

# DaVi: Ontology-Driven Semantic Data Visualization – User Guide

Practical instructions to install, run and use the DaVi system. This document is written in a concise, scholarly style with screenshots and example commands for a quick hands-on experience.

## Contents

- [Overview](#)
- [Quickstart](#)
- [API](#)
- [UI Walkthrough](#)

## 1.

### Authors

**Nastasiu Stefan**

# Lazurca Samuel-Ionut

## Version

### 1.0 – User Guide

## Repository

[Project repository \(source & code\)](#)

## Live demo

[Open live demo](#)

## 2. Overview

The DaVi system provides a data-agnostic, ontology-driven interface to explore RDF datasets. It supports:

- Interactive graph exploration (2D/3D force graphs).
- Analytics builder with dynamic filters read from the meta-ontology.
- Customizable queries over RDF data.

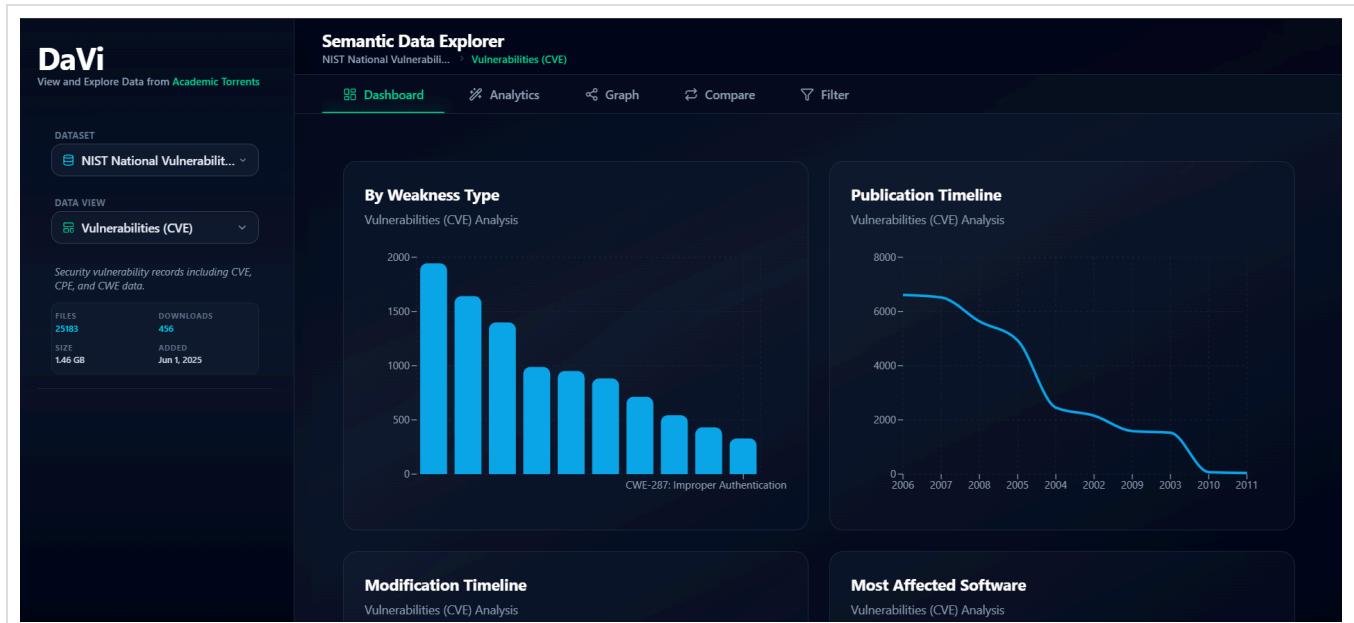


Figure 1 – DaVi dashboard (presets).

### 3. Quickstart – run DaVi locally (6 steps)

This quickstart assumes you have git, docker and node installed.

1. Clone the repository and change directory:

```
git clone https://github.com/SamuelLazurca/CrawlWebbers-WADE-Project.git  
cd CrawlWebbers-WADE-Project
```

2. Download the apache-jena-fuseki archive and extract it:

```
wget https://archive.apache.org/dist/jena/binaries/apache-jena-fuseki-5.6  
tar -xvf apache-jena-fuseki-5.6.0.tar.gz
```



3. You can load a small RDF sample into Fuseki using the interface at localhost:3030 (TTL files from the results/ folder):

4. Set up the following environment variables in a .env file in the backend folder:

```
FUSEKI_ENDPOINT=http://localhost:3030/davi/sparql  
API_KEY=your_api_key_here  
FRONTEND_URL=http://localhost:5173
```

Set up the following environment variables in a .env file in the frontend folder:

```
VITE_API_URL=http://localhost:8000/api/v1  
VITE_API_KEY=your_api_key_here
```

5. Start the backend (FastAPI) in a virtualenv:

```
# Python venv approach  
cd rest-api  
python -m venv .venv  
source .venv/bin/activate  
pip install -r requirements.txt  
uvicorn main:app --reload --port 8000
```

6. Start the frontend (React + Vite):

```
cd spa  
npm install  
npm run dev -- --port 5173
```

7. Open the UI: <http://localhost:5173> and connect to the backend API at <http://localhost:8000>.

## 4. SWAGGER

### 4.1. How to use swagger

The backend API is documented using OpenAPI (Swagger). You can access the interactive documentation at <http://localhost:8000/docs>. Here you can test endpoints directly from the browser.

