**ĐẶC TẢ NGHIỆP VỤ TÌM KIẾM BẠN BÈ VÀ BÀI VIẾT**

**1. TỔNG QUAN HỆ THỐNG**

**1.1 Mục tiêu**

Xây dựng hệ thống tìm kiếm thông minh cho mạng xã hội trường học, hỗ trợ tìm kiếm bạn bè và bài viết với khả năng xử lý 5000 người dùng đồng thời.

**1.2 Công nghệ sử dụng**

* Elasticsearch: Công cụ tìm kiếm chính
* PostgreSQL: Cơ sở dữ liệu gốc
* Redis: Cache và xử lý job queue

**2. NGHIỆP VỤ TÌM KIẾM BẠN BÈ**

**2.1 Tìm kiếm Multi-Match**

**Mô tả nghiệp vụ:**

* Đầu vào: Người dùng nhập "Nguyễn Văn An" vào ô tìm kiếm
* Xử lý: Tìm kiếm đồng thời trong nhiều trường thông tin
* Đầu ra: Danh sách bạn bè được xếp hạng theo độ phù hợp

**Chi tiết xử lý:**

|  |  |  |
| --- | --- | --- |
| **Trường tìm kiếm** | **Boost** | **Ví dụ** |
| Họ tên | x3 | "Nguyễn Văn An" |
| Username | x2 | "nguyenvanan123" |
| Nickname | x1 | "An Nguyễn" |
| Bio/Mô tả | x1 | "Tôi là An từ lớp CNTT1" |

**Tính năng đặc biệt:**

* Fuzziness: "Nguyen Van An" → tìm thấy "Nguyễn Văn An"
* Stemming: "Anh" → tìm cả "An", "Anh An"
* Stop words: Bỏ qua "của", "là", "từ"

**2.2 Lọc dữ liệu (Bool Query)**

**Cấu trúc lọc:**

**Must (Bắt buộc)**

* Chứa từ khóa tìm kiếm
* Tài khoản đã kích hoạt
* Không bị khóa/cấm

**Filter (Lọc nhanh)**

* Giới tính: Nam, Nữ, Khác
* Khoa: CNTT, Kinh tế, Ngoại ngữ
* Năm học: K18, K19, K20, K21
* Tỉnh thành: Hà Nội, TP.HCM, Đà Nẵng
* Trạng thái: Online, Offline, Busy

**Must\_not (Loại trừ)**

* Đã chặn lẫn nhau
* Tài khoản bị báo cáo
* Tài khoản giả/spam

**2.3 Gợi ý tự động (Auto-complete)**

**Luồng hoạt động:**

1. Người dùng gõ "N" → Không gợi ý (quá ngắn)
2. Người dùng gõ "Ng" → Gợi ý: "Nguyễn Văn A", "Nguyễn Oanh", "Ngô Thị"
3. Người dùng gõ "Nguyen" → Gợi ý: "Nguyễn Văn A", "Nguyễn Oanh"
4. Người dùng gõ "Nguyen V" → Gợi ý: "Nguyễn Văn A"

**Loại gợi ý:**

* Tên thường gặp
* Username phổ biến
* Người từng tương tác
* Lịch sử tìm kiếm

**2.4 Sắp xếp và xếp hạng**

**Công thức tính điểm:**

Friend Score = (Mutual Friends × 0.4) + (Activity × 0.3) + (Relevance × 0.2) + (Same Background × 0.1)

**Chi tiết thang điểm:**

**Mutual Friends Score (40%)**

* 100+ bạn chung: 100 điểm
* 50-99 bạn chung: 80 điểm
* 20-49 bạn chung: 60 điểm
* 10-19 bạn chung: 40 điểm
* 1-9 bạn chung: 20 điểm
* Không có bạn chung: 0 điểm

**Activity Score (30%)**

* Online hiện tại: 100 điểm
* Hoạt động trong 1 giờ: 90 điểm
* Hoạt động trong ngày: 70 điểm
* Hoạt động trong tuần: 50 điểm
* Không hoạt động lâu: 20 điểm

**Relevance Score (20%)**

* Exact match họ tên: 100 điểm
* Partial match họ tên: 80 điểm
* Match username: 60 điểm
* Match nickname: 40 điểm
* Match bio: 20 điểm

