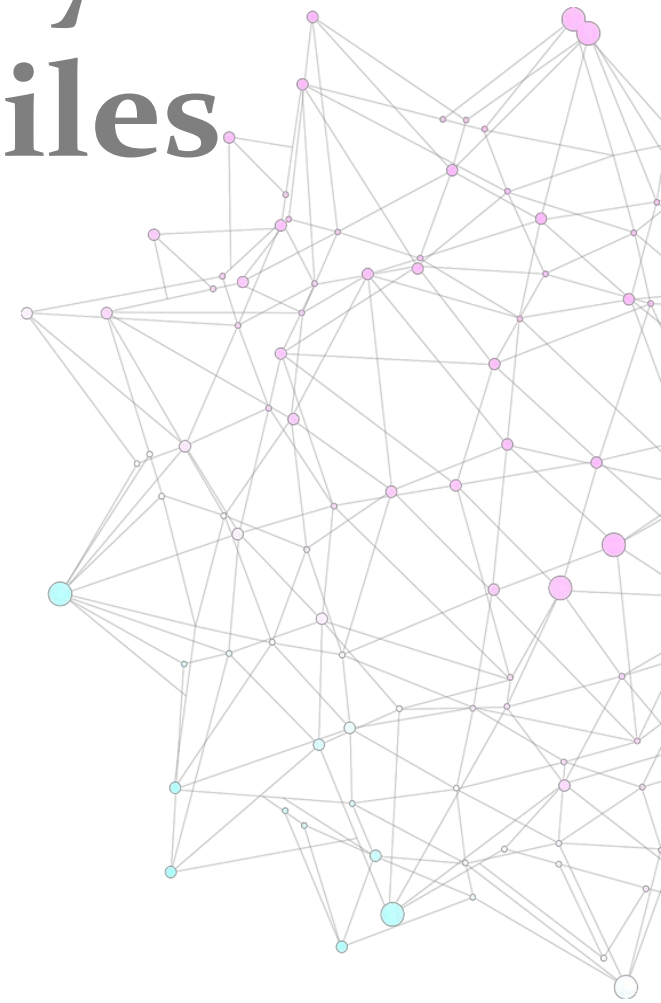
La Teoría de grafos

Una rama de las matemáticas y la computación que contiene miles de aplicaciones



Angie Katherine Reyes

UAN
UNIVERSIDAD
ANTONIO NARIÑO

Agenda

Introducción personal

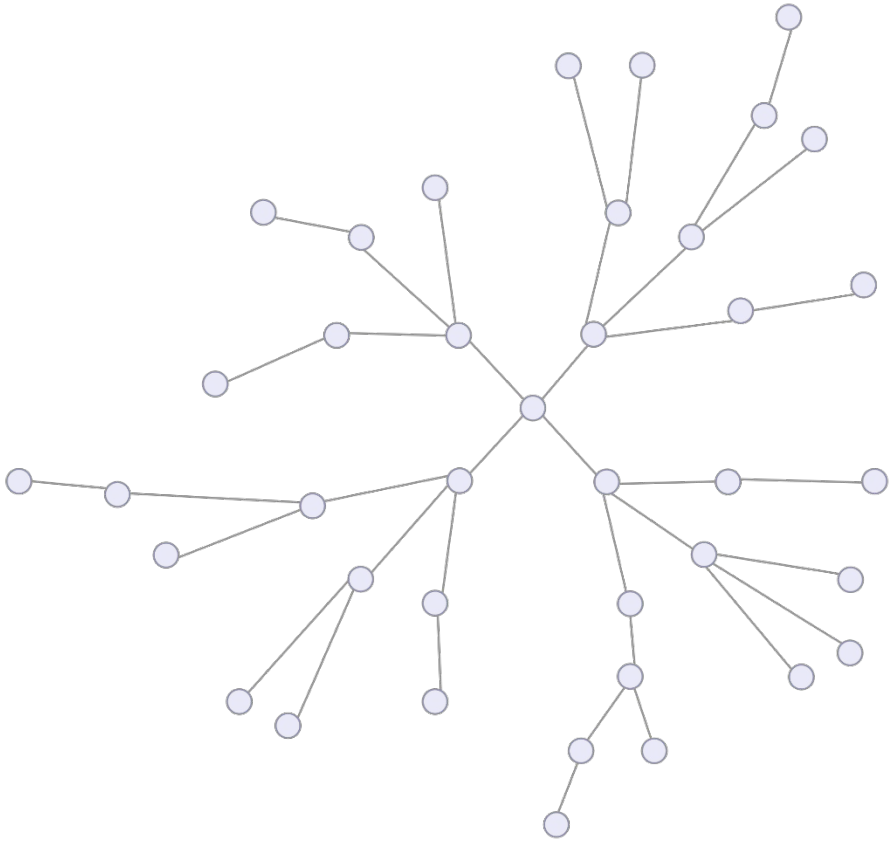
Introducción teoría de grafos

- Teoría de grafos
  - Teoría de redes
  - Análisis de redes

Introducción Neo4j

- Base de datos orientada a grafos: Neo4j
- Lenguaje de consulta: Chyper
- librería de análisis: NetworkX

