Визуализация и анализ технологических зависимостей

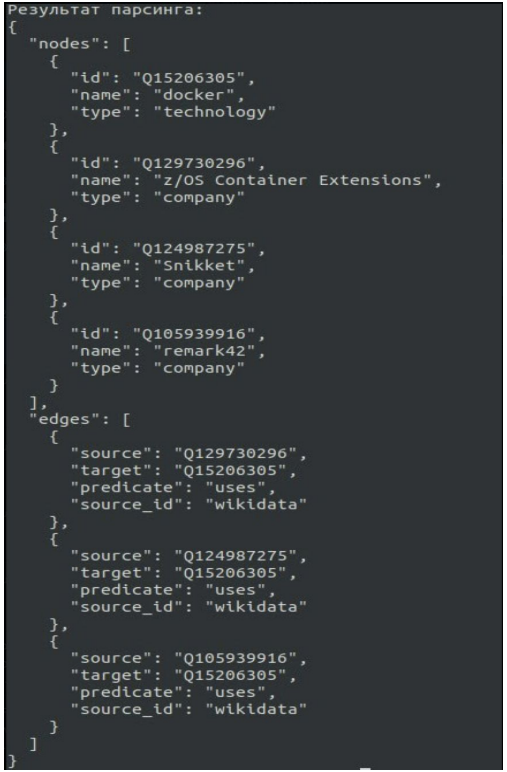
Итерация 1

План на итерацию

1. Разработать прототип парсера данных об интересующих пользователя технологиях
2. Разработать прототип модуля выполнения операций для сохранения ребер и отношений в графовую СУБД Neo4j
3. Разработать макет и демо-версию пользовательского интерфейса, описать сценарии использования
4. Подготовить прототип FastAPI бэкенда для обработки основных сценариев использования
5. Подготовить основу для сборки проекта в единую экосистему (Docker + docker-compose)

Результаты итерации: парсер технологий

- Реализован прототип парсера ресурса wikidata
- Добавлена возможность получить информацию о компаниях, использующих технологию
- Создана структура модуля для дальнейшего масштабирования
- https://github.com/moevm/mse1h2026-req-viz/pull/9



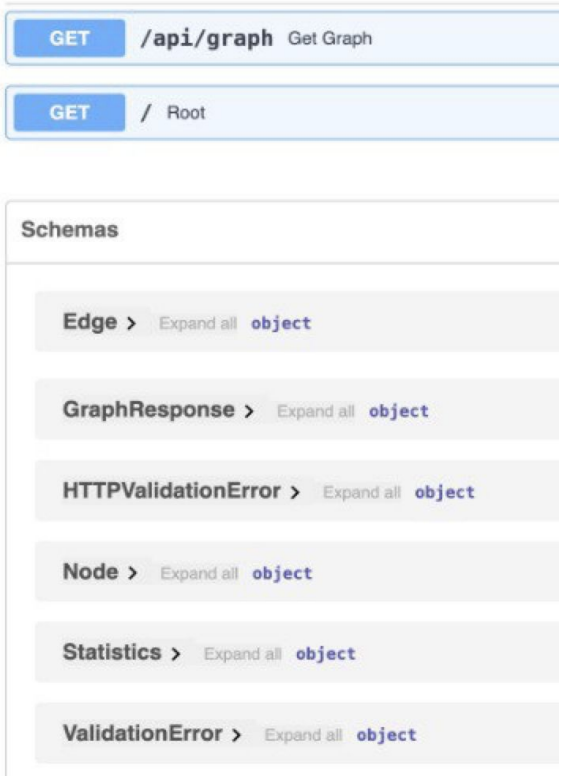
Пример работы модуля

Результаты итерации: модуль взаимодействия с Neo4j

- Реализована гибкая модель данных и отношений, позволяющая хранить любые сущности без изменения схемы данных
- Создан модуль подключения к Neo4j с retry-логикой и менеджером контекста
- Добавлены Pydantic-модели для описания и валидации параметров запросов в БД
- Реализован репозиторий с полным CRUD для узлов и отношений, поддержкой взвешенных рёбер и извлечением подграфов (по центру и по фильтрам)
- Добавлен сервисный слой с бизнес-логикой (проверка дубликатов, существования узлов, логирование).
- https://github.com/moevm/mse1h2026-req-viz/pull/8

