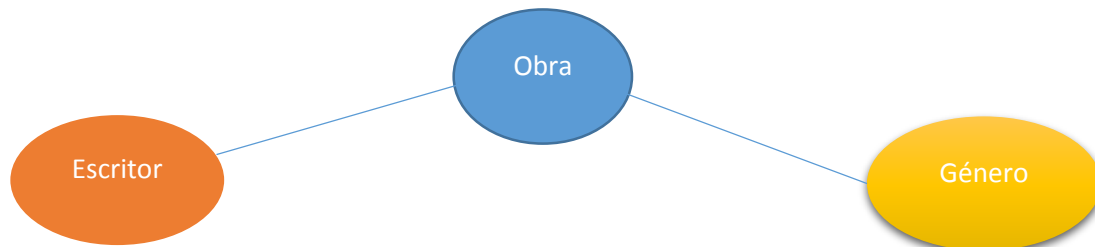
Annotations:

- Inicia una nueva transacción**: Points to the `graphDb.beginTx()` line.
- Creación de un nodo, con `createNode()`, donde se le asigna el tipo en este caso `Person`**: Points to the `graphDb.createNode(NodeType.Person)` line.
- Al nodo creado se le asigna atributos con `setProperty()`**: Points to the `bobNode.setProperty()` lines.
- Creación de Relaciones con `createRelationshipTo()`, en donde se le indica el nodo con el cual se quiere relacionar y el tipo de relación entre los nodos.**: Points to the `bobNode.createRelationshipTo()` line.
- Para hacer commit**: Points to the `tx.success();` line.
- Cerrar la base**: Points to the `graphDb.shutdown();` line.

Una transacción cuando se crea, está lista para dar rollback a todo lo que ocurra dentro de ella, al final de las operaciones, debemos llamar su método `success()` para que se haga commit y no rollback. Al final de la transacción debemos cerrar la base llamando al método **`shutdown()`**

Explicación del prototipo

Creamos un prototipo de consulta de Obras Literarias. Internamente el Grafo se comporta como en el esquema siguiente:



Como se presenta, se tiene una Obra, la cual se relaciona directamente con un Escritor y con un Género.

Utilizando los conceptos básicos de Bases de Datos de Grafos, nuestro prototipo incluye en la base de datos una obra y realiza la relación con el autor y el género especificado.

Tanto el Escritor como el Género, deben de estar incluidos previamente en la base de datos.

El prototipo permite consultar los datos de la base con todas las combinaciones posibles.

Referencias Bibliográficas

- Cía, J. F. (24 de 05 de 2015). *BBVA*. Obtenido de Neo4j: qué es y para qué sirve una base de datos orientada a grafos: <http://bbvaopen4u.com/es/actualidad/neo4j-que-es-y-para-que-sirve-una-base-de-datos-orientada-grafos>
- Community, N. (2014). *Uses of Interface org.neo4j.graphdb.GraphDatabaseService*. Obtenido de Uses of Interface org.neo4j.graphdb.GraphDatabaseService: http://neo4j.com/api_docs/2.0.3/org/neo4j/graphdb/class-use/GraphDatabaseService.html
- ezamudio. (05 de 02 de 2012). *Neo4J: Base de datos orientada a grafos*. Obtenido de Neo4J: Base de datos orientada a grafos : http://www.javamexico.org/blogs/ezamudio/neo4j_base_de_datos_orientada_grafos
- Fernández, J. d. (17 de 04 de 2015). *SlideShare*. Obtenido de Neo4j - A Graph Database : <http://es.slideshare.net/JavierdelaRosaFernan/neo4j-47124820>
- Flores, D. (02 de 2014). *Neo4j - Una guia rapido de Devniel.com*. Obtenido de Neo4j - Una guia rapido de Devniel.com: <http://devniel.com/wp-content/uploads/2014/02/Neo4j-Una-gu%C3%ADa-r%C3%A1pida-de-devniel.com-Parte-I.pdf>
- González-de-Artaza, D. L.-d.-l. (05 de 07 de 2012). *SlideShare*. Obtenido de Bases de Datos No Relacionales (NoSQL): Cassandra, CouchDB, MongoDB y Neo4j : <http://es.slideshare.net/dipina/nosql-cassandra-couchdb-mongodb-y-neo4j>
- Ian Robinson, J. W. (2015). *Graph Databases*. United States of America: O'Reilly Media, Inc.
- Morales, A. (21 de 10 de 2014). *Youtube*. Obtenido de Neo4j en NetBeans : <https://www.youtube.com/watch?v=cYkmaF1LCH0>
- Nasca, A. A. (12 de 08 de 2014). *Prezi*. Obtenido de Introducción a NoSQL - Graph DataBase Neo4j : <https://prezi.com/xaitejgkpsv/introduccion-a-nosql-graph-database-neo4j/>
- Sanchez, J. (04 de 04 de 2014). *Xurxo Developer*. Obtenido de Instalación de Neo4j y un primer vistazo al Neo4j Browser : <http://xurxodeveloper.blogspot.com/2014/04/neo4j-instalacion-y-neo4j-browser.html>
- Solis, S. M. (27 de 04 de 2013). *Herramientas para Big Data. Neo4J (Parte 2)*. Obtenido de Herramientas para Big Data. Neo4J (Parte 2): <http://santiagomarquezsolis.com/herramientas-para-big-data-neo4j-parte-2/>
- Stuetz, T. (12 de 02 de 2015). *neo4j with Java / Netbeans*. Obtenido de neo4j with Java / Netbeans : https://www.youtube.com/watch?v=-g-vCsZO_3g
- Technology, N. (2015). *Neo4j*. Obtenido de Download Neo4j: <http://neo4j.com/download/>
- Technology, N. (2015). *Neo4j*. Obtenido de Neo4j: <http://neo4j.com/>
- Tomas, E. (24 de 04 de 2014). *Qué es REST*. Obtenido de Qué es REST: <http://www.desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html>
- tutorialspoint. (2015). *Tutorialspoint Simply Easy Learning*. Obtenido de Neo4j Tutorial: <http://www.tutorialspoint.com/neo4j/index.htm>