Aplicación



Introducción personal

Acerca de mi

Colombiana

25 años

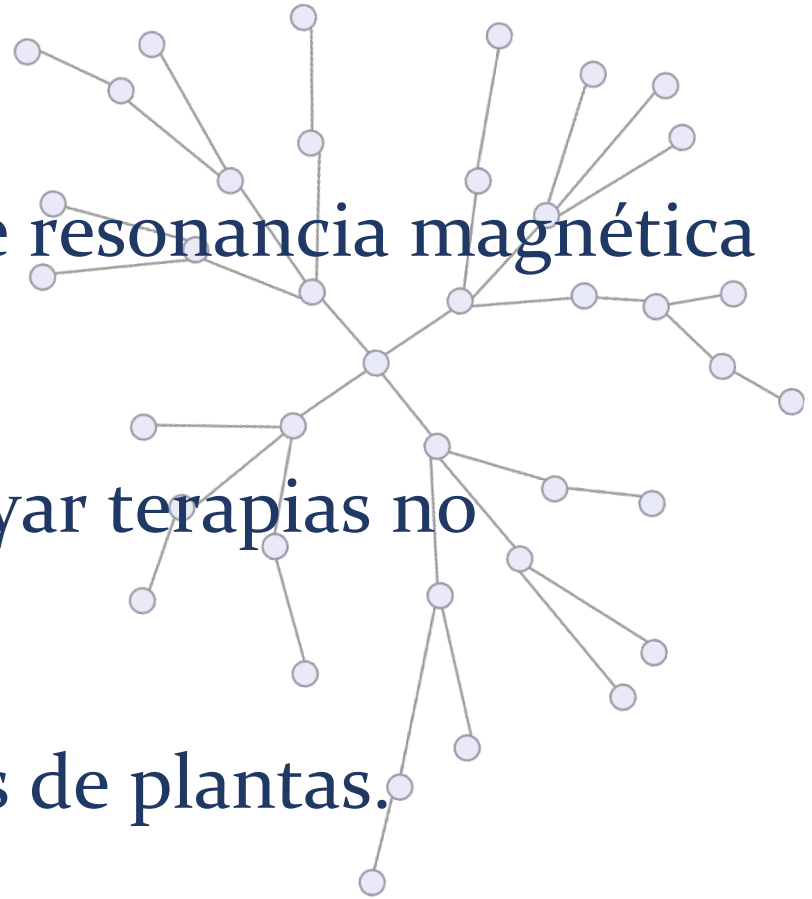
Ingeniera de Sistemas y computación

Chica TIC 2016

Estudiante de Doctorado

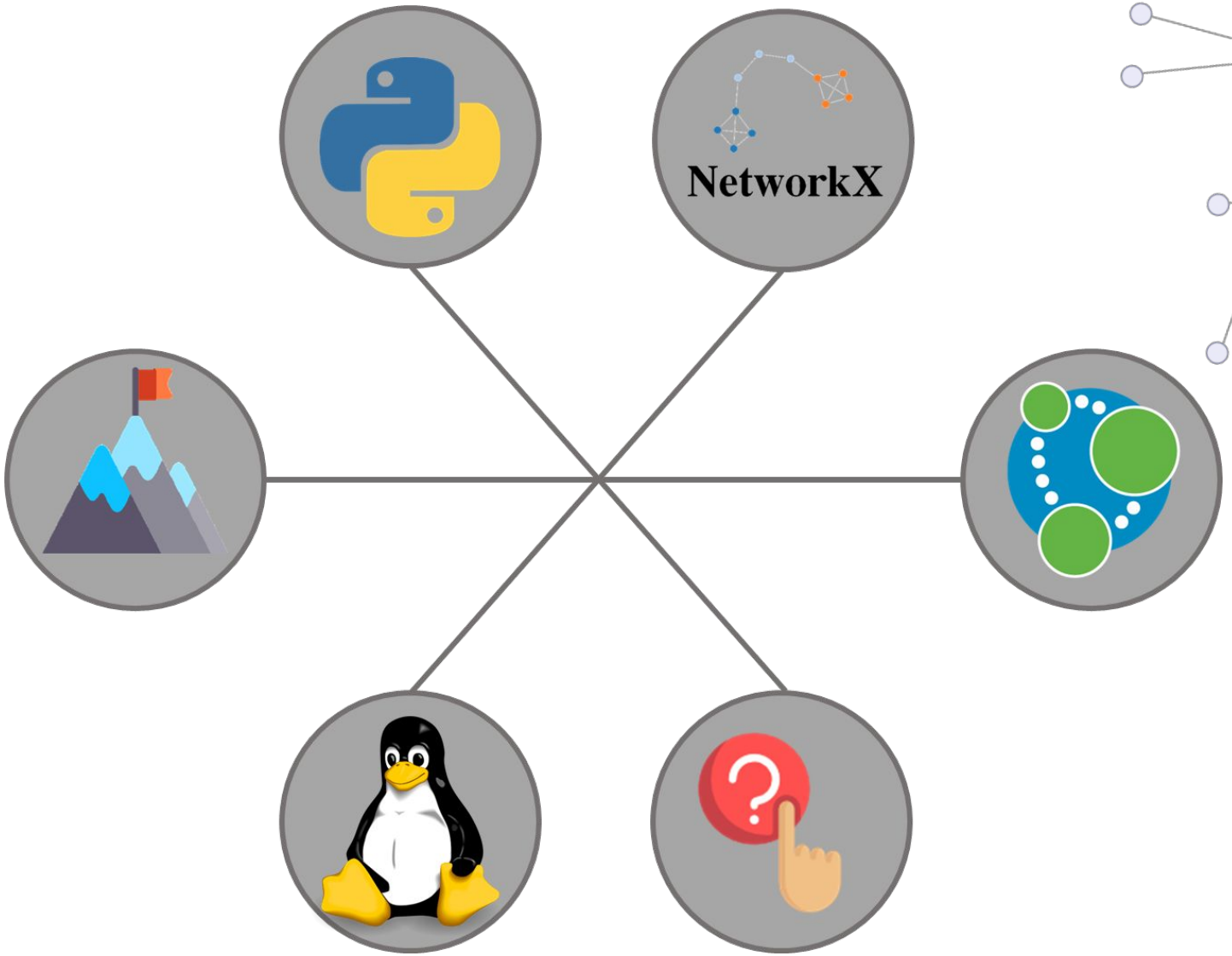
Desarrolladora Back-end

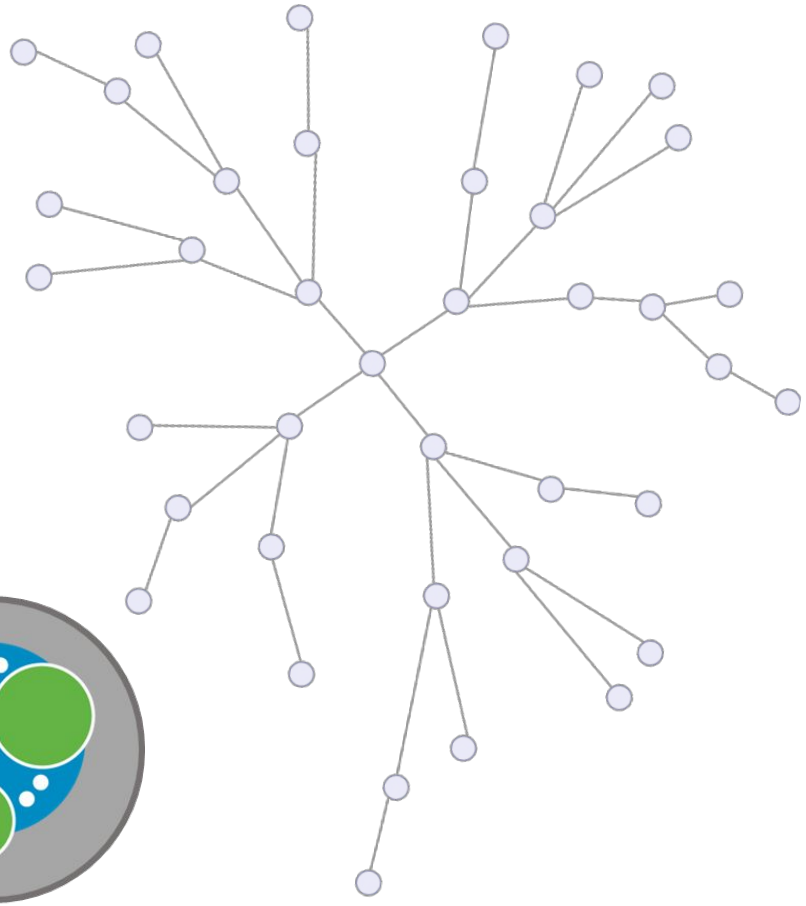
Experiencia



- **Support vector machines (SVM)** imágenes de resonancia magnética & audios de aves colombianas.
- Desarrollo de plataforma web y móvil para apoyar terapias no farmacológicas en pacientes con Alzheimer.
- **Deep Learning** en la identificación de especies de plantas.
- Líder de proyectos (Junior) en el departamento de Big data y Machine Learning en la industria.
- Aplicación de teoría de grafos para el modelado y optimización de microrredes eléctricas inteligentes.

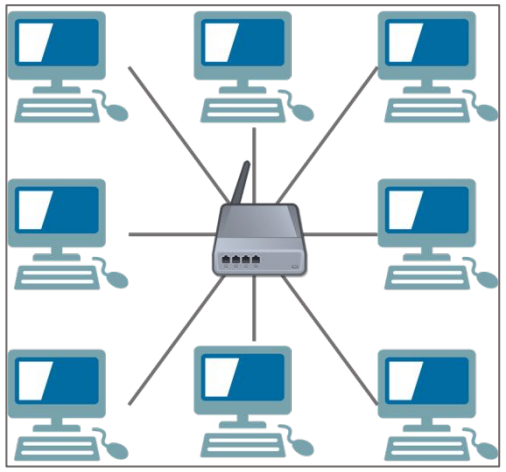
Acerca de la charla

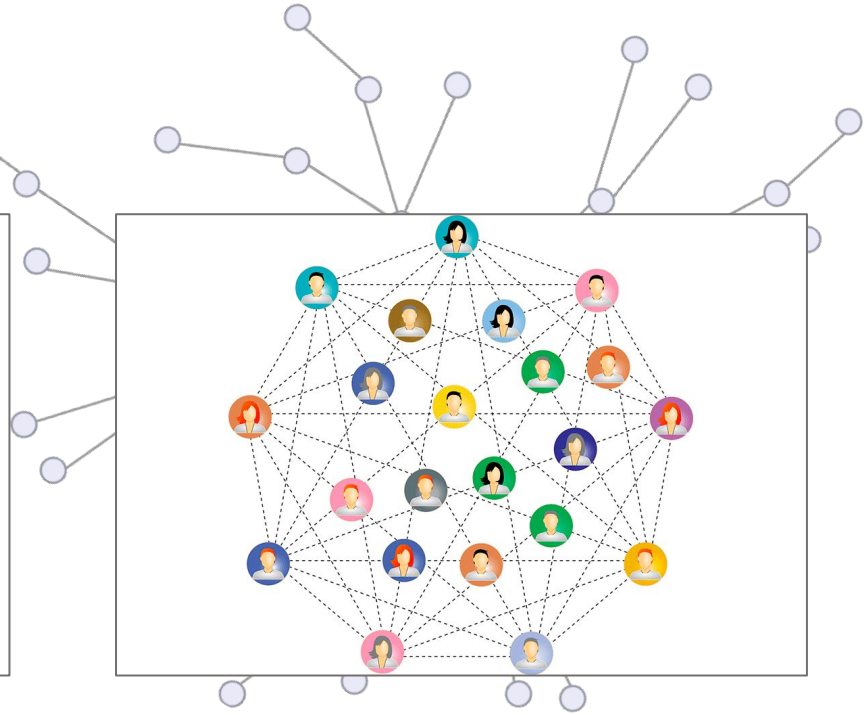


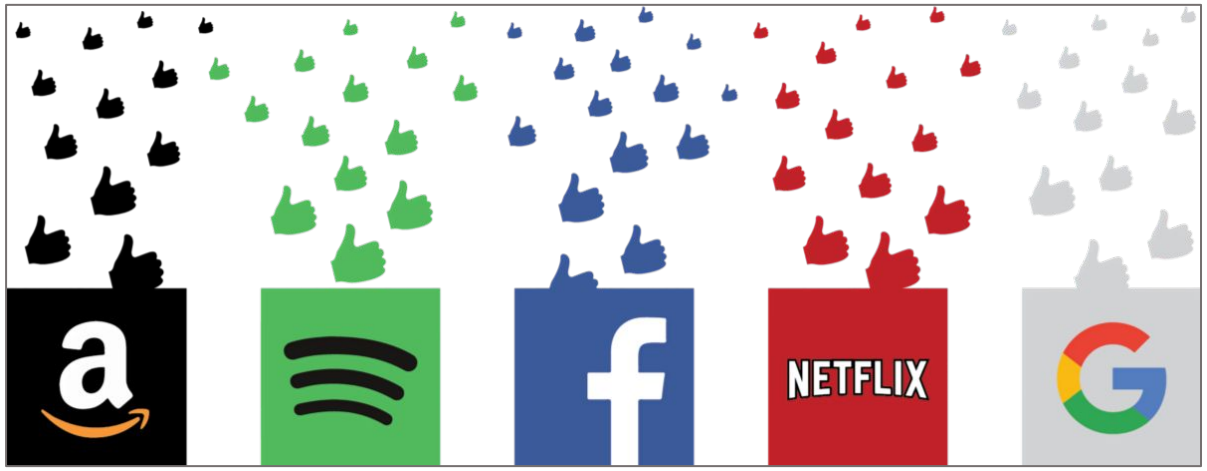


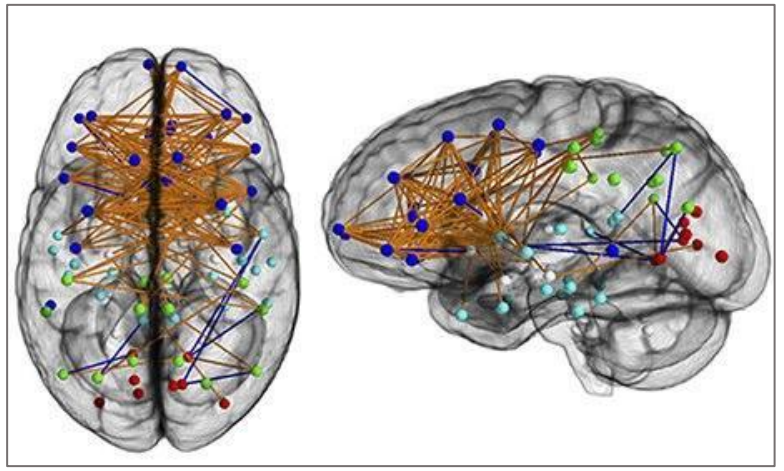
Introducción

¿Por qué los grafos?



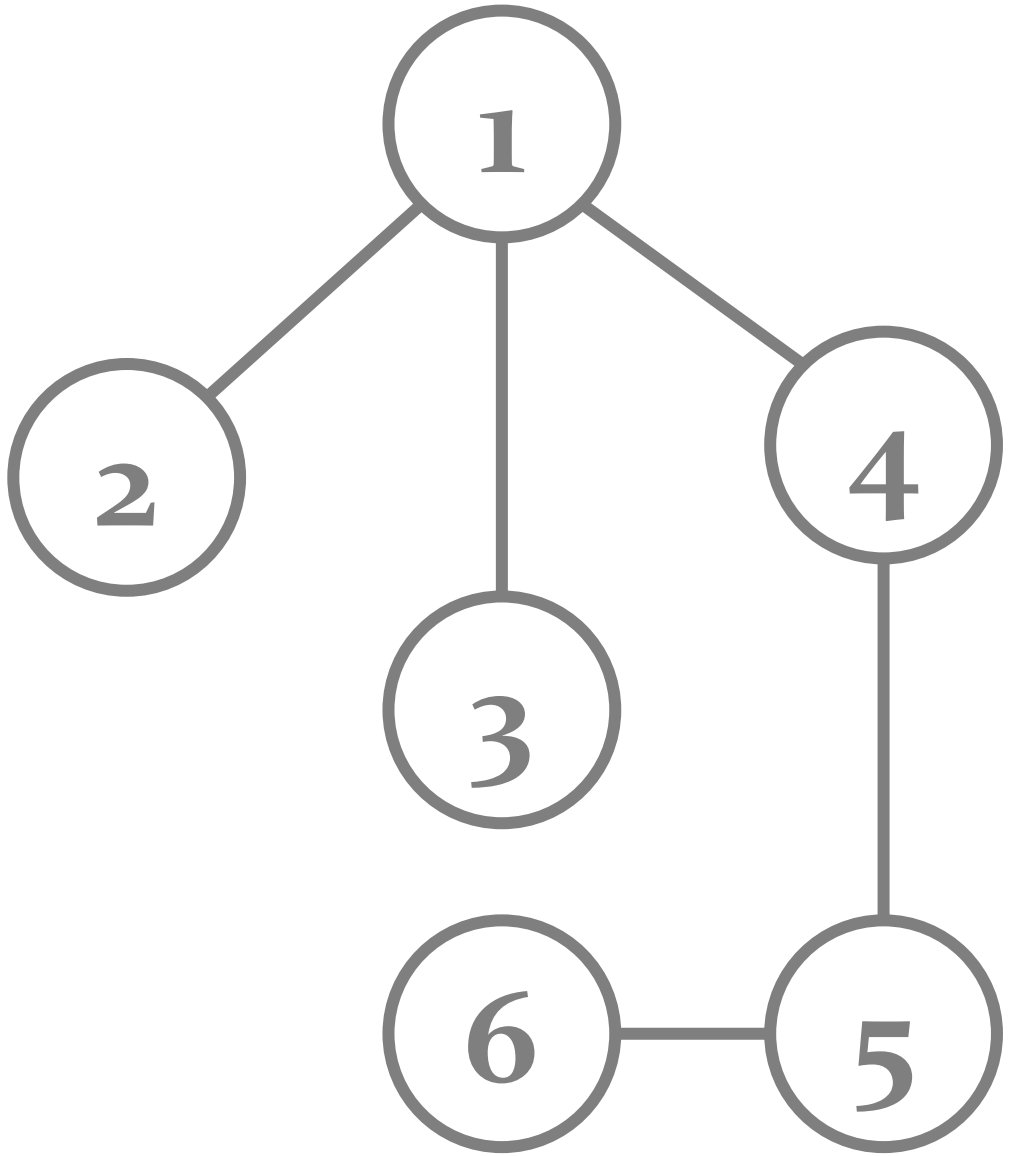






http://johnbald.typepad.com/language/2013/12/national-academy-of-sciences-mens-and-womens-brains-are-much-as-we-think-they-are.html