**Same Background Score (10%)**

* Cùng lớp: 60 điểm
* Cùng khoa: 30 điểm
* Khác background: 0 điểm

**2.5 Phân trang Infinite Scroll**

**Cơ chế hoạt động:**

1. Batch đầu tiên: Load 20 bạn bè đầu tiên, sắp xếp theo Friend Score
2. Batch tiếp theo: Load 20 bạn bè tiếp theo từ cursor
3. Điều kiện dừng: Hết dữ liệu hoặc đạt giới hạn 200 kết quả

**3. NGHIỆP VỤ TÌM KIẾM BÀI VIẾT**

**3.1 Tìm kiếm Multi-Match**

**Mô tả nghiệp vụ:**

* Đầu vào: Người dùng nhập "JavaScript tutorial"
* Xử lý: Tìm kiếm trong các trường nội dung
* Đầu ra: Danh sách bài viết được xếp hạng

**Chi tiết xử lý:**

|  |  |  |
| --- | --- | --- |
| **Trường tìm kiếm** | **Boost** | **Ví dụ** |
| Tiêu đề | x3 | "JavaScript Tutorial for Beginners" |
| Nội dung | x2 | "Hôm nay mình sẽ hướng dẫn JavaScript..." |
| Hashtags | x1.5 | "#javascript #tutorial #coding" |
| Tên tác giả | x1 | "JavaScript Developer" |

**3.2 Lọc dữ liệu (Bool Query)**

**Cấu trúc lọc:**

**Must (Bắt buộc)**

* Chứa từ khóa tìm kiếm
* Bài viết đã xuất bản
* Không bị xóa/ẩn

**Filter (Lọc nhanh)**

* Thời gian: Hôm nay, tuần này, tháng này
* Loại media: Text, Image, Video
* Quyền riêng tư: Public, Friends, Private
* Trạng thái: Published, Draft, Archived

**Must\_not (Loại trừ)**

* Bài viết bị báo cáo
* Tác giả bị chặn
* Nội dung spam/giả

**3.3 Sắp xếp và xếp hạng**

**Công thức tính điểm:**

Post Score = (Relevance × 0.6) + (Recency × 0.25) + (Engagement × 0.15)

**Chi tiết thang điểm:**

**Relevance Score (60%)**

* Exact match title: 100 điểm
* Partial match title: 80 điểm
* Match trong content: 60 điểm
* Match trong hashtag: 40 điểm
* Match tên tác giả: 30 điểm

**Recency Score (25%)**

* Đăng trong 1 giờ: 100 điểm
* Đăng trong ngày: 90 điểm
* Đăng trong tuần: 70 điểm
* Đăng trong tháng: 50 điểm
* Đăng cũ hơn: 20 điểm

**Engagement Score (15%)**

* Like: 1 điểm/like
* Comment: 3 điểm/comment
* Share: 5 điểm/share

**Điểm thưởng:**

* Từ giảng viên: +20 điểm
* Hashtag trending: + 20 điểm
* Bài viết được ghim: +15 điểm
* Có file đính kèm: +10 điểm
* Từ bạn bè: +8 điểm

**3.4 Bộ lọc nhanh (Quick Filters)**

**"Mới nhất"**

* Sắp xếp theo thời gian tạo giảm dần
* Ưu tiên bài viết trong 24h
* Hiển thị timestamp rõ ràng

**"Phổ biến"**

* Sắp xếp theo tương tác (likes + comments×3 + shares×5)
* Ngưỡng tối thiểu: ≥5 tương tác
* Ưu tiên bài viết trong 7 ngày

**"Từ bạn bè"**

* Lọc theo danh sách bạn bè + following
* Sắp xếp theo độ thân thiết
* Loại trừ unfriended, blocked

**"Có hình ảnh"**

* Lọc media\_type chứa "image" hoặc "video"
* Ưu tiên: video > image

**"Từ giảng viên"**

* Lọc author\_role = "teacher"
* Boost điểm số cao
* Hiển thị badge "Giảng viên"

