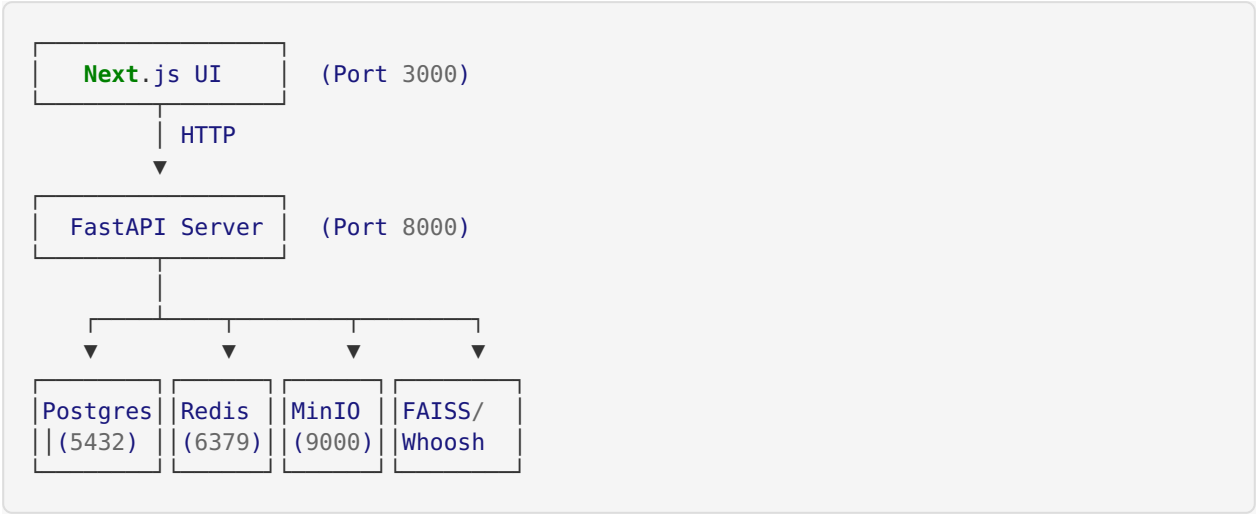
System Architecture

Overview

The GREDs AI Reference Library follows a microservices architecture with clear separation between frontend, backend, and infrastructure services.

Component Diagram



Core Components

Frontend (Next.js 14)

- Server-side rendering with App Router
- shadcn/ui component library
- Tailwind CSS for styling
- TypeScript for type safety

Backend (FastAPI)

- RESTful API with automatic OpenAPI documentation
- Async request handling with Uvicorn
- Pydantic models for data validation
- Structured logging with structlog

Database Layer

- **PostgreSQL**: Metadata, chunks, sessions, citations
- **FAISS**: Vector embeddings for semantic search
- **Whoosh**: Inverted index for BM25 lexical search

Storage Layer

- **MinIO/S3**: Immutable audit logs, raw documents, artifacts
- **Redis**: Task queue for background jobs

Data Flow

Ingestion Pipeline

1. User uploads document or provides repository URL
2. Backend extracts text and creates chunks
3. Embeddings generated via sentence-transformers
4. Chunks indexed in FAISS and Whoosh
5. Summaries generated at three levels
6. Metadata stored in PostgreSQL

Query Pipeline

1. User submits natural language query
2. Query embedded using same model
3. Parallel search in FAISS (semantic) and Whoosh (lexical)
4. Results merged with hybrid scoring (0.7 semantic + 0.3 lexical)
5. Top-K results returned with citations

Verification Pipeline

1. Claims extracted from LLM output
2. Each claim compared to cited chunk
3. Cosine similarity calculated
4. Pass/Partial/Fail decision based on thresholds
5. Results logged to audit trail

Security Considerations

- Environment-based configuration
- CORS restricted to frontend origin
- Database credentials in environment variables
- S3 access via IAM roles (production)

Scalability

- FAISS index sharding for large corpora
- Redis queue for distributed workers
- PostgreSQL connection pooling
- Docker Compose for local development
- Kubernetes-ready containerization