Teoría de grafos

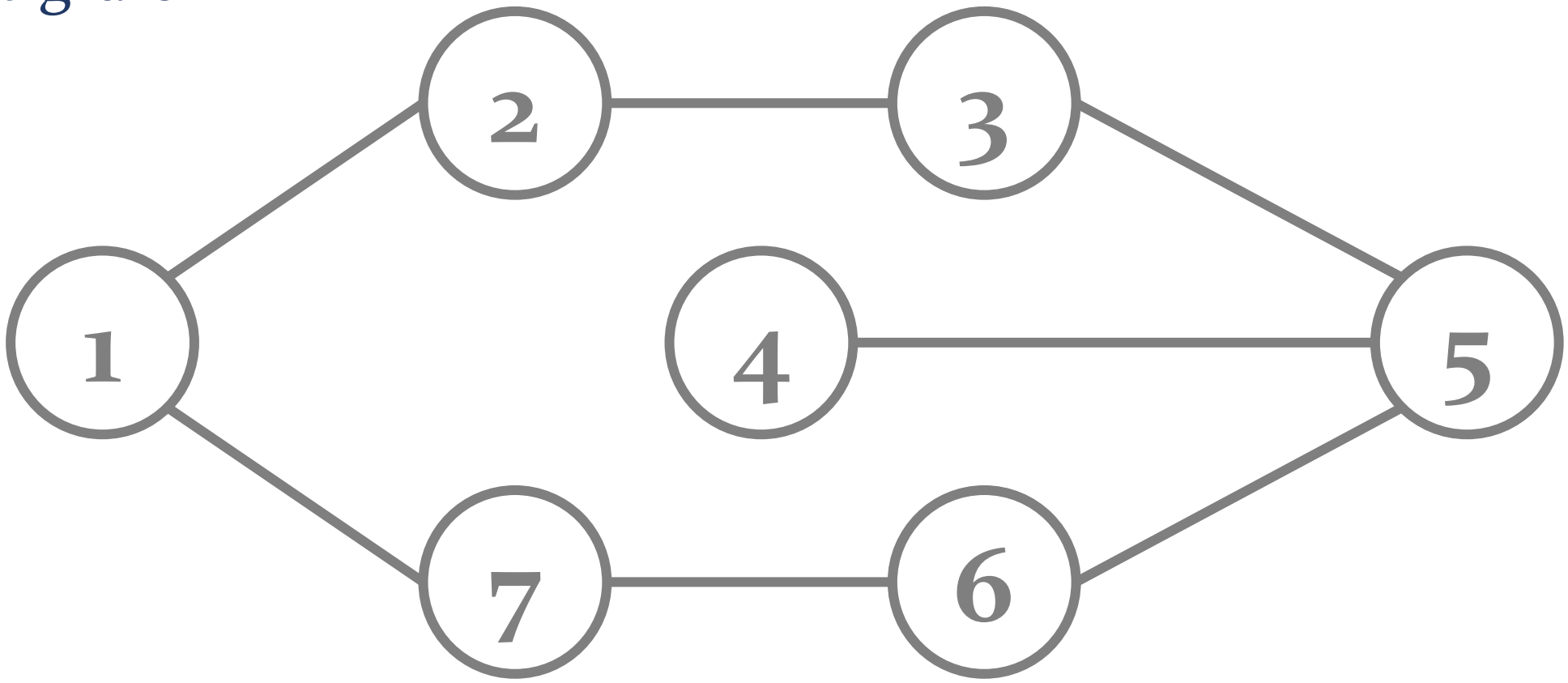


Un grafo es un conjunto de puntos que son denominados nodos y de líneas que son llamados aristas ¡y eso es todo, muy sencillo!

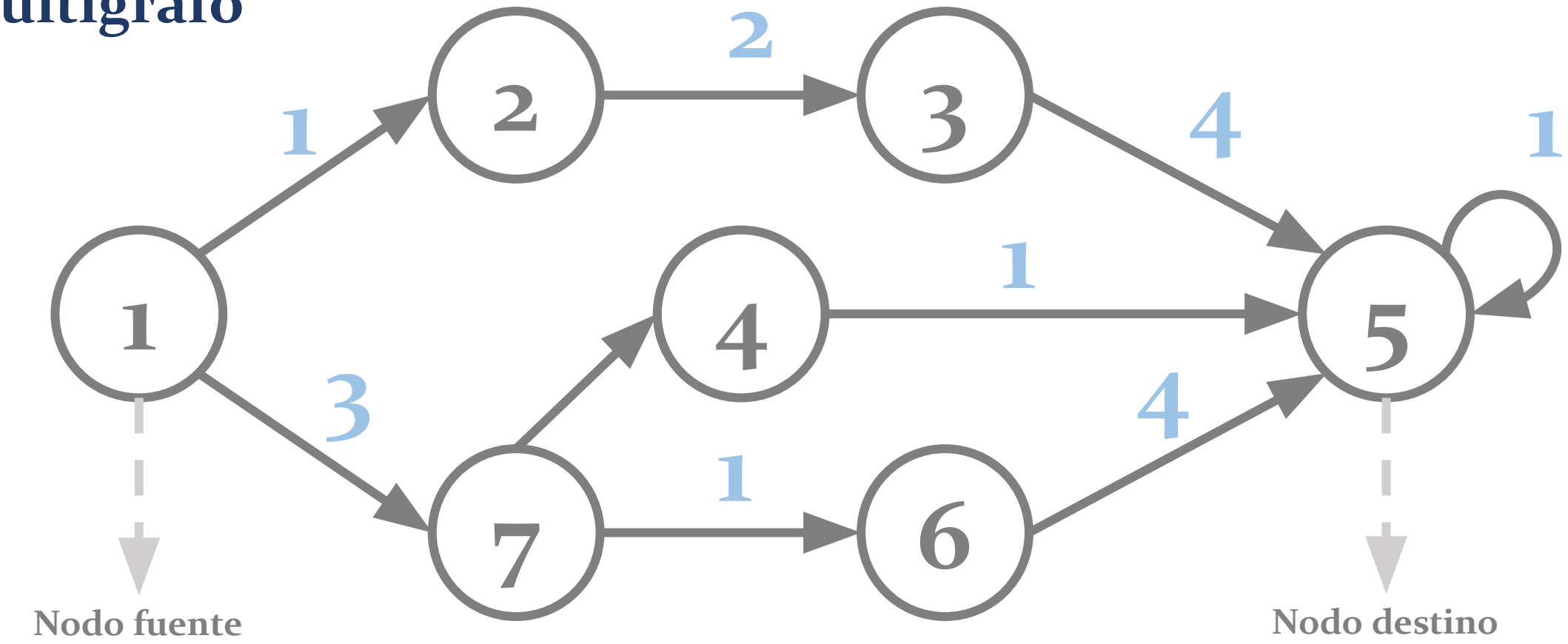
G=(V,E)

V, es un conjunto de vértices o nodos
E, es un conjunto de aristas o arcos, que relacionan los nodos.

- **Grafo simple**
- **Grafo no dirigido**
- Grafo ponderado
- Grafo dirigido (digrafo)
- Multígrafo



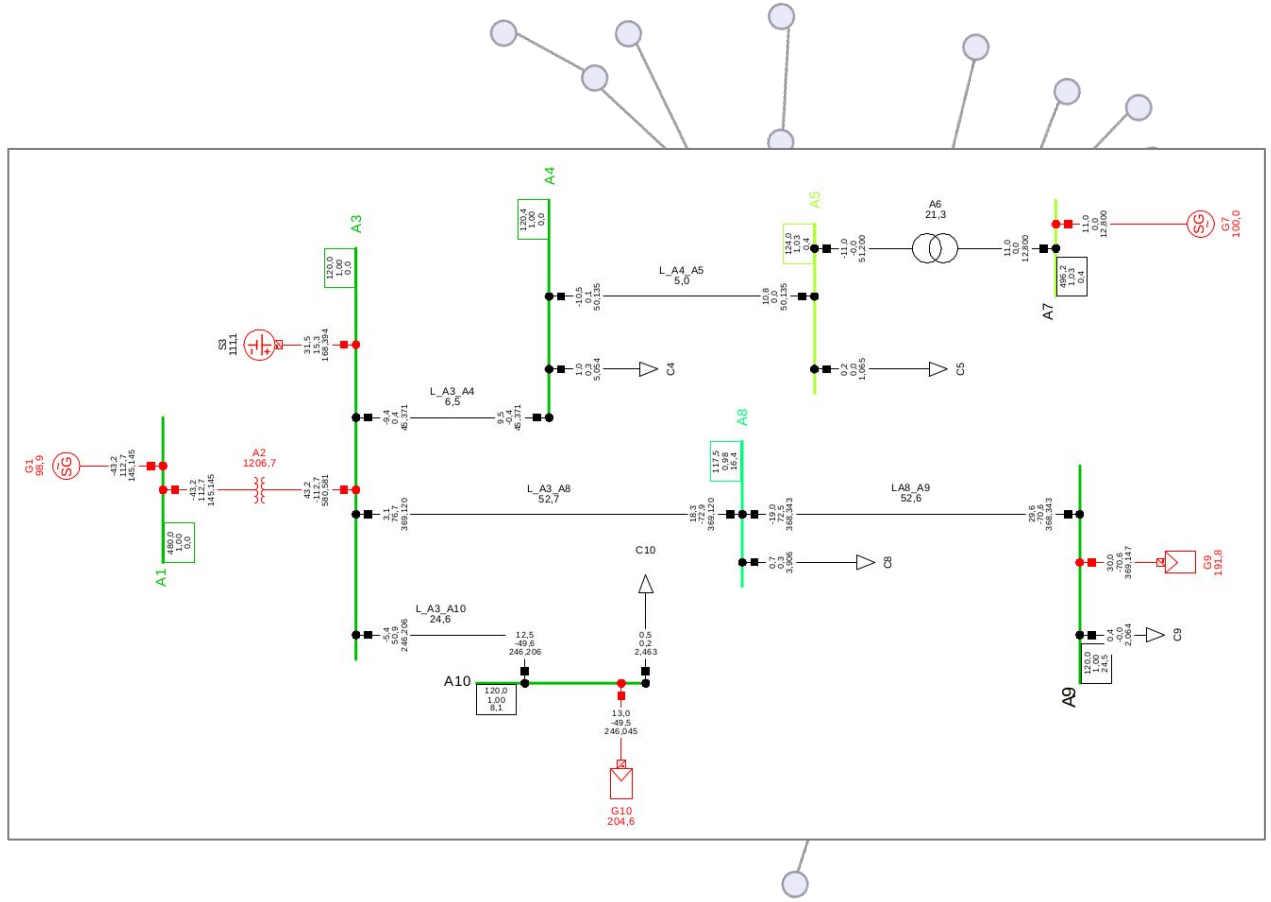
- Grafo simple
- Grafo no dirigido
- **Grafo ponderado**
- **Grafo dirigido (digrafo)**
- **Multígrafo**

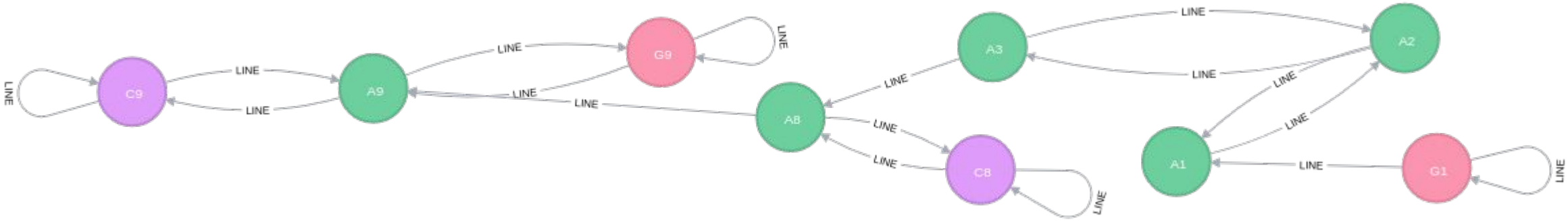


Teoría de redes

Analizar las redes mediante la teoría de grafos.

