MES/SCADA RAG Documentation System

Projektová Specifikace a Architektura

1. EXECUTIVE SUMMARY

Projekt: Inteligentní systém pro správu a vyhledávání dokumentace MES/SCADA s podporou ISA-95 hierarchie

Cíl: Vytvořit snadno nasaditelný (Docker Compose) RAG systém pro průmyslové zákazníky, který kombinuje tradiční dokumentový management s inteligentním vyhledáváním a strukturou podle výrobních standardů.

Klíčové hodnoty:

- Okamžité nasazení (on-premise i cloud)
- Intuitivní chat interface pro koncové uživatele
- Flexibilní organizační struktura (ISA-95 inspirovaná, ale přizpůsobitelná)
- Podpora stovek GB dokumentů, tisíc souborů
- Enterprise security (SSO integrace)

2. STAKEHOLDER ANALÝZA

2.1 Primární uživatelé

- **Operátoři strojů** - hledají rychlé odpovědi na provozní problémy
- **Maintenance technici** - potřebují manuály, postupy, troubleshooting
- **Proces inženýři** - přístup k specifikacím, standardům, výkresům
- **Management** - reporty, přehledy, compliance dokumenty

2.2 Administrátoři systému

- **IT správci** - nasazení, údržba, monitoring
- **Dokumentový správci** - nahrávání, kategorizace, aktualizace
- **Struktura správci** - definice a úpravy organizační hierarchie

2.3 Stakeholder požadavky

- **Rychlost:** Odpověď do 3 sekund na typickou otázku
- **Přesnost:** Relevantní výsledky v 90%+ případů
- **Dostupnost:** 99.5% uptime pro produkční prostředí
- **Bezpečnost:** Firemní standard (AD/SSO + RBAC)

3. FUNKČNÍ POŽADAVKY

3.1 Core Features - MVP

1. **Chat Interface**
   - Conversational AI pro dotazy na dokumenty
   - Kontextové odpovědi s odkazy na zdroje
   - Historie konverzací (session-based)
2. **Hierarchická Struktura (Manufacturing Hierarchy Manager)**
   - Flexibilní strom organizace (ISA-95 inspirovaný)
   - CRUD operace na struktuře
   - Drag & drop reorganizace
3. **Document Management**
   - Upload multiple formátů (PDF, DOCX, XLSX, TXT, HTML)
   - Automatic text extraction a indexing
   - Metadata management (typ, autor, verze, tagy)
   - Přidělení dokumentů k hierarchii
4. **RAG Search Engine**
   - Sémantické vyhledávání v dokumentech
   - Kombinace s hierarchickým kontextem
   - Relevance scoring a ranking

3.2 Advanced Features - Post MVP

1. **Enterprise Integration**
   - SSO integrace (SAML, OIDC, AD)
   - Role-based access control
   - Audit logging
2. **Document Lifecycle**
   - Verzování dokumentů
   - Approval workflows
   - Archive management
3. **Analytics & Reporting**
   - Usage statistics
   - Popular queries
   - Content gap analysis

4. NEFUNKČNÍ POŽADAVKY

4.1 Performance

- **Response Time:** < 3s pro typickou RAG query
- **Throughput:** 100+ concurrent users
- **Scalability:** Horizontální škálování přes container orchestration

4.2 Storage

- **Document Storage:** Stovky GB, tisíce souborů
- **Vector Database:** Miliony embeddings
- **Backup:** Automated daily backups s point-in-time recovery

4.3 Security

- **Authentication:** Firemní SSO priorita, fallback local accounts
- **Authorization:** Role-based permissions na úrovni hierarchie
- **Data Protection:** Encryption at rest a in transit
- **Compliance:** Audit trail pro všechny operace

4.4 Deployment

- **Container-first:** Docker Compose pro jednoduché nasazení
- **Environment Support:** Development, staging, production configs
- **Cloud-agnostic:** Běží on-premise i v cloudu
- **Monitoring:** Health checks, metrics, logging

