CONSULTAS CYPHER - BACKEND PINTEREST CLONE

1. OBTENER TODOS LOS USUARIOS

Endpoint: GET /api/users
Descripción: Lista todos los usuarios para el selector de usuarios

```
MATCH (u:User)
RETURN u.id_user AS id, u.name AS name, u.profile_picture AS
profile_picture
ORDER BY u.name
```

2. OBTENER PINES (Feed Principal)

Endpoint: GET /api/pins
Descripción: Obtiene todos los pines con información del creador, likes y estado de follow

```
MATCH (p:Pin)
OPTIONAL MATCH (u:User)-[:CREATES]->(p)
OPTIONAL MATCH (:User)-[l:LIKES]->(p)
OPTIONAL MATCH (me:User {id_user: $userId})-[myLike:LIKES]->(p)
OPTIONAL MATCH (me)-[followRel:FOLLOWS]->(u)

WITH DISTINCT p, u, count(DISTINCT l) AS likesCount,
     (myLike IS NOT NULL) AS likedByMe,
     (followRel IS NOT NULL) AS isFollowing

RETURN p, u.name AS creator, u.id_user AS creatorId,
u.profile_picture AS creatorPic,
       likesCount, likedByMe, isFollowing, p.created_at AS createdAt
ORDER BY p.created_at DESC
LIMIT 150
```

3. OBTENER PINES TRENDING (Ordenados por likes)

Endpoint: GET /api/pins/trending
Descripción: Pines más populares ordenados por cantidad de likes

```
MATCH (p:Pin)
OPTIONAL MATCH (u:User)-[:CREATES]->(p)
OPTIONAL MATCH (:User)-[l:LIKES]->(p)
OPTIONAL MATCH (me:User {id_user: $userId})-[myLike:LIKES]->(p)

WITH p, u, count(l) AS likesCount, myLike
WHERE likesCount > 0

RETURN p, u.name AS creator, u.id_user AS creatorId,
       likesCount, (myLike IS NOT NULL) AS likedByMe
ORDER BY likesCount DESC
LIMIT 50
```

4. OBTENER PIN POR ID (Detalle de un pin)

Endpoint: GET /api/pin/:id
Descripción: Obtiene los datos completos de un pin incluyendo comentarios

```
MATCH (p:Pin {id_pin: $pinId})
OPTIONAL MATCH (u:User)-[:CREATES]->(p)
OPTIONAL MATCH (b:Board)-[:CONTAINS]->(p)
OPTIONAL MATCH (:User)-[l:LIKES]->(p)
OPTIONAL MATCH (me:User {id_user: $userId})-[myLike:LIKES]->(p)
OPTIONAL MATCH (me)-[followRel:FOLLOWS]->(u)

// Comentarios
OPTIONAL MATCH (author:User)-[:WROTE]->(c:Comment)-[:ON]->(p)

WITH p, u, b, count(DISTINCT l) AS likesCount, myLike, followRel, c,
author
ORDER BY c.created_at DESC
```

```
WITH p, u, b, likesCount, myLike, followRel,
     collect(CASE WHEN c IS NOT NULL THEN {
        id: c.id_comment,
        text: c.body,
        author: author.name,
        authorPic: author.profile_picture,
     Date: c.created_at
     } ELSE null END) AS comments

RETURN p,
       u.name AS creator, u.id_user AS creatorId, u.profile_picture
AS creatorPic,
       b.title AS board, b.id_board AS boardId,
       likesCount,
       (myLike IS NOT NULL) AS likedByMe,
       (followRel IS NOT NULL) AS isFollowing,
       comments
```