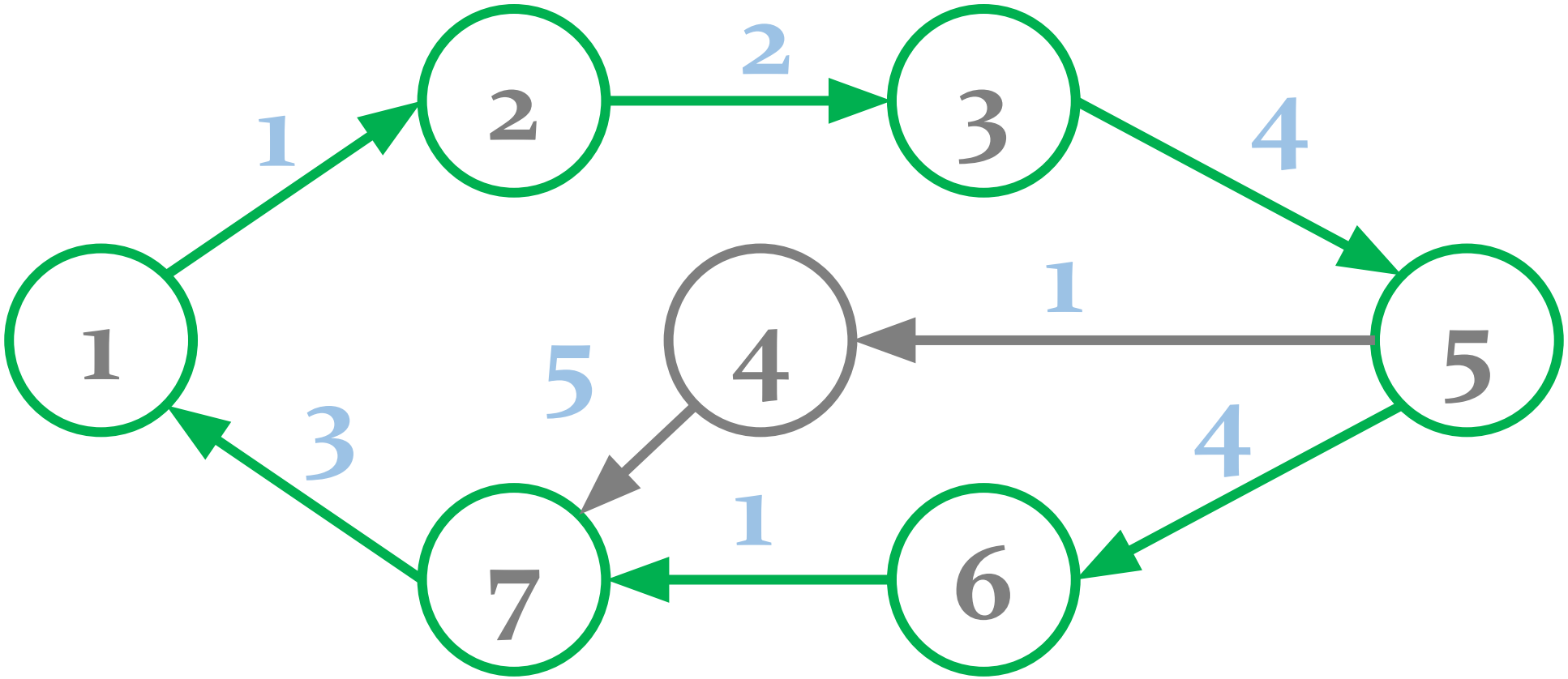
Una red representa un grafo con atributos.





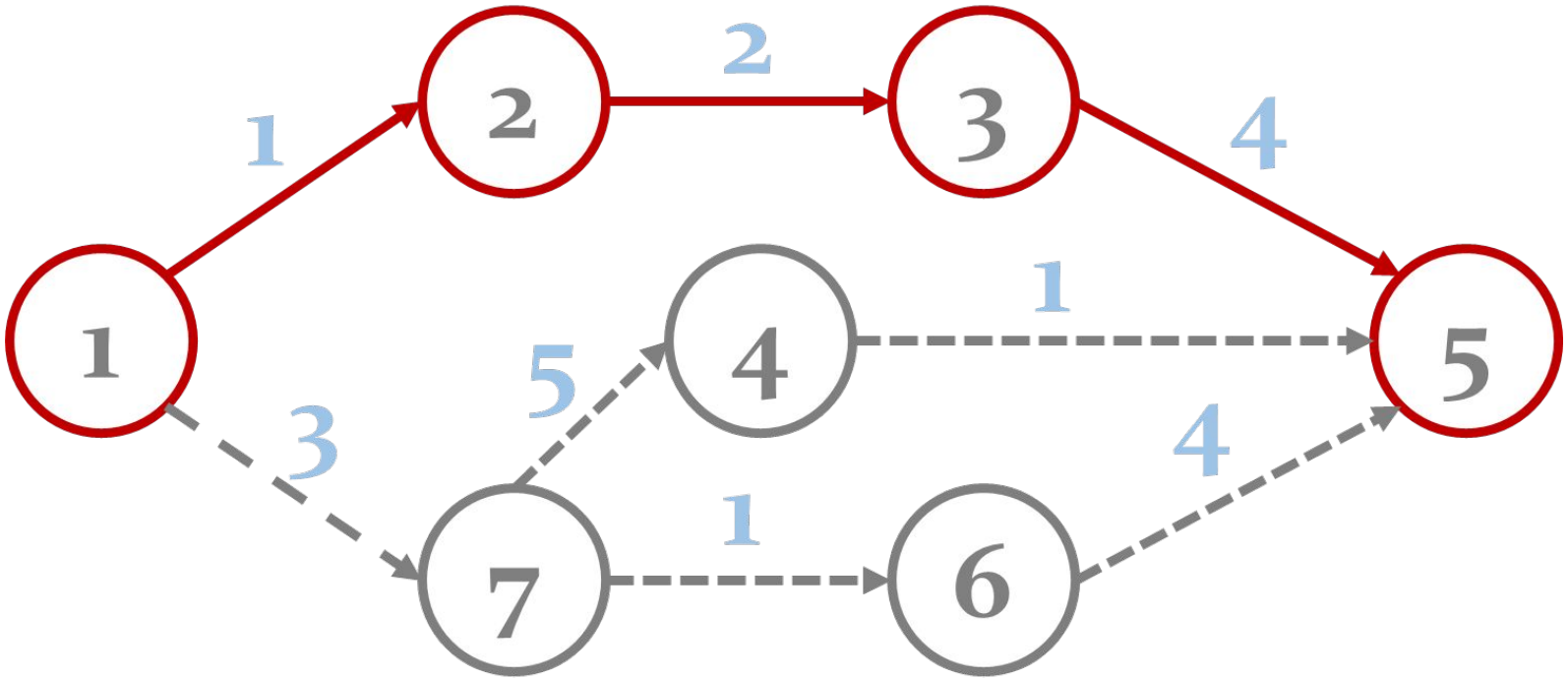
- **Qué es una cadena?**
- **Qué es un camino?**
- **Qué es un ciclo?**
- Cuál es el camino más corto?

Cadena =[1-2,2-3,3-5]
Camino = [1,2,3,5]
Ciclo = [1,2,3,5,6,7,1]



- Qué es una cadena?
- Qué es un camino?
- Qué es un ciclo?
- **Cuál es el camino más corto?**