5. TECHNICKÝ STACK - DECISION MATRIX

5.1 Frontend

- **Framework:** Vanilla JS + Tailwind CSS
- **Důvod:** Jednoduchost, rychlost, malá footprint
- **Architecture:** SPA s API-first approach
- **Key Libraries:** Axios (HTTP), Chart.js (analytics)

5.2 Backend

- **Framework:** FastAPI (Python)
- **Důvod:** Rychlý development, excellent API docs, async support
- **Architecture:** Clean Architecture + Domain-Driven Design patterns

- **Key Libraries:** SQLAlchemy (ORM), Pydantic (validation)

5.3 Data Layer

- **Primary Database:** PostgreSQL
  - Důvod: Mature, excellent JSON support, scalability
- **Vector Database:** Qdrant
  - Důvod: Production-ready, excellent metadata filtering, scalable
- **Caching:** Redis (session, query cache)

5.4 Document Storage

ROZHODNUTÍ POTŘEBNÉ: Doporučení pro storage layer:

Opce A: Filesystem (doporučeno pro MVP)

- Prostý filesystem s organizovanou strukturou
- Backup přes standard filesystem tools
- Jednoduchá implementace

Opce B: MinIO (S3-compatible)

- Object storage s S3 API
- Built-in redundancy a versioning
- Cloud-ready

Opce C: Existující DMS (SharePoint, Alfresco)

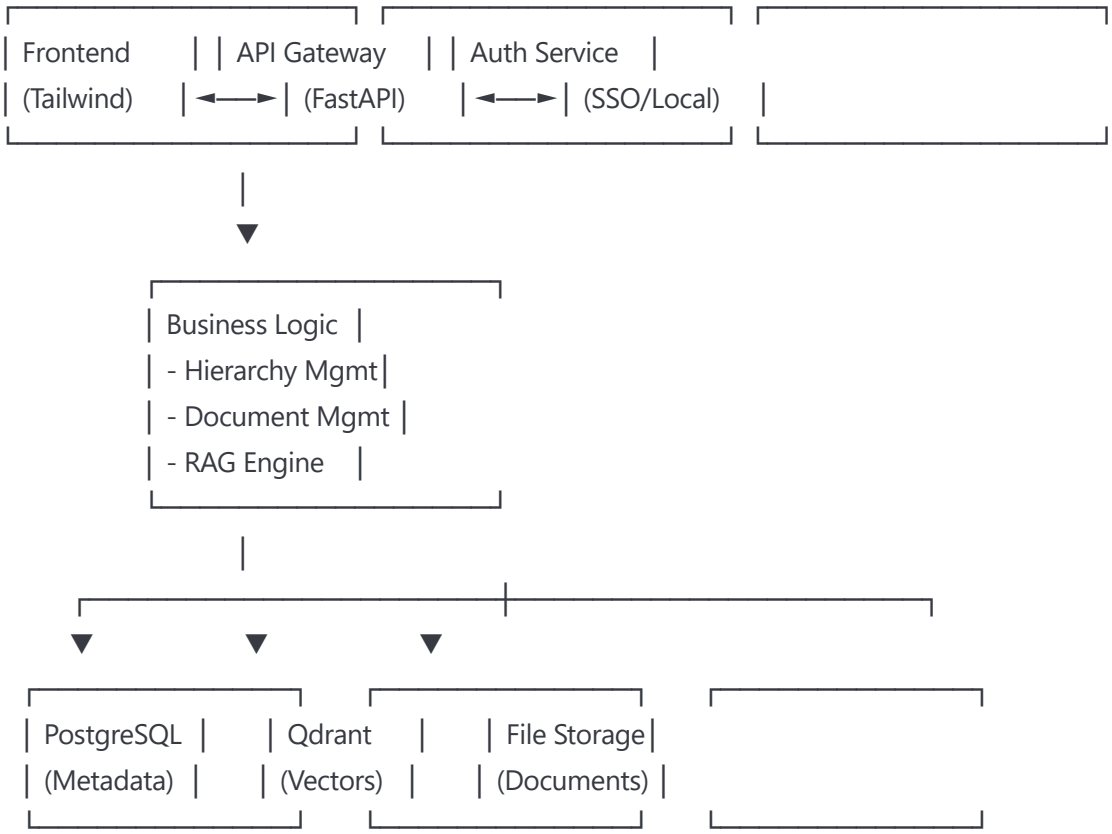
- Integrace s existijícími systémy
- Komplexnější implementace
- Vendor lock-in risk

