Tema 2 – Ejercicio Neo4j

Marta Zorrilla – Diego García Saiz

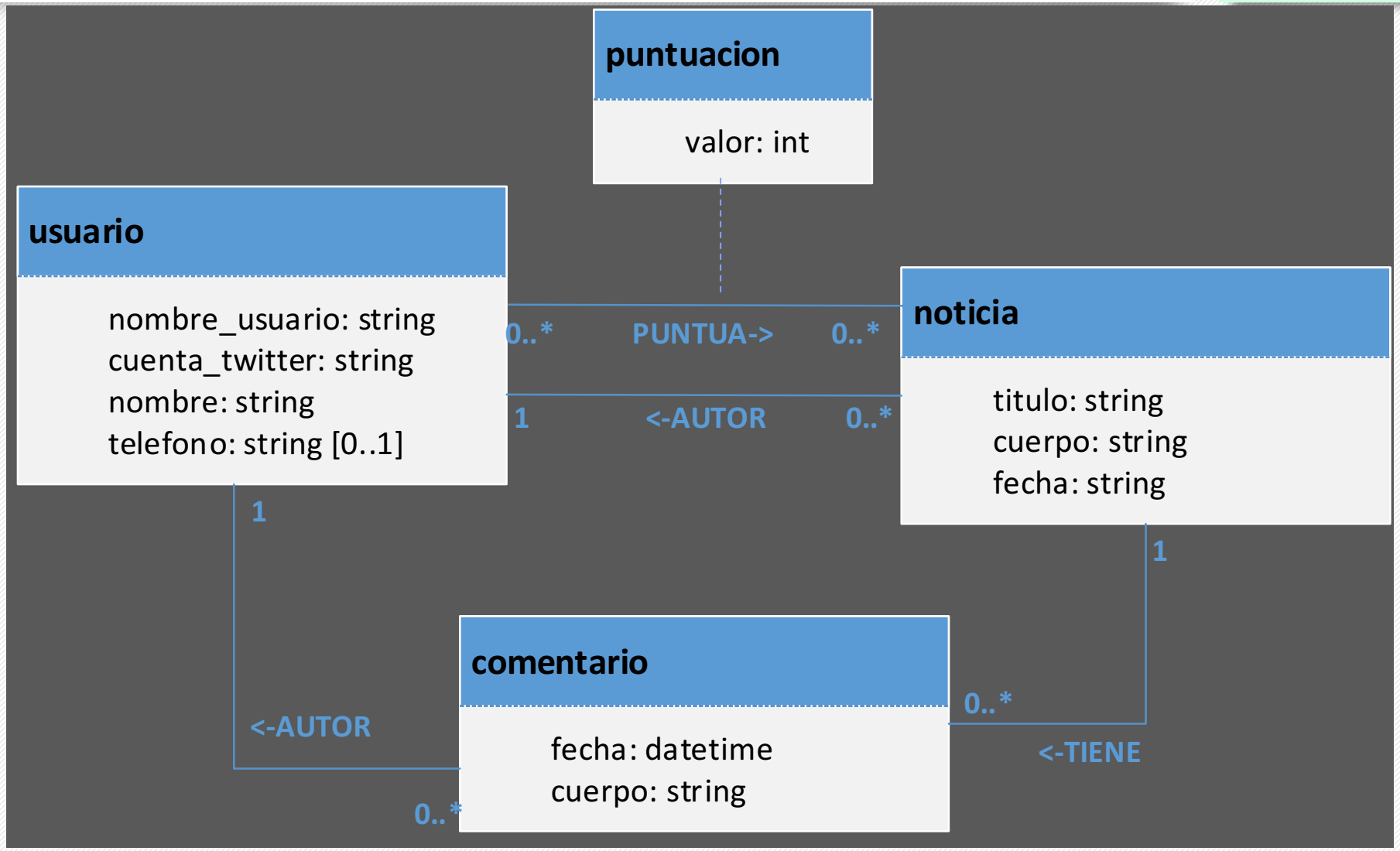
Enero 2017

Enunciado

- Se precisa diseñar un blog de noticias donde los usuarios registrados pueda publicar sus comentarios:
  - Cada autor tiene un nombre, un nombre de usuario único) y una cuenta de Twitter (única). Además, de forma opcional, los usuarios pueden tener un teléfono de contacto.
  - Las noticias tienen un título, un cuerpo y una fecha de publicación. Son publicadas por un autor. Además, los usuarios pueden puntuar las noticias publicadas por otros usuarios.
  - Las noticias reciben comentarios, quedando registrado la persona que lo escribió, el comentario escrito y el momento en el que lo hizo.