C1 =[1,2,3,5] = (1+2+4) = 7
C2 = [1,7,6,5] = (3+1+4) = 8
C3 = [1,7,4,5] = (3+5+1) = 9

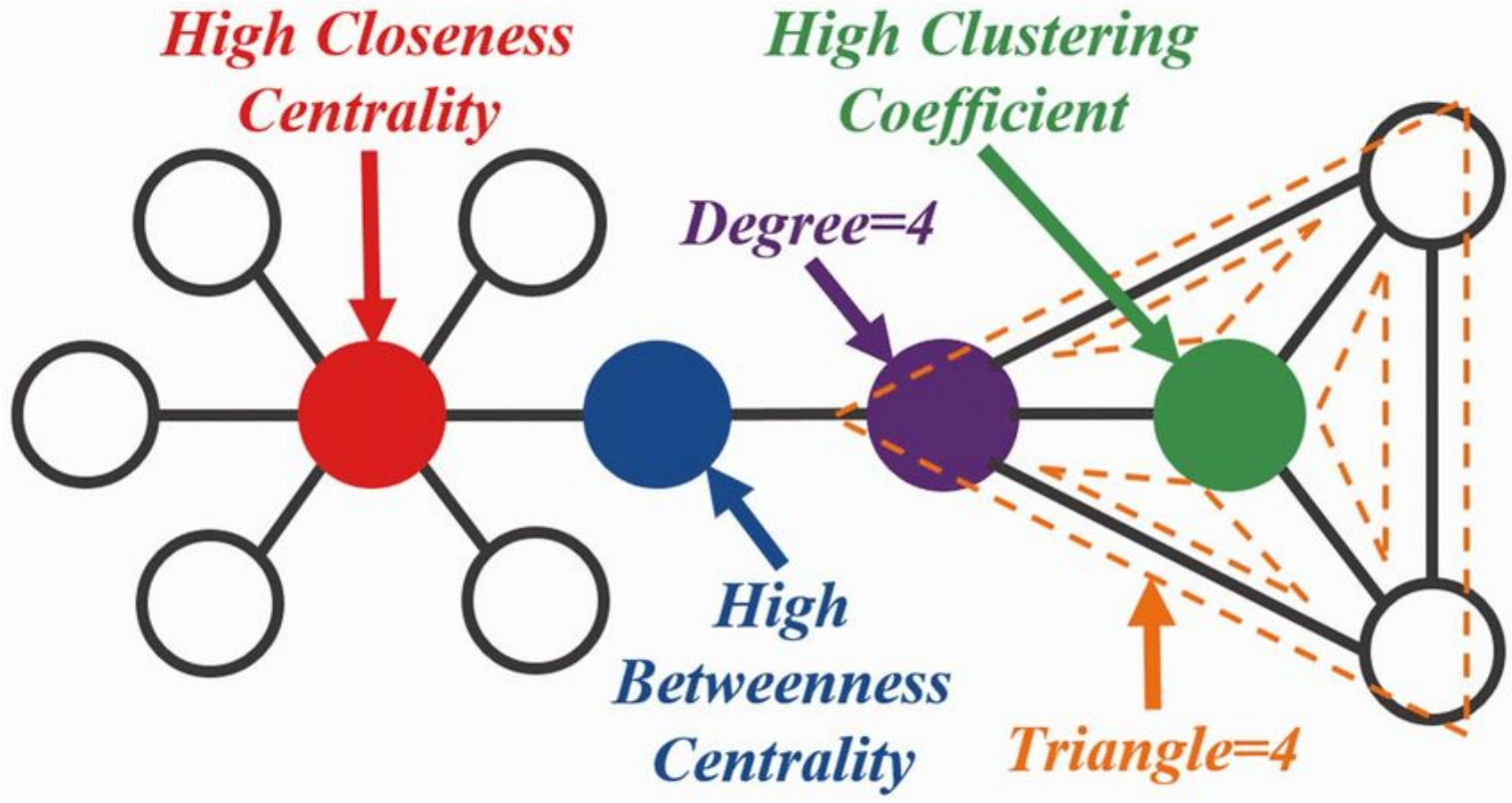


Análisis de redes

Contextualización y definición de métricas

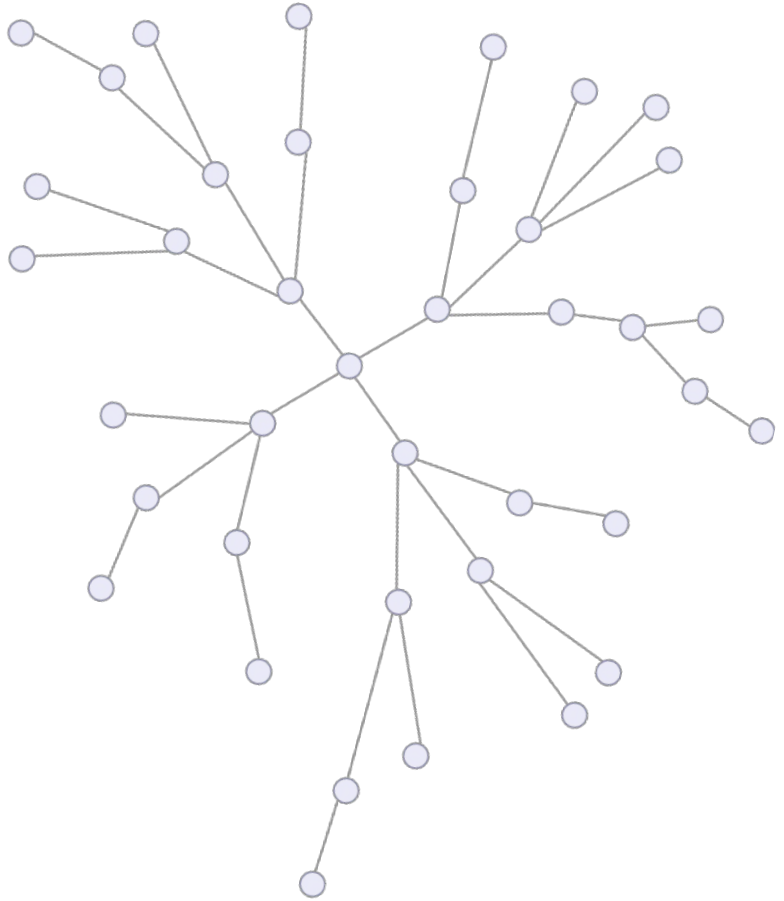
Las métricas ayudan a entender la red, la importancia y rol de las entidades y sus relaciones.

Las medidas de centralidad son las más usadas porque ayudan en la definición de los nodos centrales: nodos con un acceso más fácil y rápido a los demás actores de la red

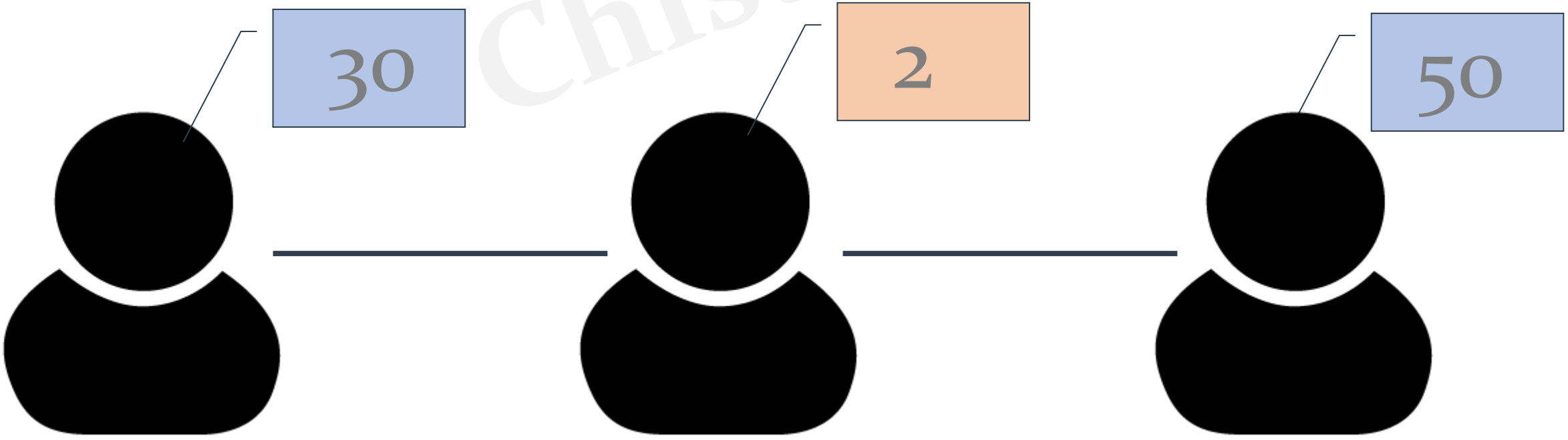


https://aksakalli.github.io/2017/07/17/network-centrality-measures-and-their-visualization.html

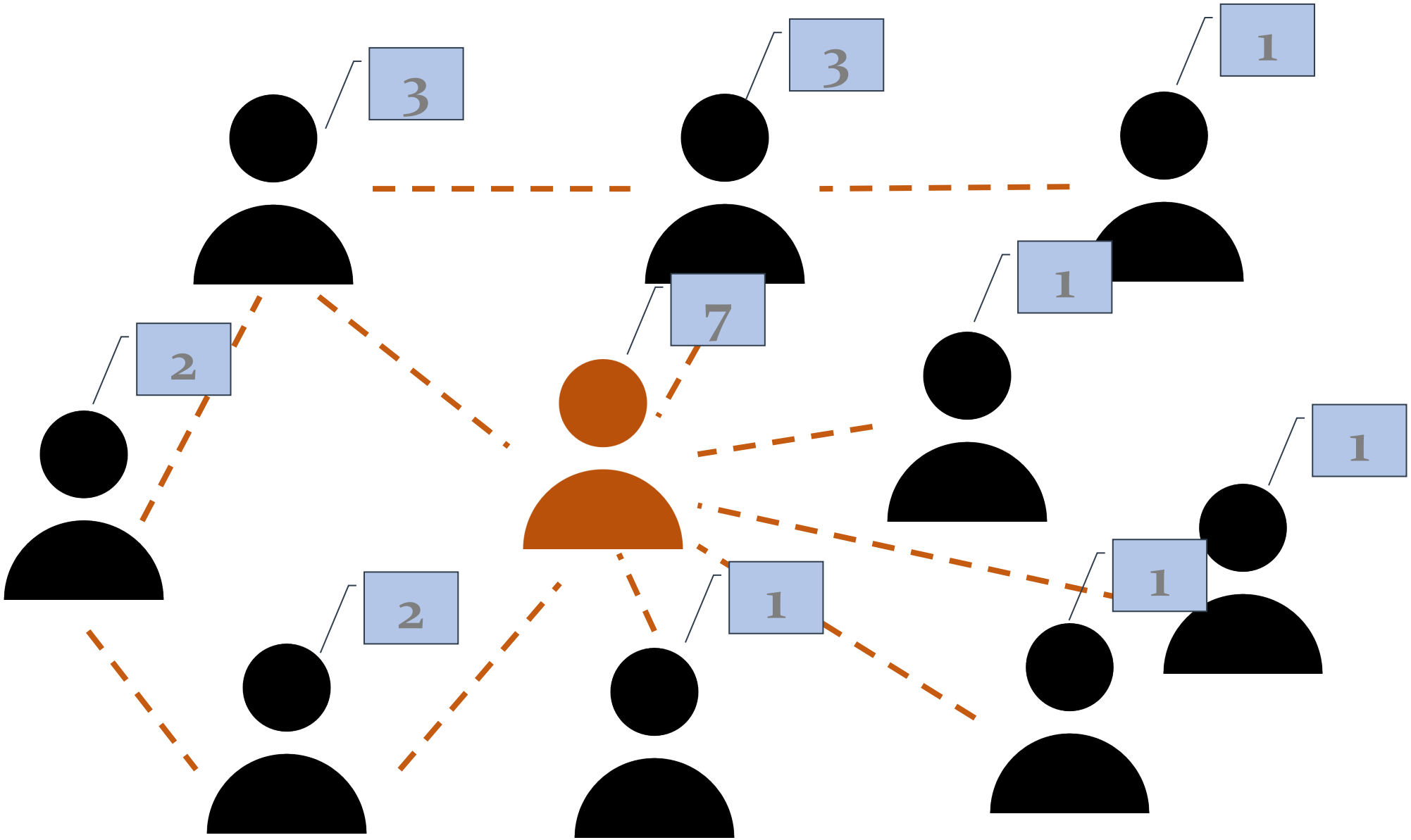
¿Cuál de tus amigos es el más importante en las redes sociales?



¿Cuál de tus amigos es el más importante?