5. SUGERENCIAS DE PINES SIMILARES

Endpoint: GET /api/pin/:id (segunda consulta)
Descripción: Encuentra pines similares basados en board, tags y creador

```
MATCH (main:Pin {id_pin: $pinId})

// A. Mismo Board
OPTIONAL MATCH
(main)<-[:CONTAINS]-(b:Board)-[:CONTAINS]->(pBoard:Pin)
WHERE pBoard <> main

// B. Mismos Tags (Si tienes tags implementados)
OPTIONAL MATCH (main)-[:HAS_TAG]->(t:Tag)<-[:HAS_TAG]-(pTag:Pin)
WHERE pTag <> main

// C. Mismo Creador
OPTIONAL MATCH (main)<-[:CREATES]-(u:User)-[:CREATES]->(pUser:Pin)
WHERE pUser <> main
```

```
// Recolectar todo y unificar
WITH collect(DISTINCT pBoard) AS boardPins,
     collect(DISTINCT pTag) AS tagPins,
     collect(DISTINCT pUser) AS userPins

// Unimos las listas dando prioridad implícita por orden
WITH boardPins + tagPins + userPins AS allSuggestions
UNWIND allSuggestions AS p

// Eliminamos duplicados finales y traemos datos básicos
WITH DISTINCT p
OPTIONAL MATCH (creator:User)-[:CREATES]->(p)

RETURN p.id_pin AS id,
       p.title AS title,
       p.url_image AS url_image,
       creator.name AS creator
LIMIT 20
```

6. LIKE/UNLIKE PIN (Toggle)

Endpoint: POST /api/pins/:id/like
Descripción: Verifica, crea o elimina relación LIKES

```
// Verificar si ya existe el like
MATCH (u:User {id_user: $userId})-[r:LIKES]->(p:Pin {id_pin:
$pinId})
RETURN r

// Eliminar like (si existe)
MATCH (u:User {id_user: $userId})-[r:LIKES]->(p:Pin {id_pin:
$pinId})
DELETE r

// Crear like (si no existe)
MATCH (u:User {id_user: $userId}), (p:Pin {id_pin: $pinId})
MERGE (u)-[:LIKES {date: datetime()}]->(p)
```

```
// Contar total de likes
MATCH (:User)-[l:LIKES]->(p:Pin {id_pin: $pinId})
RETURN count(l) AS likesCount
```

7. CREAR COMENTARIO

Endpoint: POST /api/pins/:id/comment
Descripción: Crea un nuevo comentario en un pin

```
MATCH (u:User {id_user: $userId})
MATCH (p:Pin {id_pin: $pinId})
CREATE (c:Comment {id_comment: $commentId, body: $text, created_at:
datetime()})
CREATE (u)-[:WROTE]->(c)-[:ON]->(p)
```

8. FOLLOW/UNFOLLOW USER (Toggle)

Endpoint: POST /api/users/:id/follow
Descripción: Verifica, crea o elimina relación FOLLOWS

```
// Verificar si ya existe el follow
MATCH (follower:User {id_user: $userId})-[r:FOLLOWS]->(target:User
{id_user: $targetUserId})
RETURN r

// Eliminar follow (si existe)
MATCH (follower:User {id_user: $userId})-[r:FOLLOWS]->(target:User
{id_user: $targetUserId})
DELETE r

// Crear follow (si no existe)
MATCH (follower:User {id_user: $userId}), (target:User {id_user:
$targetUserId})
MERGE (follower)-[:FOLLOWS {since: datetime()}]->(target)
```

```
// Contar total de seguidores
MATCH (target:User {id_user:
$targetUserId})<-[:FOLLOWS]-(other:User)
RETURN count(other) AS followersCount
```

9. OBTENER USUARIOS QUE SIGUE (Following)

Endpoint: GET /api/users/:id/following
Descripción: Lista de usuarios que un usuario sigue

```
MATCH (u:User {id_user: $userId})-[:FOLLOWS]->(other:User)
RETURN other
ORDER BY other.name
```

10. OBTENER SEGUIDORES (Followers)

Endpoint: GET /api/users/:id/followers
Descripción: Lista de seguidores de un usuario

```
MATCH (u:User {id_user: $userId})<-[:FOLLOWS]-(other:User)
RETURN other
ORDER BY other.name
```

11. OBTENER TODOS LOS BOARDS

Endpoint: GET /api/boards
Descripción: Lista todos los boards del sistema

```
MATCH (b:Board)
RETURN b.id_board AS id, b.title AS title
```

```
ORDER BY b.created_at DESC
```

12. OBTENER BOARDS DE UN USUARIO

Endpoint: GET /api/:user/boards
Descripción: Boards creados por un usuario con preview de imágenes

```
MATCH (u:User {id_user: $userId})-[:CREATES]->(b:Board)
OPTIONAL MATCH (b)-[:CONTAINS]->(p:Pin)
WITH b, p ORDER BY p.created_at DESC
WITH b, collect(p.url_image)[0..3] AS imgs
RETURN b.id_board AS id, b.title AS title, b.description AS
description, imgs AS images
ORDER BY b.created_at DESC
```