5.5 ML/AI Stack

- **Embeddings:** Ollama (local deployment)
  - Důvod: Privacy, cost control, offline capability
- **Text Extraction:** PyMuPDF (PDF), python-docx (Word), pandas (Excel)
- **Chunking Strategy:** Semantic chunking s overlap

6. SYSTÉMOVÁ ARCHITEKTURA

6.1 High-Level Architecture



6.2 Docker Compose Services

yaml

```yaml
services:
  frontend:    # Nginx + static files
  api:         # FastAPI aplikace
  postgres:    # Metadata database
  qdrant:      # Vector database
  redis:       # Cache & sessions
  ollama:      # Local LLM for embeddings
  nginx:       # Reverse proxy
```

7. DEVELOPMENT METHODOLOGY

7.1 Vibe Coding Rules

1. **Architecture First:** Žádný kód bez jasného návrhu
2. **Domain-Driven:** Vždy začínat od business logic, ne od technologie
3. **Contract-First:** API endpoints definovány před implementací
4. **Test-Driven:** Unit testy pro business logic, integration testy pro API
5. **Documentation-Driven:** Vše musí být dokumentováno pro budoucí přebírání

7.2 Development Phases

Phase 1: Foundation (2-3 týdny)

- Docker Compose setup
- Database schema design
- Basic API structure
- Authentication framework

Phase 2: Core MVP (3-4 týdny)

- Hierarchy management
- Document upload & processing
- Basic RAG implementation
- Simple frontend

Phase 3: Enhancement (2-3 týdny)

- Advanced search features
- UI/UX improvements
- Performance optimization
- Production deployment prep

7.3 Quality Gates

- **Code Review:** Vše reviewed před merge
- **Testing:** Min 80% code coverage
- **Performance:** Load testing před production
- **Security:** Security audit před deployment
- **Documentation:** Technical & user documentation complete

8. RISK MANAGEMENT

8.1 Technical Risks

Risk	Probability	Impact	Mitigation
Ollama performance issues	Medium	High	Fallback na cloud embeddings
Qdrant scalability limits	Low	Medium	Horizontal scaling, collection sharding
Large file processing	High	Medium	Chunking strategy + async processing

8.2 Business Risks

Risk	Probability	Impact	Mitigation
Customer SSO integration complexity	High	Medium	Phased rollout s local auth fallback
Regulatory compliance	Medium	High	Security audit + compliance framework
Performance at scale	Medium	High	Load testing + horizontal scaling

9. SUCCESS METRICS

9.1 Technical KPIs

- **System Uptime:** >99.5%
- **Query Response Time:** <3 seconds (95th percentile)
- **Document Processing Time:** <30 seconds per document
- **Search Accuracy:** >90% relevant results

9.2 Business KPIs

- **User Adoption:** >70% active users monthly
- **Query Success Rate:** >85% queries get useful answers
- **Document Coverage:** >80% documents actively accessed
- **Customer Satisfaction:** >4.5/5 rating

10. NEXT STEPS - IMMEDIATE ACTIONS

10.1 Week 1: Foundation Setup

1. **Environment Setup**
   - Docker Compose environment
   - Development database
   - CI/CD pipeline basics
2. **Architecture Validation**
   - API contract definition
   - Database schema design
   - Security framework