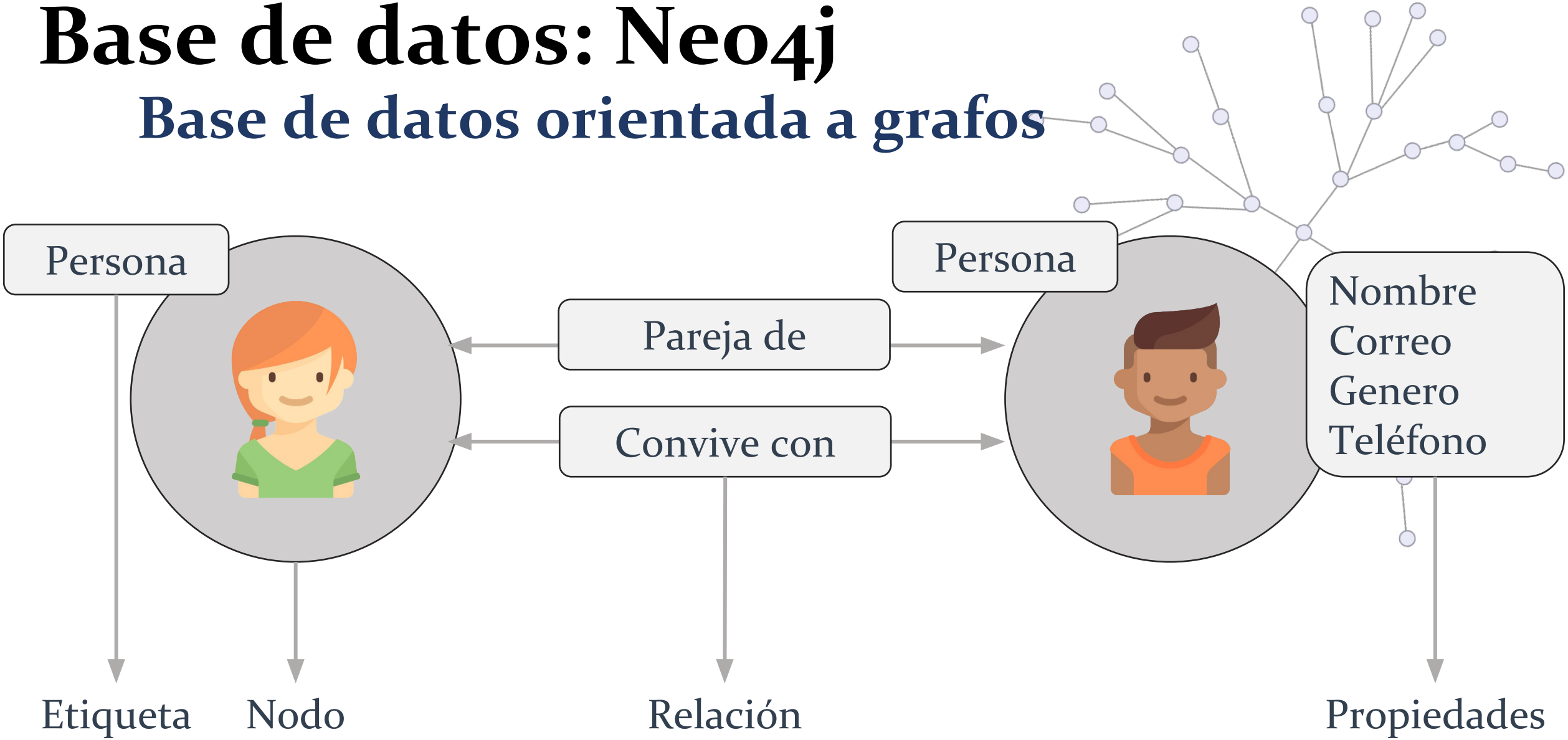
¿Cuál de tus amigos es el más importante?



Introducción Neo4j

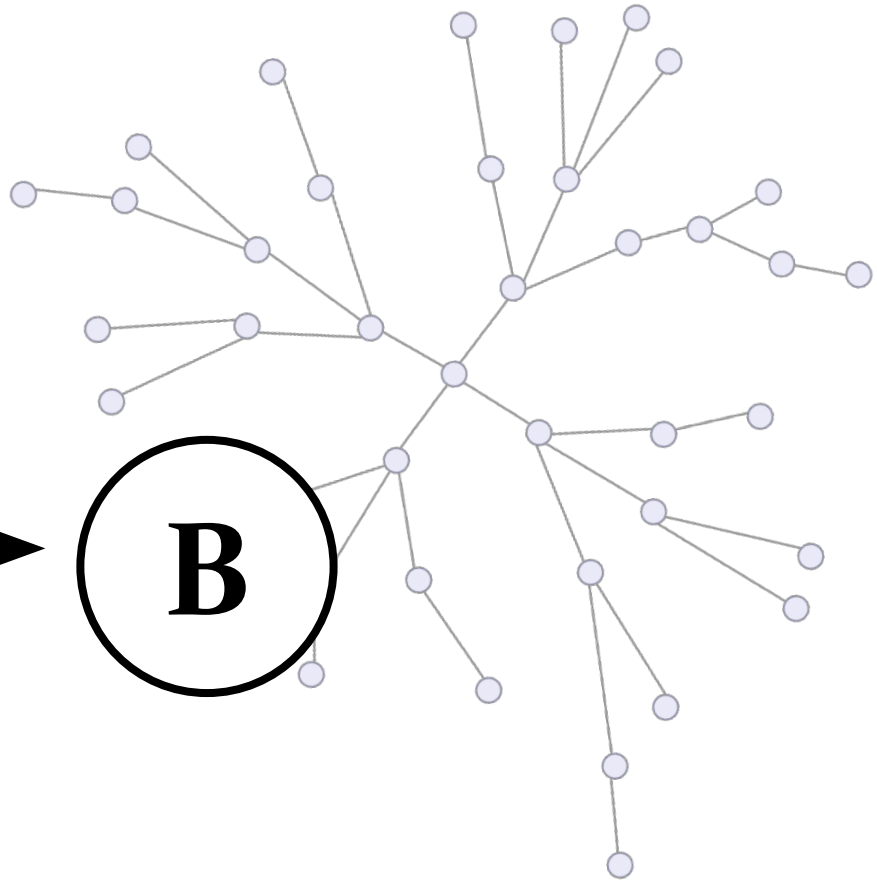
Base de datos: Neo4j

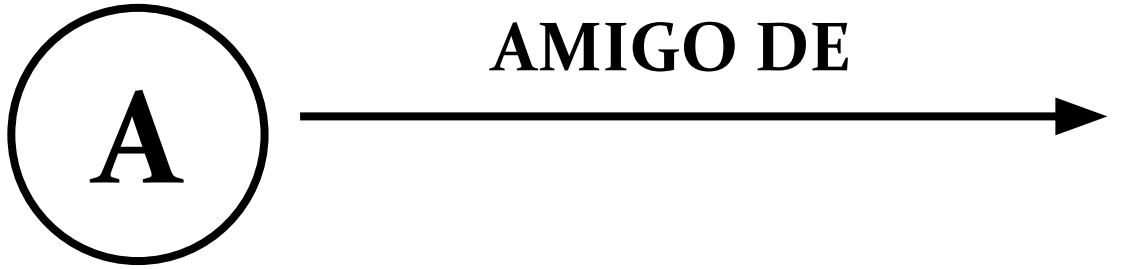
Base de datos orientada a grafos



Cypher

Lenguaje de consultas



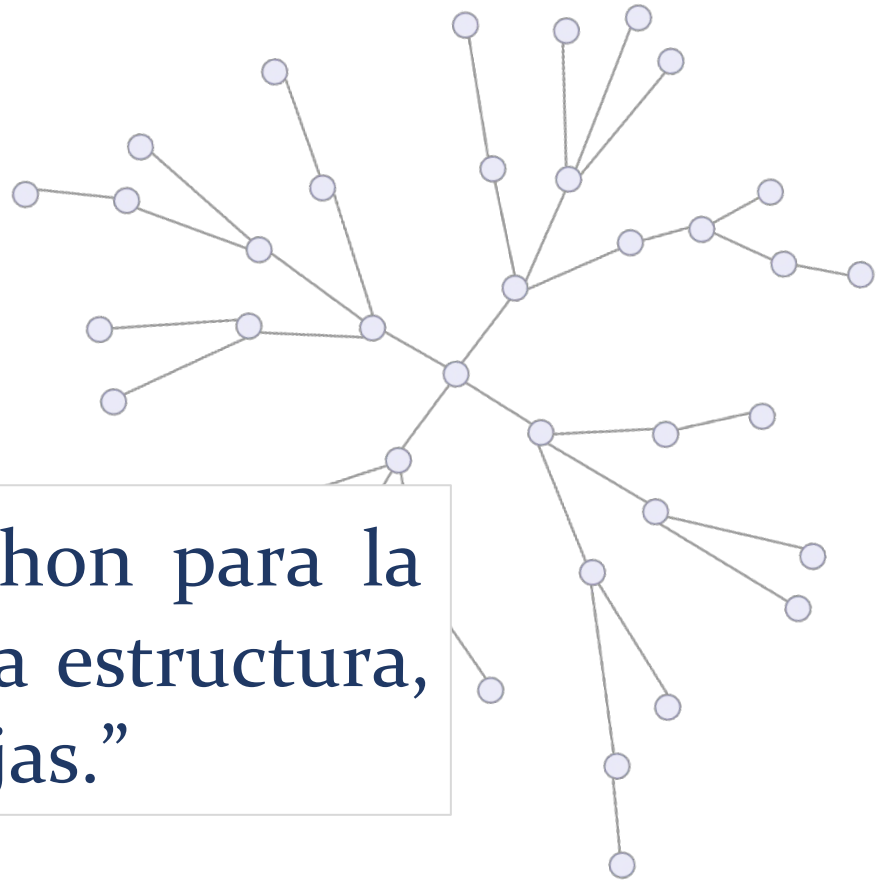


(A) –[: AMIGO DE] -> (B)

https://neo4j.com/developer/cypher-query-langua

NetworkX

Software para redes complejas



"**NetworkX** es un paquete de Python para la creación, manipulación y estudio de la estructura, dinámica y funciones de redes complejas."

```
>>> import networkx as nx
>>> G = nx.Graph()
```

https://networkx.github.io/

Sintaxis de nodo

```
()
(matrix)
(:Movie)
(matrix:Movie)
(matrix:Movie {title: "The Matrix"})
(matrix:Movie {title: "The Matrix", released: 1997})
```

https://neo4j.com/docs/developer-manual/current/get-started/cypher/#cypher-intro-patterns-node-syntax

Sintaxis de relación

```
-->
-[role]->
-[:ACTED_IN]->
-[role:ACTED_IN]->
-[role:ACTED_IN {roles: ["Neo"]}]->
```

https://neo4j.com/docs/developer-manual/current/get-started/cypher/#cypher-intro-patterns-node-syntax

Sintaxis de búsqueda

Retornar todos los nodos

```
MATCH (n)
RETURN n
```

Retornar los nodos de tipo película

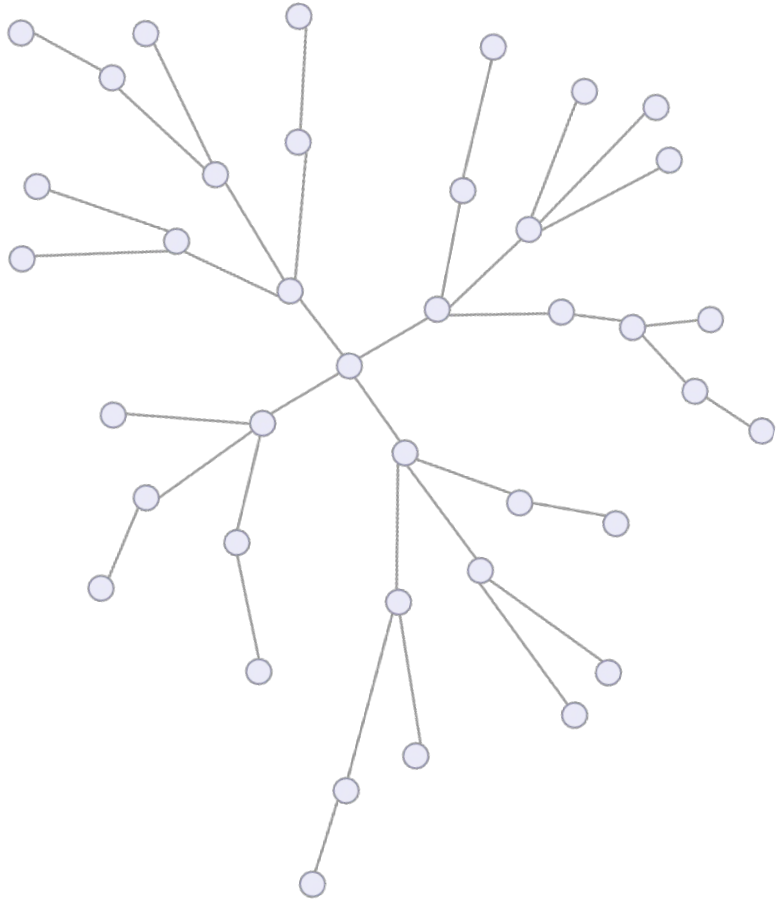
```
MATCH (movie:Movie)
RETURN movie.title
```