13. OBTENER DETALLE DE UN BOARD

Endpoint: GET /api/boards/:boardId
Descripción: Información del board con todos sus pines

```
MATCH (b:Board {id_board: $boardId})
OPTIONAL MATCH (b)-[:CONTAINS]->(p:Pin)
OPTIONAL MATCH (creator:User)-[:CREATES]->(p)
RETURN b.title AS title, b.description AS description,
       collect({
           id_pin: p.id_pin,
           title: p.title,
           url_image: p.url_image,
           creator: creator.name
       }) AS pins
```

14. AGREGAR PIN A UN BOARD

Endpoint: POST /api/boards/:boardId/add-pin
Descripción: Crea relación CONTAINS entre board y pin

```
MATCH (b:Board {id_board: $boardId})
MATCH (p:Pin {id_pin: $pinId})
MERGE (b)-[:CONTAINS]->(p)
RETURN p.id_pin AS addedPin
```

15. CREAR BOARD

Endpoint: POST /api/boards
Descripción: Crea un nuevo board asociado a un usuario

```
MATCH (u:User {id_user: $userId})
CREATE (b:Board {id_board: $newId, title: $title, description:
$description, created_at: datetime()})
CREATE (u)-[:CREATES]->(b)
```

16. CREAR PIN

Endpoint: POST /api/pins
Descripción: Crea un nuevo pin asociado a usuario y board

```
MATCH (u:User {id_user: $userId})
MATCH (b:Board {id_board: $boardId})
CREATE (p:Pin {
    id_pin: $newId,
    title: $title,
 Description: $description,
    url_image: $url_image,
    created_at: datetime()
})
```

```
CREATE (u)-[:CREATES]->(p)
CREATE (b)-[:CONTAINS]->(p)
```

17. OBTENER PERFIL DE USUARIO

Endpoint: GET /api/user/:id
Descripción: Datos del usuario con conteos de boards, pins y likes

```
MATCH (u:User {id_user: $userId})
OPTIONAL MATCH (u)-[:CREATES]->(b:Board)
OPTIONAL MATCH (u)-[:CREATES]->(p:Pin)
OPTIONAL MATCH (u)-[:LIKES]->(liked:Pin)
RETURN u, count(DISTINCT b) AS boardsCount, count(DISTINCT p) AS
pinsCount, count(DISTINCT liked) AS likesCount
```

18. OBTENER PINES GUARDADOS POR USUARIO

Endpoint: GET /api/user/:id/saved-pins
Descripción: Pines que están en los boards del usuario

```
MATCH (u:User {id_user:
$userId})-[:CREATES]->(b:Board)-[:CONTAINS]->(p:Pin)
OPTIONAL MATCH (creator:User)-[:CREATES]->(p)
RETURN DISTINCT p, creator.name AS creatorName, b.title AS
boardTitle
ORDER BY p.created_at DESC
```

19. OBTENER PINES CON LIKE DEL USUARIO

Endpoint: GET /api/user/:id/liked-pins
Descripción: Pines a los que el usuario ha dado like

```
MATCH (u:User {id_user: $userId})-[l:LIKES]->(p:Pin)
OPTIONAL MATCH (creator:User)-[:CREATES]->(p)
RETURN p, creator.name AS creatorName, l.date AS likedAt
ORDER BY l.date DESC
```

20. BÚSQUEDA DE PINES

Endpoint: GET /api/search?type=pins
Descripción: Busca pines por título, descripción o tags

```
MATCH (p:Pin)
OPTIONAL MATCH (p)-[:HAS_TAG]->(t:Tag)
WITH p, collect(DISTINCT t.name) AS tags
OPTIONAL MATCH (u:User)-[:CREATES]->(p)
WITH p, u, tags
WHERE
    (p.title IS NOT NULL AND toLower(p.title) CONTAINS toLower($q))
    OR
    (p.description IS NOT NULL AND toLower(p.description) CONTAINS
toLower($q))
    OR
    any(tag IN tags WHERE tag IS NOT NULL AND toLower(tag) CONTAINS
toLower($q))
RETURN p, u.name AS creator, u.profile_picture AS creatorPic, tags
ORDER BY p.created_at DESC
LIMIT 30
```

