RAG System

Database Advisory Use Case

1. Welche Datenarten existieren?

- **Dokumente (PDFs, Specs)** → MongoDB (flexibel, schemalos)
- **Relational (User, Credits)** → PostgreSQL (ACID, Foreign Keys)
- **JSON (Architektur-Schemas)** → MongoDB (semi-strukturiert)
- **Vektoren (Embeddings)** → PostgreSQL pgvector (semantische Suche)

2. Wo entsteht Konsistenzdruck?

- **Credits-Abrechnung** → PostgreSQL ACID (keine Doppelabbuchung)
- **User-Permissions** → PostgreSQL Row-Level Security
- **Audit/Compliance** → PostgreSQL (unveränderlich, 7 Jahre)

MongoDB ist "eventually consistent", Redis ist flüchtig → nicht für kritische Daten

3. Wo entsteht flüchtiger State?

- **Sessions (24h)** → Redis (User eingeloggt)
- **Rate Limits (60s)** → Redis (10 Requests/Min Counter)
- **Chat Buffer (30min)** → Redis (letzte 5 Nachrichten)
- **Query Cache (7d)** → Redis (häufige Suchergebnisse)

Alles In-Memory, extrem schnell, automatisches Löschen (TTL)

4. Wo werden Vektoren benötigt?

- **Semantische Doc-Suche** → "Finde ähnliche Docs zu Microservices"
- **Architektur-Muster Matching** → "Zeig ähnliche Architekturen wie Netflix"
- **User-Intent Erkennung** → "Wie skaliere ich?" → System erkennt Absicht

Alle in PostgreSQL pgvector (stabil, kostenlos, gut für <1M Vektoren)

5. Wo braucht ihr Audit Trails?

- **Recommendations (7 Jahre)** → Jede Architekturempfehlung (Compliance)
- **User Prompts (DSGVO)** → Was hat User gefragt? (Datenschutz)
- **RAG Traces (1 Jahr)** → System-Debugging, Performance-Analyse
- **Access Logs (90d)** → Wer hat wann auf was zugegriffen? (Sicherheit)

Alle in PostgreSQL: unveränderlich (Append-Only), ACID-Garantie

6. Logische Trennung?

- **READ Path (Semantik)** → PostgreSQL pgvector + MongoDB Docs
- **WRITE Path (Konsistenz)** → PostgreSQL ACID (Credits, Logs)
- **SESSION Path (Ephemär)** → Redis (Sessions, Rate Limits)

Trennung erlaubt: READ skaliert horizontal, WRITE bleibt ACID, SESSION-Fehler = neu einloggen (unkritisch)