Architecture & Data Flow (IndexedDB + Neo4j → MongoDB Lake)

- • IndexedDB (browser) → provides client-side sensor data.
- • Neo4j → stores connected graph relationships (Farm–Device–Reading).
- • Node.js pipeline integrates both datasets.
- • MongoDB Atlas Data Lake stores unified, tagged documents.

- Metadata fields stored with each record:
- • sourceDB (IndexedDB or Neo4j)
- • ingestedAt (UTC timestamp)
- • tags   (descriptive keywords)

Lessons Learned from Extending the Pipeline

- ✅ What Worked Well:
- • Integrated two heterogeneous databases successfully.
- • MongoDB handled flexible schema and metadata tagging easily.
- • Neo4j driver allowed efficient traversal of relationships.
- • UUID and metadata fields improved data traceability.

- 💡 New Skills Learned:
- • Working with multiple database drivers in Node.js.
- • Using insertMany() for batch ingestion.
- • Querying and filtering data via MongoDB Compass.

Challenges & Reflections

- ⚙️ Challenges:
- • Managing both MongoDB and Neo4j services concurrently.
- • Handling connection errors (ECONNREFUSED) and async issues.
- • Simulating IndexedDB in Node.js (browser data limitation).
- • Ensuring both database sessions closed properly.

- 🚀 Future Improvements:
- • Automate ingestion with cron jobs.
- • Add validation and error logging.
- • Connect real IndexedDB API instead of mock data.
- • Extend metadata tags for analytics (e.g., 'IoT', 'graph-import').