Generación de datos

Creación de categorías

```
// 20 categorías tipo Pinterest WITH [ 'Art', 'Design', 'Technology',
'Food', 'Travel', 'Fitness', 'Fashion', 'DIY', 'Photography',
'Education', 'Gaming', 'Music', 'Movies', 'Quotes', 'Home Decor',
'Nature', 'Business', 'Sports', 'Memes' ] AS categories UNWIND range(0,
size(categories)-1) AS i CREATE (:Category { id_category: 'CAT-' +
toString(i+1), name: categories[i] });
```

Creación de tags

```
// 150 tags variados (se mezclan palabras reales + sufijo para que no
choquen) WITH [
'travel','food','design','tech','art','fitness','music','gaming','coding
','nature',
'pets','fashion','quotes','diy','coffee','books','photography','cars','s
ports','movies', 'series','anime','cyberpunk','rpg','open
world','scifi','romance','streetwear', 'architecture','data
science','ai','ml','python','javascript','cooking','mexico',
'japan','europe','usa','beach','mountains','city','startup','productivit
y','study',
'selfcare','motivation','memes','cats','dogs','luxury','minimalism','col
orful',
'black&white','abstract','landscape','portrait','interior','exterior','3
d','flat' ] AS baseTags UNWIND range(1,150) AS i WITH i,
baseTags[toInteger(rand()*size(baseTags))] AS base CREATE (:Tag {name:
base + '_' + toString(i)});
```

Creación de usuarios

Para simular a usuarios realistas, asignamos categorías de interés a la hora de crearlos. Usaremos esta para definir su tipo de comportamiento al crear el resto de componentes asociados a un usuario.

```
MATCH (u:User)
WITH collect(u) AS users
MATCH (c:Category)
WITH users, collect(c) AS cats

UNWIND users AS u
WITH u, apoc.coll.randomItems(cats, 3) AS interests
```

```
SET u._interests = [x IN interests | x.id_category];
```

Crear boards por usuario

Creamos boards por los intereses del usuario en particular, asociado a la categoría

```
MATCH (u:User)
UNWIND u._interests AS catId
MATCH (c:Category {id_category: catId})
CREATE (u)-[:CREATES]->(b:Board {
    id_board: 'UB-' + u.id_user + '-' + catId,
    title: 'Board de ' + c.name,
    description: 'Colección sobre ' + c.name,
    created_at: datetime()
})
MERGE (b)-[:IN_CATEGORY]->(c);
```

Crear cinco pins por board

Por cada board creamos cinco pins, de modo que cada usuario ha creado 15 pins.

```
MATCH (u:User)-[:Creates]->(b:Board)-[:IN_CATEGORY]->(c:Category)
UNWIND range(1,5) AS i
CREATE (p:Pin {
    id_pin: b.id_board + '-P' + toString(i),
    title: c.name + ' Pin ' + toString(i),
    description: 'Contenido sobre ' + c.name,
    url_image: 'https://picsum.photos/seed/' + b.id_board + '-' +
toString(i) + '/400/300',
    created_at: datetime()
})
MERGE (b)-[:CONTAINS]->(p);
MERGE (u)-[:CREATES]->(p)
```

Agrupar tags con categorías

Agrupamos los tags relevantes con sus categorías respectivas.

```
MATCH (t:Tag)
MATCH (c:Category)
WHERE toLower(t.name) CONTAINS toLower(c.name)
MERGE (t)-[:RELEVANT_TO]->(c);
```

Asignar tags a pins

Asignamos tags a los pins que están en determinadas categorías.

```
MATCH (p:Pin)-[:IN_CATEGORY]->(c:Category)
MATCH (t:Tag)-[:RELEVANT_TO]->(c)
WITH p, collect(t) AS relevants
WITH p, apoc.coll.randomItems(relevants, 3) AS pick
UNWIND pick AS t
MERGE (p)-[:HAS_TAG]->(t);
```