**3.5 Phân trang Infinite Scroll**

**Cơ chế hoạt động:**

1. Batch đầu tiên: Load 15 bài viết, preload ảnh/video batch tiếp theo
2. Batch tiếp theo: Load 15 bài viết từ cursor, lazy load media
3. Điều kiện dừng: Hết dữ liệu hoặc đạt giới hạn 300 kết quả

**4. KIẾN TRÚC DỮ LIỆU**

**4.1 Phân quyền dữ liệu**

|  |  |  |
| --- | --- | --- |
| Thành phần | Vai trò | Mô tả |
| PostgreSQL | Source of Truth | Nguồn dữ liệu gốc, chứa toàn bộ thông tin |
| Elasticsearch | Search Engine | Bản sao phục vụ tìm kiếm, cache tăng hiệu suất |
| Redis | Queue | Xử lý job queue |

**4.2 Đồng bộ dữ liệu**

**Khi tạo mới**

1. Lưu vào PostgreSQL
2. Tạo document mới trong Elasticsearch

**Khi cập nhật**

1. Update PostgreSQL thành công
2. Update document tương ứng trong Elasticsearch
3. Chỉ cập nhật các trường liên quan đến tìm kiếm

**Khi xóa**

1. Xóa bản ghi PostgreSQL
2. Xóa document khỏi Elasticsearch

**4.3 Xử lý lỗi đồng bộ**

**Khi Elasticsearch lỗi:**

1. Ghi job lỗi vào Redis (es:failures)
2. Worker chạy 10s/lần để retry
3. Retry >3 lần → chuyển sang dead-letter queue
4. Xử lý thủ công các job thất bại

**5. LUỒNG XỬ LÝ NGHIỆP VỤ**

**5.1 Luồng tìm kiếm bạn bè**

1. Người dùng nhập "Nguyễn Văn"

↓

2. Auto-complete gợi ý tên trong lúc gõ

↓

3. Multi-match tìm kiếm trong thông tin cá nhân

↓

4. Bool query áp dụng filters (giới tính, khoa, năm học)

↓

5. Ranking tính điểm dựa trên bạn chung, hoạt động

↓

6. Trả về top 20 kết quả đầu tiên

↓

7. User áp dụng quick filters (cùng lớp, online)

↓

8. Infinite scroll load thêm kết quả

**5.2 Luồng tìm kiếm bài viết**

1. Người dùng nhập "JavaScript tutorial"

↓

2. Auto-complete gợi ý hashtag, từ khóa

↓

3. Multi-match tìm kiếm trong title, content, hashtags

↓

4. Bool query áp dụng filters (thời gian, media, quyền riêng tư)

↓

5. Ranking tính điểm dựa trên relevance, recency, engagement

↓

6. Trả về top 15 kết quả đầu tiên

↓

7. User áp dụng quick filters (mới nhất, phổ biến, từ bạn bè)

↓

8. Infinite scroll load thêm kết quả

**5.3 Xử lý lỗi và fallback**

**Khi Elasticsearch lỗi:**

* Hiển thị thông báo: "Tìm kiếm tạm thời bị gián đoạn"
* Ghi log lỗi để xử lý

**Khi kết nối chậm:**

* Hiển thị loading indicator
* Timeout sau 10 giây
* Cho phép retry thủ công

**6. XỬ LÝ TRƯỜNG HỢP ĐẶC BIỆT**

**6.1 Người dùng bị khóa**

* Xử lý: Không hiển thị trong kết quả tìm kiếm
* Cách thức: Xóa khỏi Elasticsearch ngay lập tức
* Lọc: Sử dụng must\_not (status = "locked")

**6.2 Người dùng ẩn profile**

* Điều kiện: is\_hidden = true
* Xử lý: Không index vào Elasticsearch
* Cập nhật: Khi chuyển từ true → false mới được index trở lại

**6.3 Người dùng bị báo cáo/spam**

* Index: Vẫn được index nếu chưa bị khóa
* Xếp hạng: Giảm điểm xếp hạng
* Lọc: Thêm must\_not: {reported: true} nếu cần

**6.4 Bài viết riêng tư/bị ẩn**

* Private/Draft: Không index hoặc chỉ visible với tác giả
* Bị báo cáo: Loại khỏi kết quả hoặc đánh điểm thấp
* Quyền truy cập: Kiểm tra permission trước khi hiển thị