Результаты итерации: FastAPI бэкенд

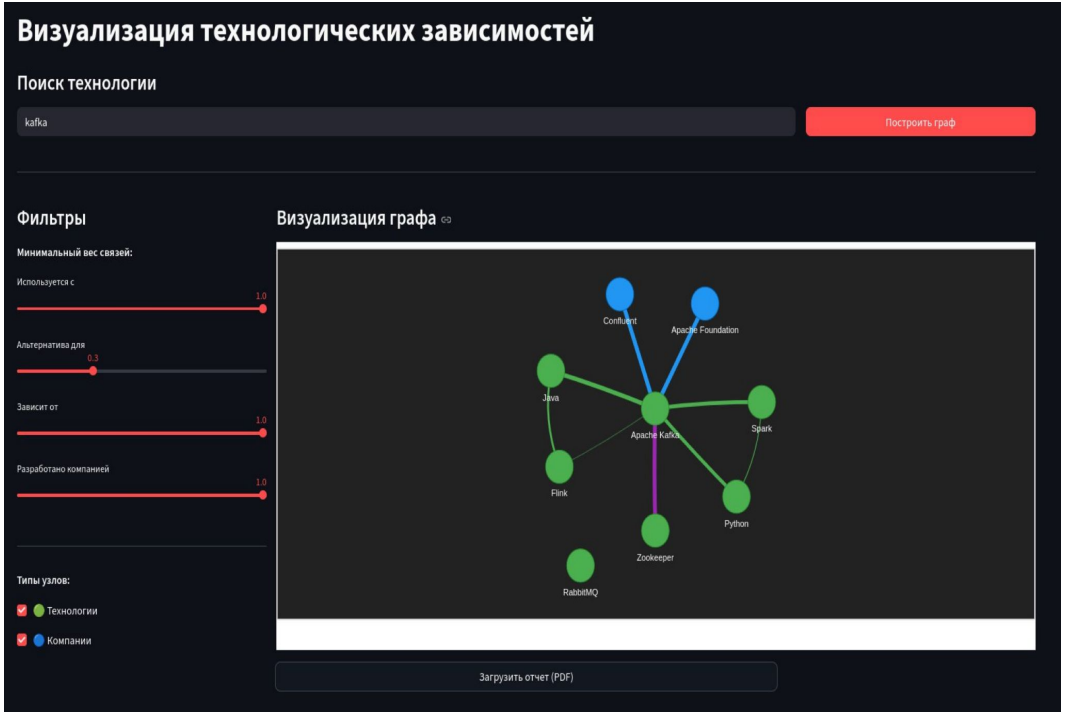
- Добавлены основные эндпоинты
- Реализована модель GraphResponse, соответствующая модели данных для UI
- Создана заглушка SimpleDB с in-memory хранилищем
- Создана заглушка SimpleParser с моковыми данными
- Добавлен CORS для интеграции со Streamlit
- https://github.com/moevm/mse1h2026-req-viz/pull/7




```
{
  "nodes": [
    {
      "id": "tech_001",
      "label": "Apache Kafka",
      "type": "Technology"
    },
    {
      "id": "tech_002",
      "label": "RabbitMQ",
      "type": "Technology"
    },
    {
      "id": "comp_001",
      "label": "Confluent",
      "type": "Company"
    },
    {
      "id": "lic_001",
      "label": "Apache 2.0",
      "type": "License"
    }
  ],
  "edges": [
    {
      "source": "tech_001",
      "target": "tech_002",
      "type": "ALTERNATIVE_TO",
      "weight": 0.9
    },
    {
      "source": "tech_001",
      "target": "comp_001",
      "type": "DEVELOPED_BY",
      "weight": 1
    },
    {
      "source": "tech_001",
      "target": "lic_001",
      "type": "LICENSED_UNDER",
      "weight": 1
    }
  ],
  "statistics": {
    "total_nodes": 4,
    "total_edges": 3,
    "max_depth": 2
  }
}
```

Результаты итерации: прототип UI

- Добавлен интерфейс Streamlit с панелями поиска и фильтрации
- Реализована визуализация графов с помощью PyVis (узлы, рёбра, веса)
- Создана заглушка MockBackendService с тестовыми данными в памяти
- Добавлена модульная структура (app.py, config.py, services.py, visualization.py)
- Реализована фильтрация по типу узлов и рёбер с помощью ползунков и чекбоксов
- Добавлены кнопки быстрого запуска для демонстрационных технологий



https://github.com/moevm/mse1h2026-req-viz/pull/6

Результаты итерации: подготовка единой экосистемы

- Создан Dockerfile для бэкенда с установкой зависимостей
- Создан Dockerfile для фронтенда на Streamlit с установкой зависимостей
- Настроен docker-compose.yml для совместного развертывания:
- backend с подключением к Neo4j
- frontend с пробросом порта 8501 наружу
- https://github.com/moevm/mse1h2026-req-viz/pull/10

План на следующую итерацию

1. Реализовать сбор идентификаторов технологий с wikidata при вводе их имени
2. Расширить конфигурацию парсера для сбора необходимых связей
3. Реализовать логику на стороне бэкенда для общения с парсером и Neo4j
4. Реализовать логику фильтрации графа
5. Подготовить прототип генерации отчета в формате PDF
6. Настроить сборку проекта
7. Покрыть основной функционал UNIT-тестами
8. Настроить автоматическое тестирование

Цель: подготовить минимально жизнеспособный продукт