Retornar nodos con relación

```
MATCH (director { name: 'Oliver Stone' })--(movie)
RETURN movie.title
```

https://neo4j.com/docs/developer-manual/current/cypher/clauses/match/

Ejemplo



Vamos a modelar la relación de **Lionel Messi** con el **Fútbol Club Barcelona**

https://tn.com.ar/deportes/after-play/el-inesperado-cambio-de-look-de-lionel-messi-que-sorprende-todos-chau-barba_898937

Modelo

Código

```
CREATE (j:Jugador { nombre:'Lionel Messi', nacimiento:'24-06-1987', nacionalidad:'Argentina' })
CREATE (c:Club {nombre:'F. C. Barcelona', fundado:'29-11-1899'})
CREATE (j)-[r:jugador_de {desde:'16-10-2004', liga:'Primera división de España', posición:'delantero'}]->(c)
RETURN type(r)
```

Aplicación

py2neo

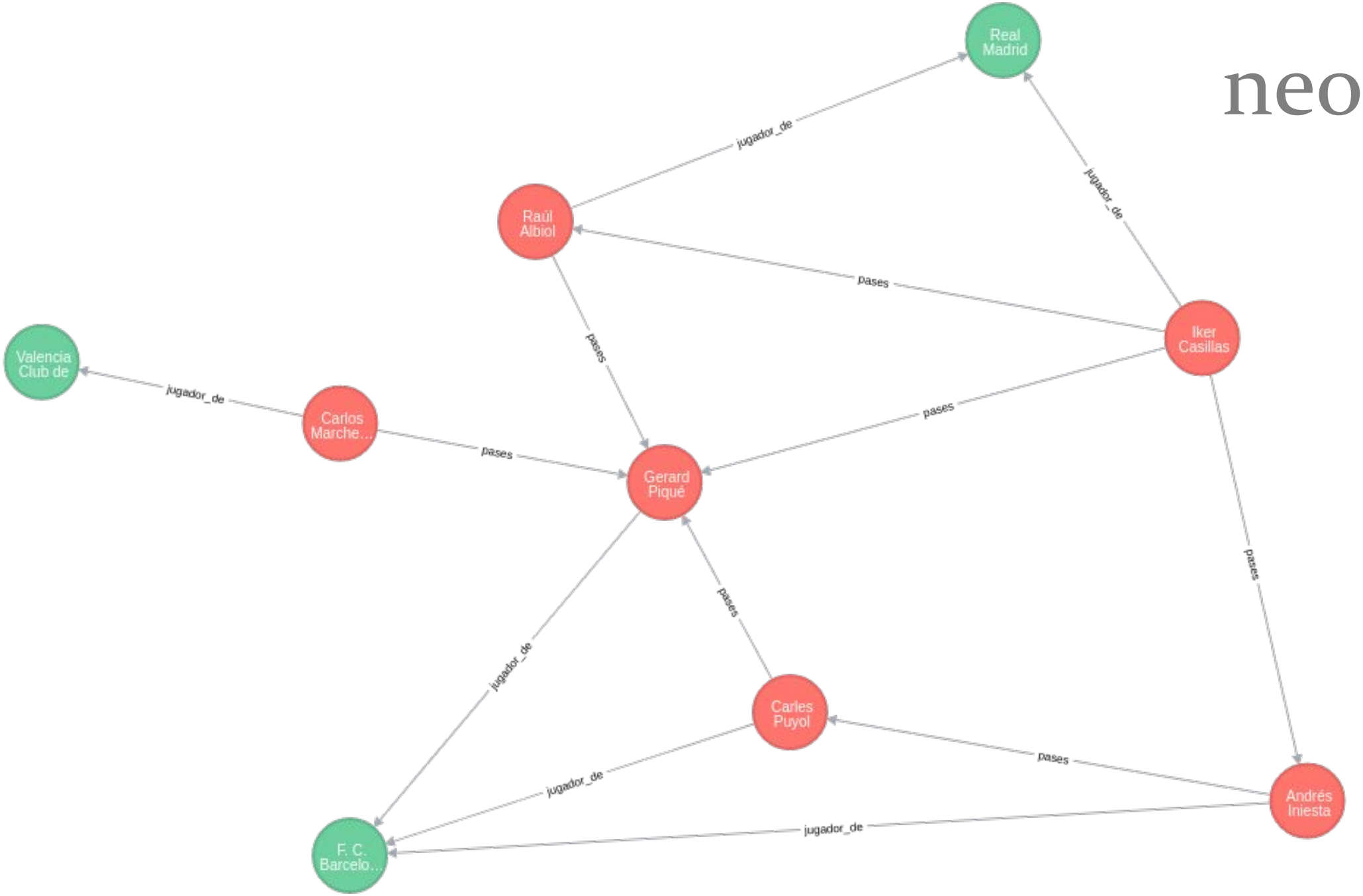
```
In [1]: import os
        from py2neo import Graph
        from py2neo import Node, Relationship
```

```
In [2]: def connectGraphDatabase():
            graph = Graph(password="pydata2018")
            return graph
```

```
In [3]: query = "CREATE (j1:Jugador { nombre: 'Iker Casillas'}) \n"
        query += "CREATE (j2:Jugador { nombre: 'Raúl Albiol'}) \n"
        query += "CREATE (j3:Jugador { nombre: 'Gerard Piqué'}) \n"
        query += "CREATE (j4:Jugador { nombre: 'Carlos Marchena'}) \n"
        query += "CREATE (j5:Jugador { nombre: 'Carles Puyol'}) \n"
        query += "CREATE (j6:Jugador { nombre: 'Andrés Iniesta'}) \n"

        query += "CREATE (rm:Club {nombre: 'Real Madrid'})\n"
        query += "CREATE (fcb:Club {nombre: 'F. C. Barcelona'})\n"
        query += "CREATE (vcf:Club {nombre: 'Valencia Club de Fútbol'})\n"

        query += "CREATE (j1)-[r1:jugador_de {posición: 'portero'}]->(rm)\n"
        query += "CREATE (j2)-[r2:jugador_de {posición: 'defensa'}]->(rm)\n"
        query += "CREATE (j3)-[r3:jugador_de {posición: 'defensa'}]->(fcb)\n"
        query += "CREATE (j4)-[r4:jugador_de {posición: 'defensa'}]->(vcf)\n"
        query += "CREATE (j5)-[r5:jugador_de {posición: 'defensa'}]->(fcb)\n"
        query += "CREATE (j6)-[r6:jugador_de {posición: 'mediocampista'}]->(fcb)\n"

        query += "CREATE (j1)-[p1:pases {num_pases: '30'}]->(j2)\n"
        query += "CREATE (j1)-[p2:pases {num_pases: '2'}]->(j3)\n"
        query += "CREATE (j4)-[p3:pases {num_pases: '3'}]->(j3)\n"
        query += "CREATE (j5)-[p4:pases {num_pases: '1'}]->(j3)\n"
        query += "CREATE (j6)-[p5:pases {num_pases: '6'}]->(j5)\n"
        query += "CREATE (j2)-[p6:pases {num_pases: '26'}]->(j3)\n"
        query += "CREATE (j1)-[p7:pases {num_pases: '12'}]->(j6)"
```

```
In [4]: G = connectGraphDatabase()
        G.run(query)

Out[4]: <py2neo.database.Cursor at 0x7f97d1f02588>
```

neo4j


Real Madrid
Raúl Albiol
Iker Casillas
Valencia Club de
Carlos Marche...
Gerard Piqué
Carles Puyol
Andrés Iniesta
F. C. Barcelo...
jugador_de
pases

Club al que pertenece el jugador Raúl Albiol

```
MATCH (Jugador { nombre: 'Raúl Albiol' })--(c:Club)
RETURN c.nombre
```

"c.nombre"
"Real Madrid"

Jugadores del club F.C Barcelona

```
MATCH (c:Club { nombre: 'F. C. Barcelona' })<-[:jugador_de]-(j:Jugador)
RETURN j.nombre
```

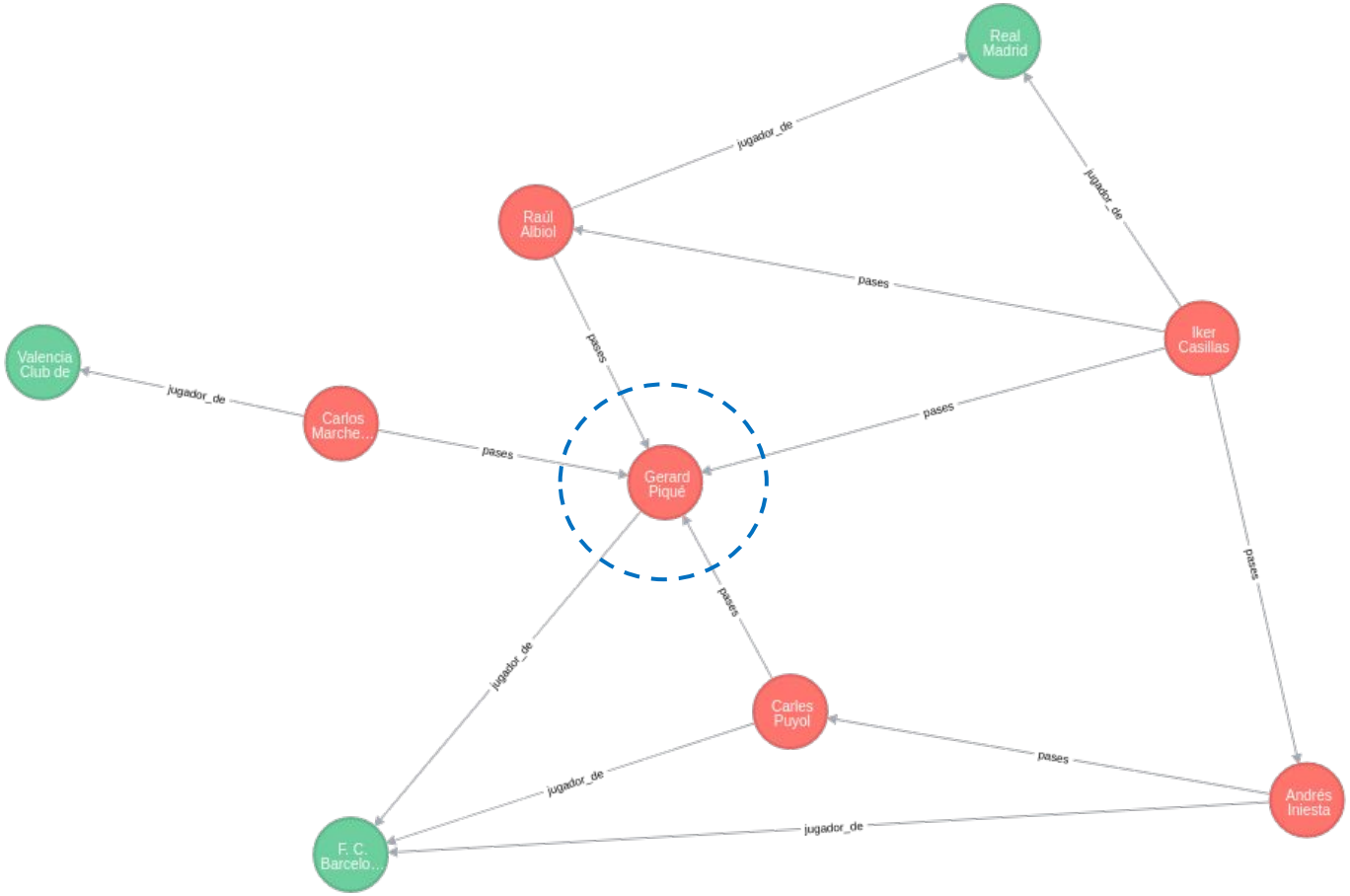
"j.nombre"
"Andrés Iniesta"
"Carles Puyol"
"Gerard Piqué"