Diseño UML

M.Zorrilla - D.García



Diseño en Neo4j

- Recordemos que en un diagrama de bases de datos de Neo4j existen dos elementos: nodos y relaciones.

- Además, ambos elementos pueden tener propiedades (equivalentes a las columnas en una tabla del modelo relacional).

- Por otro lado, los nodos pueden tener etiquetas que los denotan como parte de una misma entidad o concepto.

Diseño en Neo4j: primeras decisiones

- Primera cuestión: ¿Qué etiquetas definir?.
  - Del enunciado y de su correspondiente diagrama UML, se observa la existencia de 3 entidades diferentes. En esta práctica, vamos a crear, por tanto, 3 etiquetas, una por cada entidad: **usuario, noticia y comentario.**
- Segundo cuestión: ¿Qué nodos formarán parte del diagrama?.
  - En este caso, cada nodo que se incluya en el diagrama representará a los usuarios, noticias o comentarios junto con sus propiedades. Cada nodo, por tanto, deberá ser etiquetado de forma pertinente.

Diseño en Neo4j: creación de nodos

M.Zorrilla - D.García

- Vamos a comenzar introduciendo nodos de usuario y noticias con los siguientes datos:
  - Usuarios:

```
CREATE(MiguelNodo:usuario {nombre: 'Miguel', nombreUsuario: 'Miguel_u', cuenta_twitter: 'miguelete93'})
```

```
CREATE(LaroNodo:usuario {nombre: 'Laro', nombreUsuario: 'Laro_u', cuenta_twitter: 'larocantabro85',
telefono: '1234567890'})
```

  - Noticias: dado que en CYPHER no existe el tipo de dato "fecha", almacenaremos en propiedades separadas el año, el mes y el día:

```
CREATE(NoticiaLaro1:noticia {titulo: 'Noticia1Laro', cuerpo: 'lorem ipsum…', dia: 22, mes: 5, anio: 2017})
```

```
CREATE(NoticiaLaro2:noticia {titulo: 'Noticia2Laro', cuerpo: 'lorem ipsum…2', dia: 2, mes: 3, anio: 2015})
```

```
CREATE(NoticiaMiguel:noticia {titulo: 'NoticiaMiguel', cuerpo: 'lorem ipsum…3', dia: 15, mes: 3, anio: 2017})
```

Diseño en Neo4j: restricciones

- Tercera cuestión: ¿hay que definir restricciones?.
  - En el enunciado, se indica que los nombres de usuario y las cuentas de twitter de los usuarios son únicas.
  - En este punto, cabrían dos posibilidades: definir dos restricciones de unicidad, o definir esta restricción sobre uno de los campos y hacer del otro la *KEY* de los nodos con la etiqueta "usuario".
  - En esta práctica, vamos a optar por esta segunda opción, siendo la *KEY* el nombre de usuario:

```
CREATE CONSTRAINT ON (u:usuario) ASSERT u.cuenta_twitter IS UNIQUE
```

```
CREATE CONSTRAINT ON (u:usuario) ASSERT u.nombreUsuario IS NODE KEY
```

NOTA: Si usas le versión Community, en la que no está disponible la restricción NODE KEY, crea dos restricciones UNIQUE.

Diseño en Neo4j: restricciones

M.Zorrilla - D.García

- Aunque el enunciado no lo diga explícitamente, podría haber otras restricciones necesarias.
  - En el caso de los usuarios y las noticias, tendría sentido obligar a que el nombre de usuario exista obligatoriamente (si no le hemos podido definir como *NODE KEY*).
  - Igualmente, en las noticias los campos título y cuerpo deberían de ser obligatorios:

```
CREATE CONSTRAINT ON (u:usuario) ASSERT exists(u.nombreUsuario)
```

```
CREATE CONSTRAINT ON (n:noticia) ASSERT exists(u.titulo)
```

```
CREATE CONSTRAINT ON (n:noticia) ASSERT exists(u.cuerpo)
```