Simular a usuarios guardando pins de otras personas

Revolvemos los pins de tal modo que un Board X que comparta categoría con otro Board Y querrá tener parte de sus pins, o que tenga categorías relacionadas.

```
// Esto marca el 75% de usuarios para repins
MATCH (u:User)
WITH count(u) AS count_u, collect(u) AS all
WHERE count_u > 0
WITH all, toInteger(round(count_u * 0.75)) AS num_items
WITH apoc.coll.randomItems(all, num_items) AS selected
UNWIND selected AS u
SET u._selected_repin = true;

// Para cada board del usuario, recoge boards compatibles cuya categoría
sea la misma
// o esté conectada por RELATED_TO (distancia 0..1). Guarda sus id_board
en _compatibles.
MATCH (u:User {_selected_repin:
true})-[:CREATES]->(b:Board)-[:IN_CATEGORY]->(c:Category)
MATCH (c)-[:RELATED_TO*0..1]-(c2:Category)
MATCH (other:Board)-[:IN_CATEGORY]->(c2)
WHERE other <> b
WITH b, collect(DISTINCT other) AS compatibles
SET b._compatibles = [x IN compatibles | x.id_board];

// Selecciona tableros fuente priorizando boards con más pines (top 10)
y luego toma 3 pins aleatorios de esa fuente
MATCH (b:Board)
WHERE exists(b._compatibles)
```

```
UNWIND b._compatibles AS chosenId
MATCH (source:Board {id_board:chosenId})-[:CONTAINS]->(p:Pin)
WITH b, source, collect(p) AS pins, size(collect(p)) AS pinCount
ORDER BY pinCount DESC
WITH b, collect(source.id_board)[0..10] AS topBoardIds   // top 10 por
cantidad de pines
WITH b, apoc.coll.randomItem(topBoardIds) AS chosenId
MATCH (source:Board {id_board:chosenId})-[:CONTAINS]->(p2:Pin)
WITH b, collect(p2) AS pinsFromSource
WITH b, apoc.coll.randomItems(pinsFromSource, 3) AS picks
UNWIND picks AS pPick
MERGE (b)-[:CONTAINS]->(pPick);

// Limpiar marcas temporales
MATCH (u:User) REMOVE u._selected_repin;
MATCH (b:Board) REMOVE b._compatibles;
```

Hacer que personas sigan a otras personas

```
MATCH (u:User)
WITH count(u) AS count_u, collect(u) AS all
WHERE count_u > 0
WITH all, toInteger(round(count_u * 0.75)) AS num_items
WITH apoc.coll.randomItems(all, num_items) AS selected
UNWIND selected AS u
SET u._selected_follow = true;

MATCH (u:User {_selected_follow: true})
WITH u
MATCH (target:User)
WHERE target <> u
WITH u, collect(target) AS options
WITH u, apoc.coll.randomItem(options) AS chosen
MERGE (u)-[:FOLLOWS {since: datetime()}]->(chosen);

MATCH (u:User {_selected_follow: true})
WITH u
MATCH (target:User)
WHERE target <> u
WITH u, collect(target) AS opts
WITH u, apoc.coll.randomItem(opts + opts + opts) AS chosen
MERGE (u)-[:FOLLOWS {since: datetime()}]->(chosen);

MATCH (u:User) REMOVE u._selected_follow;
```

Asignar imágenes relevantes para los pins

Quitamos los placeholders
Obtuvimos las imágenes desde [este colab.]

```
CALL
apoc.load.json("https://raw.githubusercontent.com/ariastroke2/mineria/ma
ster/backend/imageindex.json") YIELD value
SET global._imgmap = value;

MATCH (p:Pin)<-[:CONTAINS]-(b:Board)-[:IN_CATEGORY]->(c:Category),
      (global)
WITH p, c.name AS catName, global._imgmap AS imgmap
WHERE imgmap[catName] IS NOT NULL AND size(imgmap[catName]) > 0
SET p.url_image = apoc.coll.randomItem(imgmap[catName]);
```

Crear comentarios

Primero creamos una lista de comentarios que podrían ser publicados

```
// Guardar temporalmente los textos de comentarios
MERGE (tmp:TempComments)
SET tmp.texts = ["Great!", "Yeah!", "Awesome!", "Not really feelin'
it"];
```

Crear comentarios

```
// 1. Cargar la lista de textos (solo 1 fila)
MATCH (tmp:TempComments)
WITH tmp.texts AS texts, range(1, 300) AS user_ids_range

// 2. Emparejar TODOS los Pines
MATCH (p:Pin)

// 3. Aplicar la probabilidad de 0.15 para la creación del comentario.
// Para no saturar.
WHERE rand() < 0.15

// 4. Seleccionar un único ID de usuario aleatorio para este Pin.
// Usamos apoc.coll.randomItem en user_ids_range para obtener solo 1 ID.
WITH p, apoc.coll.randomItem(user_ids_range) AS random_user_num, texts

// 5. Construir el ID del usuario y seleccionar un texto aleatorio.
```

```
WITH p, 'USER-' + toString(random_user_num) AS random_user_id,
apoc.coll.randomItem(texts) AS msg

// 6. Encontrar al usuario. (Si no existe, la sentencia se detiene aquí
para este Pin/ID).
MATCH (u:User)
WHERE u.id_user = random_user_id
// Ya no necesitamos LIMIT 1, ya que solo generamos 1 random_user_id.

// 7. Crear el comentario y las relaciones
CREATE (c:Comment {
    id_comment: 'CMT-' + randomUUID(),
    text: msg,
    created_at: datetime() - duration({days: toInteger(rand()*60)})
})
MERGE (u)-[:WROTE]->(c)
MERGE (c)-[:ON]->(p)

RETURN count(c) AS Total_Comentarios_Creados;
```

Añadir likes

Asignamos likes a los pins

```
// Likes de usuarios a pins, coherentes y distribuidos
MATCH (u:User)
MATCH (p:Pin)
WITH u, p, rand() AS r
WHERE r < 0.20 // ~20% de densidad.
MERGE (u)-[:LIKES {
date: datetime() - duration({days: toInteger(rand()*30)})
}]->(p);
```