NetwokX

```
In [1]: import networkx as nx
        import matplotlib.pyplot as plt

In [2]: G = nx.Graph()

In [3]: jugadores = dict()
        jugadores['Iker Casillas'] = 1
        jugadores['Raúl Albiol'] = 2
        jugadores['Gerard Piqué'] = 3
        jugadores['Carlos Marchena'] = 4
        jugadores['Carles Puyol'] = 5
        jugadores['Andrés Iniesta'] = 6

In [4]: G.add_edges_from([(jugadores['Iker Casillas'],
                           jugadores['Raúl Albiol'],{'num_pases':30})])
        G.add_edges_from([(jugadores['Iker Casillas'],
                           jugadores['Gerard Piqué'],{'num_pases':2})])
        G.add_edges_from([(jugadores['Carlos Marchena'],
                           jugadores['Gerard Piqué'],{'num_pases':3})])
        G.add_edges_from([(jugadores['Carles Puyol'],
                           jugadores['Gerard Piqué'],{'num_pases':1})])
        G.add_edges_from([(jugadores['Raúl Albiol'],
                           jugadores['Gerard Piqué'],{'num_pases':4})])
        G.add_edges_from([(jugadores['Andrés Iniesta'],
                           jugadores['Carles Puyol'],{'num_pases':26})])
        G.add_edges_from([(jugadores['Iker Casillas'],
                           jugadores['Andrés Iniesta'],{'num_pases':12})])
```

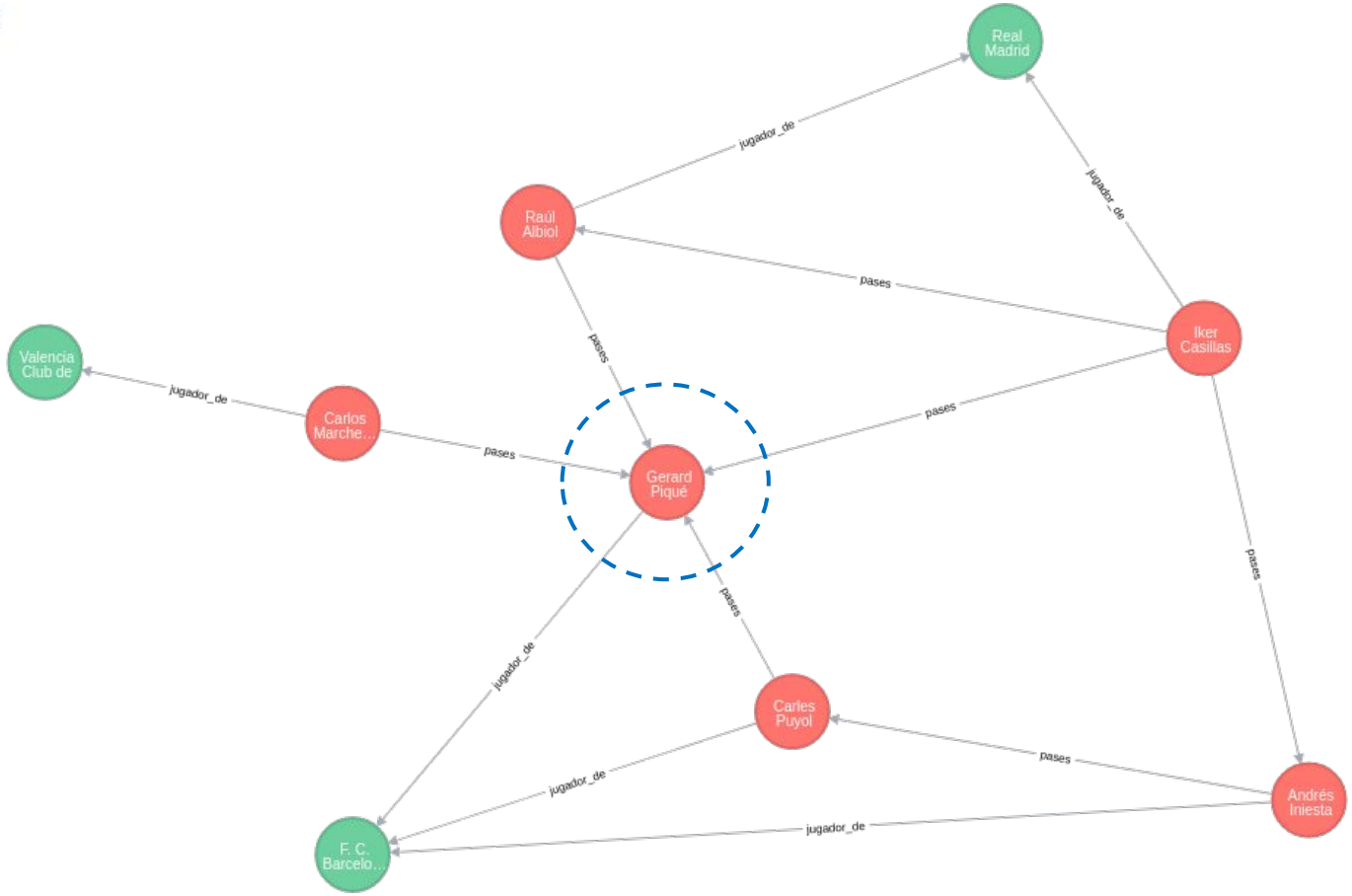
Grado de centralidad de cada uno de los nodos

```
In [7]: degree = nx.degree_centrality(G)
        degree = {jugadores[x]:degree[x] for x in degree}
        degree

Out[7]: {'Andrés Iniesta': 0.4,
         'Carles Puyol': 0.4,
         'Carlos Marchena': 0.2,
         'Gerard Piqué': 0.8,
         'Iker Casillas': 0.6000000000000001,
         'Raúl Albiol': 0.4}
```



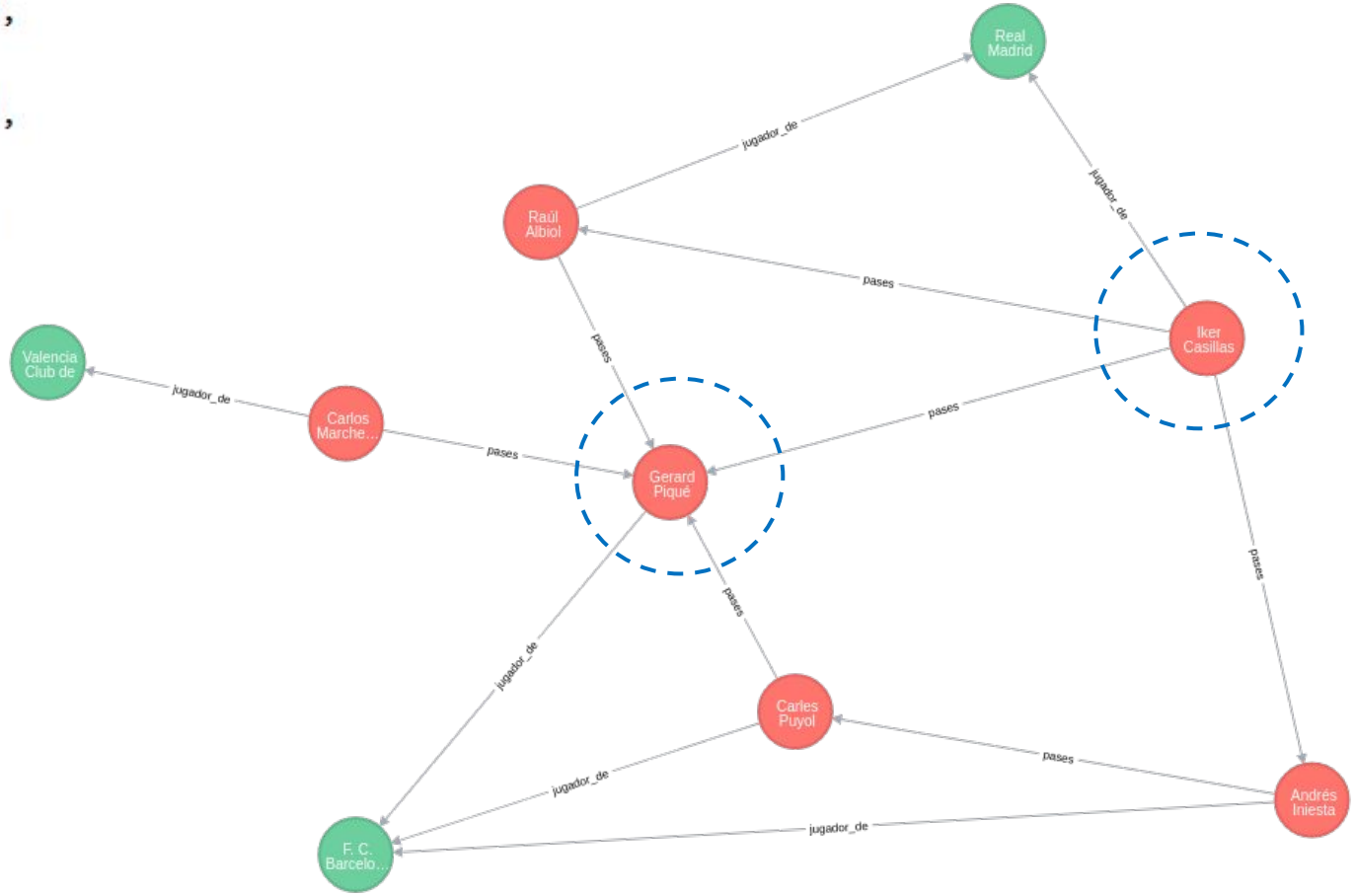
Centralidad de cercanía de cada uno de los nodos

```
In [8]: closeness = nx.closeness_centrality(G)
        closeness = {jugadores[x]:closeness[x] for x in closeness}
        closeness

Out[8]: {'Andrés Iniesta': 0.5555555555555556,
         'Carles Puyol': 0.625,
         'Carlos Marchena': 0.5,
         'Gerard Piqué': 0.8333333333333334,
         'Iker Casillas': 0.7142857142857143,
         'Raúl Albiol': 0.625}
```



Centralidad de cercanía con PESO de cada uno de los nodos

```
In [9]: closeness = nx.closeness_centrality(G, distance='num_pases')
        closeness = {jugadores[x]:closeness[x] for x in closeness}
        closeness

Out[9]: {'Andrés Iniesta': 0.065789473684210 52,
         'Carles Puyol': 0.17857142857142858,
         'Carlos Marchena': 0.1388888888888889,
         'Gerard Piqué': 0.20833333333333334,
         'Iker Casillas': 0.17857142857142858,
         'Raúl Albiol': 0.125}
```



¿Preguntas?

Gracias

angreyes@uan.edu.co

angiereyes.bet@gmail.com

angiereyesbet