21. BÚSQUEDA DE USUARIOS

Endpoint: GET /api/search?type=users
Descripción: Busca usuarios por nombre o username

```
MATCH (u:User)
WHERE toLower(u.name) CONTAINS toLower($q)
```

```
   OR (u.username IS NOT NULL AND toLower(u.username) CONTAINS
toLower($q))
RETURN u
LIMIT 30
```

22. BÚSQUEDA DE BOARDS

Endpoint: GET /api/search?type=boards
Descripción: Busca boards por título o descripción

```
MATCH (b:Board)
WHERE toLower(b.title) CONTAINS toLower($q)
   OR toLower(b.description) CONTAINS toLower($q)
RETURN b
ORDER BY b.created_at DESC
LIMIT 30
```

23. SIMILITUD JACCARD DE USUARIOS POR LIKES

Endpoint: GET /api/users/:id/similar
Descripción: Encuentra usuarios similares basados en pins que ambos han dado like

```
MATCH (me:User {id_user:
$userId})-[:LIKES]->(p:Pin)<-[:LIKES]-(other:User)
WHERE other <> me
WITH me, other, count(DISTINCT p) AS commonLikes

// likes totales de me
MATCH (me)-[:LIKES]->(pMe:Pin)
WITH me, other, commonLikes, count(DISTINCT pMe) AS myLikes

// likes totales del otro
MATCH (other)-[:LIKES]->(pOther:Pin)
```

```
WITH other, commonLikes, myLikes, count(DISTINCT pOther) AS
otherLikes

WITH other, commonLikes, myLikes, otherLikes,
     (1.0 * commonLikes) / (myLikes + otherLikes - commonLikes) AS
jaccard
RETURN other, commonLikes, jaccard
ORDER BY jaccard DESC, commonLikes DESC
LIMIT 10
```

24. TRENDING PINS (Últimos 7 días)

Endpoint: GET /api/trending/pins
Descripción: Pines con más likes en la última semana

```
MATCH (p:Pin)
OPTIONAL MATCH (:User)-[l:LIKES]->(p)
WHERE l.date >= datetime() - duration('P7D') OR l IS NULL
WITH p, count(l) AS likesLast7d
RETURN p, likesLast7d
ORDER BY likesLast7d DESC, p.created_at DESC
LIMIT 20
```

25. TRENDING TAGS (Últimos 7 días)

Endpoint: GET /api/trending/tags
Descripción: Tags más populares basados en likes de la última semana

```
MATCH (:User)-[l:LIKES]->(p:Pin)-[:HAS_TAG]->(t:Tag)
WHERE l.date >= datetime() - duration('P7D')
RETURN t.name AS tag, count(*) AS likesCount
ORDER BY likesCount DESC
LIMIT 20
```

26. CENTRALIDAD DE USUARIOS (Degree Centrality)

Endpoint: GET /api/analytics/centrality/users
Descripción: Usuarios con mayor centralidad basada en follows

```
MATCH (u:User)
OPTIONAL MATCH (u)<-[:FOLLOWS]-(f:User)
WITH u, count(f) AS followers
OPTIONAL MATCH (u)-[:FOLLOWS]->(fo:User)
WITH u, followers, count(fo) AS following
RETURN u, followers, following, (followers + following) AS degree
ORDER BY degree DESC
LIMIT 50
```

27. COMUNIDADES POR TAGS

Endpoint: GET /api/analytics/communities/tags
Descripción: Agrupa usuarios por los tags que les gustan

```
MATCH (u:User)-[:LIKES]->(p:Pin)-[:HAS_TAG]->(t:Tag)
WITH t, collect(DISTINCT u) AS users
RETURN t.name AS tag,
       [u IN users | {
            id: u.id_user,
            name: u.name,
            profile_picture: u.profile_picture
       }] AS members,
       size(users) AS size
ORDER BY size DESC
LIMIT 10
```

28. SIMILITUD DE PINES POR TAGS (Jaccard)

Endpoint: GET /api/pins/:id/similar
Descripción: Encuentra pines similares basados en tags comunes

```
MATCH (p:Pin {id_pin:
$pinId})-[:HAS_TAG]->(t:Tag)<-[:HAS_TAG]-(other:Pin)
WHERE other <> p
WITH other, count(DISTINCT t) AS commonTags
RETURN other, commonTags
ORDER BY commonTags DESC, other.created_at DESC
LIMIT 10
```