NOTA: Las restricciones exists no están disponibles en la versión Community

Diseño en Neo4j: esquema inicial

- Actualmente, el esquema que tendríamos en la base de datos es el siguiente:

nombre: 'Laro'
nombreUsuario: 'Laro_u'
cuenta_twitter: 'larocantabro85'
telefono: '1234567890'

nombre: 'Miguel'
nombreUsuario: 'Miguel_u'
cuenta_twitter: 'miguelete93'

titulo: 'Noticia1Laro'
cuerpo: 'lorem ipsum …'
dia: 22
mes: 5
anio: 2017

titulo: 'Noticia2Laro'
cuerpo: 'lorem ipsum …2'
dia: 2
mes: 3
anio: 2017

titulo: 'MiguelNoticia'
cuerpo: 'lorem ipsum …3'
dia: 15
mes:3
anio: 2017

Diseño en Neo4j: consultas

- Hagamos las siguientes consultas:
  - Devolver todos los nodos.
  - Devolver todos los nodos con etiqueta “usuario”.
  - Devolver las noticias cuyo título termine en “o”.
  - Devolver los usuarios en los que exista el campo “telefono”.
  - Devolver los usuarios que, o bien tengan número de teléfono o bien, se llamen Miguel.
  - Devolver las noticias publicadas en 2017.
  - Devolver las noticias publicadas en marzo o abril.
  - Devolver las noticias publicadas entre los años 2014 y 2016.

Diseño en Neo4j: consultas

- Hagamos las siguientes consultas:
  - Devolver todos los nodos.

```
MATCH (allNodes) RETURN allNodes
```

  - Devolver todos los nodos con etiqueta “usuario”.

```
MATCH (u:usuario) RETURN u
```

  - Devolver las noticias cuyo título termine en “o”.

```
MATCH (n:noticia) WHERE n.titulo ENDS WITH ‘o’ RETURN n
```

  - Devolver los usuarios en los que exista el campo “telefono”.

```
MATCH (u:usuario) WHERE exists(u.telefono) RETURN u
```

  - Devolver los usuarios que o bien tengan número de teléfono o bien se llamen Miguel.

```
MATCH (u:usuario) WHERE exists(u.telefono) OR u.nombre=‘Miguel’ RETURN u
```

Diseño en Neo4j: consultas

- Hagamos las siguientes consultas:
  - Devolver las noticias publicadas en 2017.

```
MATCH (n:noticia) WHERE n.anio=2017 RETURN n
```

  - Devolver las noticias publicadas en marzo o abril.

```
MATCH (n:noticia) WHERE n.mes=3 OR n.mes=4 RETURN n
```

```
MATCH (n:noticia) WHERE n.mes IN [3,4] RETURN n
```

  - Devolver las noticias publicadas entre los años 2014 y 2016.

```
MATCH (n:noticia) WHERE n.anio>=2014 AND n.anio<=2016 RETURN n
```

Diseño en Neo4j: relaciones

- Cuarta cuestión: ¿cuáles son las relaciones entre nodos y sus propiedades?
  - En este ejercicio, existiría 4 tipos de relaciones:
    - Entre las noticias y los usuarios: “usuario escribe noticia” y “usuario puntúa noticia”.
    - Entre los comentarios y los usuarios: “usuario escribe comentario”.
    - Entre los comentarios y las noticias “los comentarios de los usuarios son escritos para una noticia”.

- Siguiendo el esquema anterior, tendríamos que añadir, por tanto, las relaciones entre los dos usuarios y las tres noticias.
  - Además, cada relación de “usuario puntúa noticia” tendrá una propiedad que almacenará la puntuación que el usuario le otorgó a la noticia.

Diseño en Neo4j: relaciones

- Insertamos las relaciones en el esquema. Llamaremos “REDACTO” a la relación que indica qué usuario redactó la noticia.
- Dado que los nodos ya están creados, será necesario utilizar primero operaciones MATCH que retornen y guarden en variables los nodos sobre los que queremos crear la relación:

```
MATCH(u:usuario)  WHERE u.nombreUsuario = ‘Laro_u’
MATCH(n:noticia) WHERE   n.titulo = ‘Noticia1Laro’
CREATE (u)-[:REDACTO{}]->(n)
```

```
MATCH(u:usuario)  WHERE u.nombreUsuario = ‘Laro_u’
MATCH(n:noticia) WHERE   n.titulo = ‘Noticia2Laro’
CREATE (u)-[:REDACTO{}]->(n)
```

```
MATCH(u:usuario)  WHERE u.nombreUsuario = ‘Miguel_u’
MATCH(n:noticia) WHERE   n.titulo = ‘NoticiaMiguel’
CREATE (u)-[:REDACTO{}]->(n)
```

