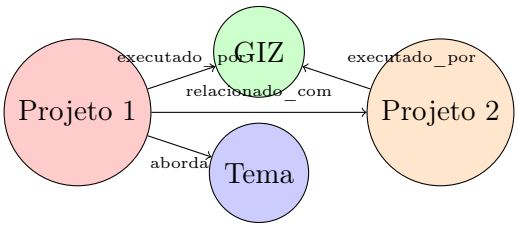
Knowledge Graph para Produtos Digitais

GIZ-Brasil

Protótipo de Sistema de Grafo de Conhecimento
para Gestão de Projetos e Metadados



31 de julho de 2025

Tecnologias: Python, Neo4j, Streamlit, Docker
Versão: 1.0
Licença: MIT

Conteúdo

1 Resumo Executivo

O projeto **Knowledge Graph GIZ-BR** é um protótipo completo que consolida metadados de projetos da GIZ-Brasil em um banco de dados orientado a grafos (Neo4j) e oferece uma interface web interativa através do Streamlit para visualização e análise das relações entre projetos, organizações e temas.

1.1 Objetivos Principais

- Centralizar informações de projetos em um formato estruturado
- Identificar relações e dependências entre projetos
- Facilitar a descoberta de conhecimento através de visualizações interativas
- Prover uma plataforma escalável para análise de portfólio de projetos

1.2 Benefícios

- **Visibilidade:** Visualização clara das conexões entre projetos
- **Análise:** Identificação de padrões e oportunidades de sinergia
- **Gestão:** Melhor tomada de decisão baseada em dados relacionais
- **Escalabilidade:** Arquitetura preparada para crescimento

2 Arquitetura do Sistema

2.1 Visão Geral

O sistema é composto por três camadas principais:

1. **Camada de Dados**: Neo4j como banco de grafos
2. **Camada de Processamento**: Scripts Python para ETL
3. **Camada de Apresentação**: Interface web Streamlit

2.2 Tecnologias Utilizadas

2.3 Arquitetura de Dados

O modelo de dados é baseado em grafos, onde:

- **Nós (Vértices):** Representam entidades como Projetos, Organizações e Temas
- **Arestas (Relacionamentos):** Representam conexões como "executado_por", "aborda", "relacionado_com"
- **Propriedades:** Metadados associados aos nós (nome, status, orçamento, etc.)

Componente	Tecnologia	Versão
Banco de Dados	Neo4j Community	5.20+
Backend	Python	3.10+
Frontend	Streamlit	1.35+
Visualização	PyVis	0.3.1+
Containerização	Docker	20.10+
Processamento de Dados	Pandas	2.2+

Tabela 1: Stack Tecnológica do Projeto

3 Componentes do Sistema

3.1 ETL (Extract, Transform, Load)

O módulo ETL é responsável pela ingestão de dados:

```python
def upsert_project(tx, props: dict):
    """
    Garante um n   Project com id   nico   e
    atualiza/insere todas as demais propriedades.
    """
    tx.run(
        """
        MERGE (p:Project {id: $id})
        SET   p += $props
        """,
        id=props["id"],
        props=props
    )
```

Listing 1: Função de Upsert de Projetos

Características:

- Operações MERGE para evitar duplicatas
- Processamento incremental de dados
- Tratamento de diferentes tipos de dados
- Log de operações realizadas

3.2 Interface Web (Streamlit)

A aplicação web oferece:

- **Seleção de Projetos:** Dropdown com todos os projetos disponíveis
- **Informações Detalhadas:** Exibição de metadados do projeto selecionado
- **Visualização de Grafo:** Representação interativa usando PyVis
- **Estatísticas:** Métricas sobre conectividade e relações
- **Dados Tabulares:** Fallback em formato tabela para análise detalhada

3.3 Banco de Dados Neo4j

Configuração do banco:

```
services:
  neo4j:
    image: neo4j:5.20-community
    restart: unless-stopped
    ports:
      - "7474:7474"  # Interface web
      - "7687:7687"  # Protocolo Bolt
    environment:
      - NEO4J_AUTH=neo4j/test12345
      - NEO4J_PLUGINS=["apoc"]
```

Listing 2: Docker Compose para Neo4j

4 Modelo de Dados

4.1 Entidades Principais

1. **Project**
   - Propriedades: id, name, status, start_date, budget
   - Exemplo: AdaptaInfra, Gênero&Infra, Clima-Amazônia
2. **Organizacao**
   - Propriedades: name, tipo
   - Exemplo: GIZ Brasil, Ministério da Infraestrutura
3. **Tema**
   - Propriedades: name, area
   - Exemplo: Sustentabilidade, Igualdade de Gênero

4.2 Relacionamentos

Origem	Relacionamento	Destino
Project	EXECUTADO_POR	Organizacao
Project	ABORDA	Tema
Project	RELACIONADO_COM	Project
Project	INFLUENCIA	Project

Tabela 2: Tipos de Relacionamentos no Grafo

5 Funcionalidades

5.1 Visualização Interativa

A interface oferece:

- **Grafo Dinâmico:** Visualização em tempo real das conexões
- **Nós Coloridos:** Diferenciação visual por tipo de entidade
- **Tooltips Informativos:** Detalhes ao passar o mouse
- **Layout Adaptativo:** Organização automática dos elementos

5.2 Análise de Dados

- **Métricas de Conectividade:** Número de relações por projeto
- **Estatísticas do Grafo:** Contadores de entidades conectadas
- **Dados Estruturados:** Exportação em formato tabular
- **Filtros:** Seleção específica por projeto

6 Guia de Instalação

6.1 Pré-requisitos

- Python 3.10 ou superior
- Docker e Docker Compose
- Git (para clonagem do repositório)
- 4GB RAM disponível (para Neo4j)

6.2 Processo de Instalação

1. **Clonar o repositório:**

```
git clone <repository-url>
cd knowledge-graph-giz-GT

```

2. **Configurar ambiente Python:**

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt

```

3. **Iniciar Neo4j:**

```
docker compose up -d

```

4. **Carregar dados:**

```
python etl/load_metadata.py
```

5. **Executar aplicação:**

```
streamlit run app/streamlit_app.py
```

7 Estrutura do Projeto

```
knowledge-graph-giz-GT/
          app/
                streamlit_app.py      # Interface principal
                secrets.toml          # Credenciais
          data/
                projects.csv          # Dados de entrada
          etl/
                load_metadata.py      # Script ETL
          docker-compose.yml        # Configura  o Neo4j
          requirements.txt          # Depend ncias Python
          README.md                 # Documenta  o
```

Listing 3: Organização de Diretórios

8 Casos de Uso

8.1 Gestão de Portfólio

- Identificar projetos relacionados para possível consolidação
- Analisar distribuição de recursos por tema
- Detectar gaps ou sobreposições no portfólio

8.2 Análise de Impacto

- Avaliar o alcance de temas específicos
- Identificar organizações parceiras estratégicas
- Mapear dependências entre projetos

8.3 Planejamento Estratégico

- Visualizar a rede de relacionamentos organizacionais
- Identificar oportunidades de colaboração
- Otimizar alocação de recursos

9 Extensões Futuras

9.1 Roadmap Técnico

1. **Busca Semântica:** Implementação de índices vetoriais
2. **APIs REST:** Exposição de endpoints para integração
3. **Autenticação:** Sistema de login e controle de acesso
4. **Dashboard Analytics:** KPIs e métricas avançadas
5. **Machine Learning:** Recomendações e predições

9.2 Melhorias de Interface

1. Interface responsiva para dispositivos móveis
2. Filtros avançados por múltiplos critérios
3. Exportação de visualizações em diferentes formatos
4. Histórico de navegação e favoritos
5. Compartilhamento de visualizações via URL

10 Segurança e Boas Práticas

10.1 Segurança

- **Credenciais:** Armazenamento em arquivos de configuração separados
- **Validação:** Sanitização de entradas de usuário
- **Criptografia:** Comunicação segura via HTTPS (produção)
- **Backup:** Procedimentos de backup do banco de dados

10.2 Performance

- **Cache:** Uso de cache do Streamlit para consultas
- **Índices:** Criação de índices apropriados no Neo4j
- **Paginação:** Limitação de resultados em consultas grandes
- **Otimização:** Consultas Cypher eficientes

11 Conclusão

O projeto Knowledge Graph GIZ-BR representa uma solução moderna e escalável para gestão de conhecimento organizacional. Através da combinação de tecnologias de grafos, processamento de dados e visualização interativa, oferece uma plataforma robusta para análise e descoberta de relações em portfólios de projetos.

11.1 Principais Contribuições

- Arquitetura modular e extensível
- Interface intuitiva para usuários não-técnicos
- Modelo de dados flexível para diferentes tipos de entidades
- Demonstração prática de tecnologias de grafo em contexto organizacional

11.2 Impacto Esperado

A implementação desta solução pode resultar em:

- Melhoria na tomada de decisões estratégicas
- Identificação de oportunidades de sinergia
- Otimização na alocação de recursos
- Maior visibilidade das atividades organizacionais

O projeto estabelece uma base sólida para futuras expansões e pode servir como modelo para outras organizações interessadas em implementar soluções similares de gestão de conhecimento.

12 Apêndices

12.1 Apêndice A: Comandos Cypher Úteis

```
-- Listar todos os projetos
MATCH (p:Project) RETURN p

-- Encontrar projetos relacionados
MATCH (p1:Project)-[r]-(p2:Project)
RETURN p1.name, type(r), p2.name

-- Estat sticas do grafo
MATCH (n) RETURN labels(n), count(n)

-- Projetos por organiza  o
MATCH (p:Project)-[:EXECUTADO_POR]->(o:Organizacao)
RETURN o.name, collect(p.name)
```

Listing 4: Consultas Cypher Essenciais

12.2 Apêndice B: Configurações Avançadas

```
g.set_options("""
var options = {
  "physics": {
    "enabled": true,
    "stabilization": {"iterations": 100}
  },
  "nodes": {
    "font": {"size": 16}
  },
  "edges": {
    "font": {"size": 14}
  }
}
""")
```

Listing 5: Configurações do PyVis