Diseño en Neo4j: relaciones

- Insertamos las relaciones en el esquema. Llamaremos "PUNTUA" a la relación que indica la puntuación que un usuario le otorgó a una noticia.

```
MATCH(u:usuario)  WHERE u.nombreUsuario = 'Laro_u'
MATCH(n:noticia) WHERE   n.titulo = 'NoticiaMiguel'
CREATE (u)-[:PUNTUA{puntuación: 3}]->(n)
```

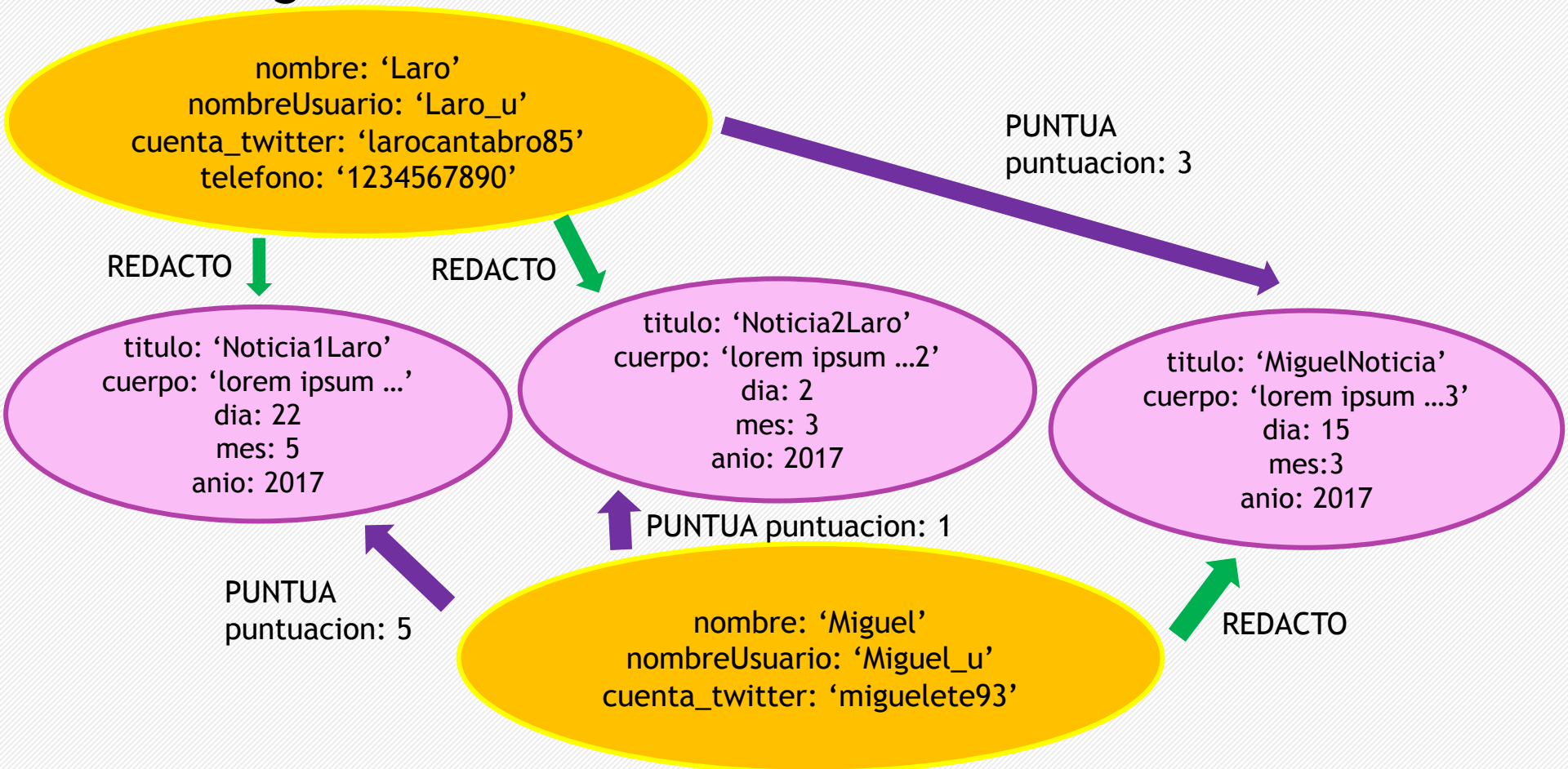
```
MATCH(u:usuario)  WHERE u.nombreUsuario = 'Miguel_u'
MATCH(n:noticia) WHERE   n.titulo = 'Noticia1Laro'
CREATE (u)-[:PUNTUA {puntuación: 5}]->( n)
```

```
MATCH(u:usuario)  WHERE u.nombreUsuario = 'Miguel_u'
MATCH(n:noticia) WHERE   n.titulo = 'Noticia2Laro'
CREATE (u)-[: PUNTUA {puntuación: 1}]->( n)
```

Diseño en Neo4j: esquema con relaciones

M.Zorrilla - D.García

- Ahora, el esquema que tendríamos en la base de datos es el siguiente:



Diseño en Neo4j: consultas

- Probemos a realizar las siguientes consultas:
  - Devolver los datos de todas las relaciones “PUNTUA” con puntuación igual a 4.
  - Devolver los datos de todas las relaciones “PUNTUA” hechas por Miguel.
  - Devolver los datos de las relaciones “REDACTO” de las noticias publicadas en 2017 cuyo autor se llame “Laro”.
  - Devolver las cuenta de twitter de todos los usuarios que hayan puntuado noticias de 2017.

Diseño en Neo4j: consultas

- Devolver los datos de todas las relaciones “PUNTUA” con puntuación de 4.

```
MATCH relation=(u:usuario)-[r:PUNTUA]->(n:noticia) WHERE n.puntuacion=4
RETURN relation
```

- Devolver los datos de todas las relaciones “PUNTUA” realizadas por Miguel.

```
MATCH relation=(u:usuario)-[r:PUNTUA]->(n:noticia) WHERE
u.nombre=‘Miguel’ RETURN relation
```

- Devolver los datos de las relaciones “REDACTO” de las noticias publicadas en 2017 cuyo autor se llame “Laro”.

```
MATCH relation=(u:usuario)-[r:REDACTO]->(n:noticia) WHERE r.anio=2017
AND u.nombre=‘Laro’ RETURN relation
```

- Devolver las cuentas de twitter de todos los usuarios que hayan valorado noticias de 2017.

```
MATCH relation=(u:usuario)-[r:PUNTUA]->(n:noticia) WHERE r.anio=2017
RETURN u.cuenta_twitter
```

Diseño en Neo4j: a realizar por el estudiante

- Siguiendo el diseño del ejemplo, se propone añadir al esquema los comentarios que los usuarios pueden realizar sobre las noticias. Vamos a considerar que las noticias y los comentarios, a parte de su etiqueta particular, tengan una etiqueta común que indique que estos nodos almacenan contenidos del blog. Realizar en base a ello las siguientes actividades:
  - Añadir a los nodos “noticia” una nueva etiqueta denominada “contenido”.
  - Crear nodos de comentarios con doble etiqueta (“comentario” y “contenido”).
  - Crear las restricciones que se consideren necesarias.
  - Crear las relaciones entre los comentarios y los usuarios. ¿Tienen estas relaciones alguna propiedad?.
  - Crear las relaciones entre los comentarios y las noticias. ¿Tienen estas relaciones alguna propiedad?.

Diseño en Neo4j: consultas sobre el nuevo diseño

- En base al nuevo diseño al que se han añadido los comentarios, realizar las siguientes consultas:
  - Devolver todos los nodos con etiqueta “contenido”.
  - Devolver todos los nodos etiquetados como “contenido” y “comentario”.
  - Contar el número de nodos que almacenan comentarios de los usuarios.
  - Devolver los comentarios realizados en 2017 sobre noticias publicadas en 2016.
  - Contar y devolver el número de contenidos publicados por el usuario llamado “Laro”.
  - Contar y devolver el número de comentarios que tiene cada noticia, junto al título de la misma.