## 5. UI Walkthrough

This section explains the main panels and their common interactions.

1. **Dataset selector** – choose a dataset

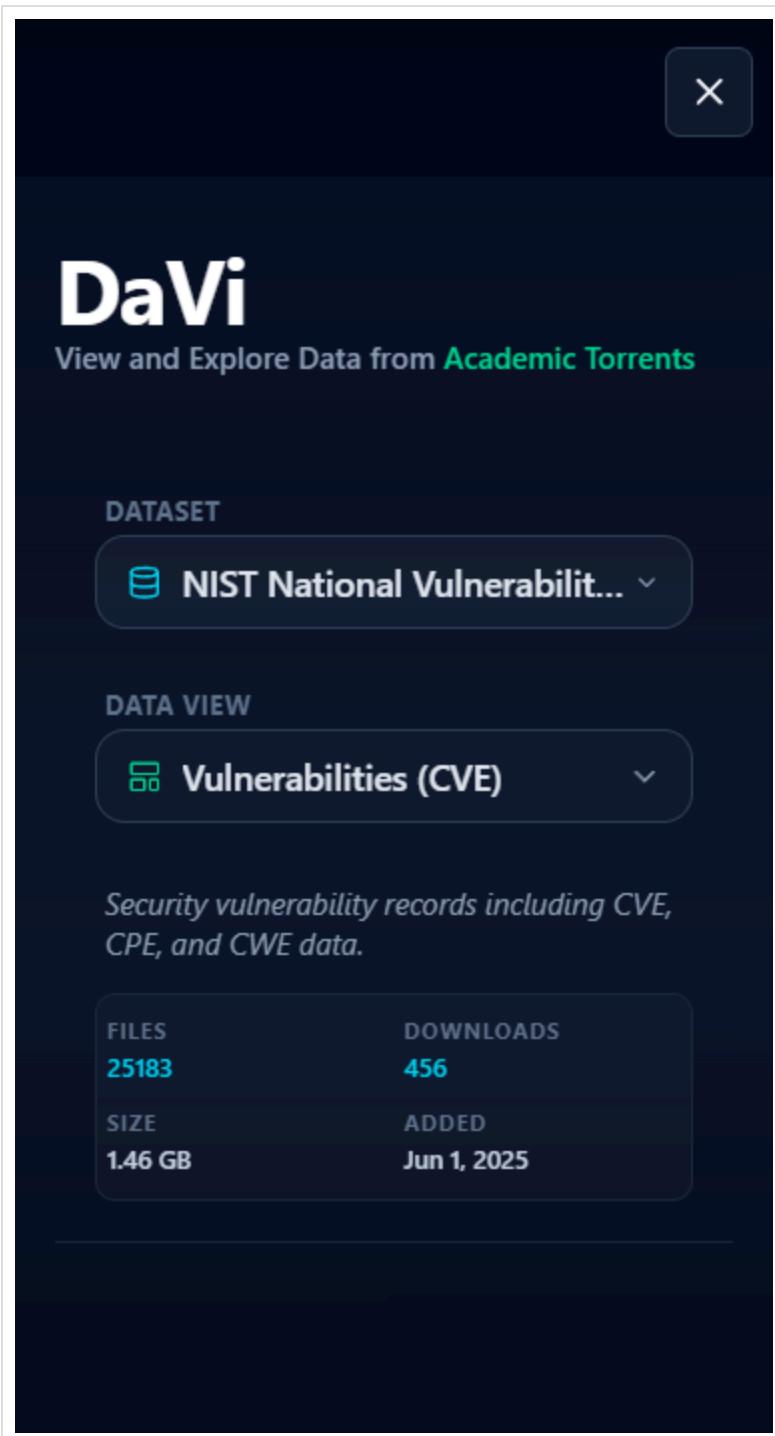


Figure 2 – Sidebar.

2. **Analytics Builder** – dynamically generated controls based on the davi-meta ontology.



Figure 3 – Analytics Builder.

3. **Graph** – expand nodes, focus, and inspect triples; double-click to open the resource detail panel.

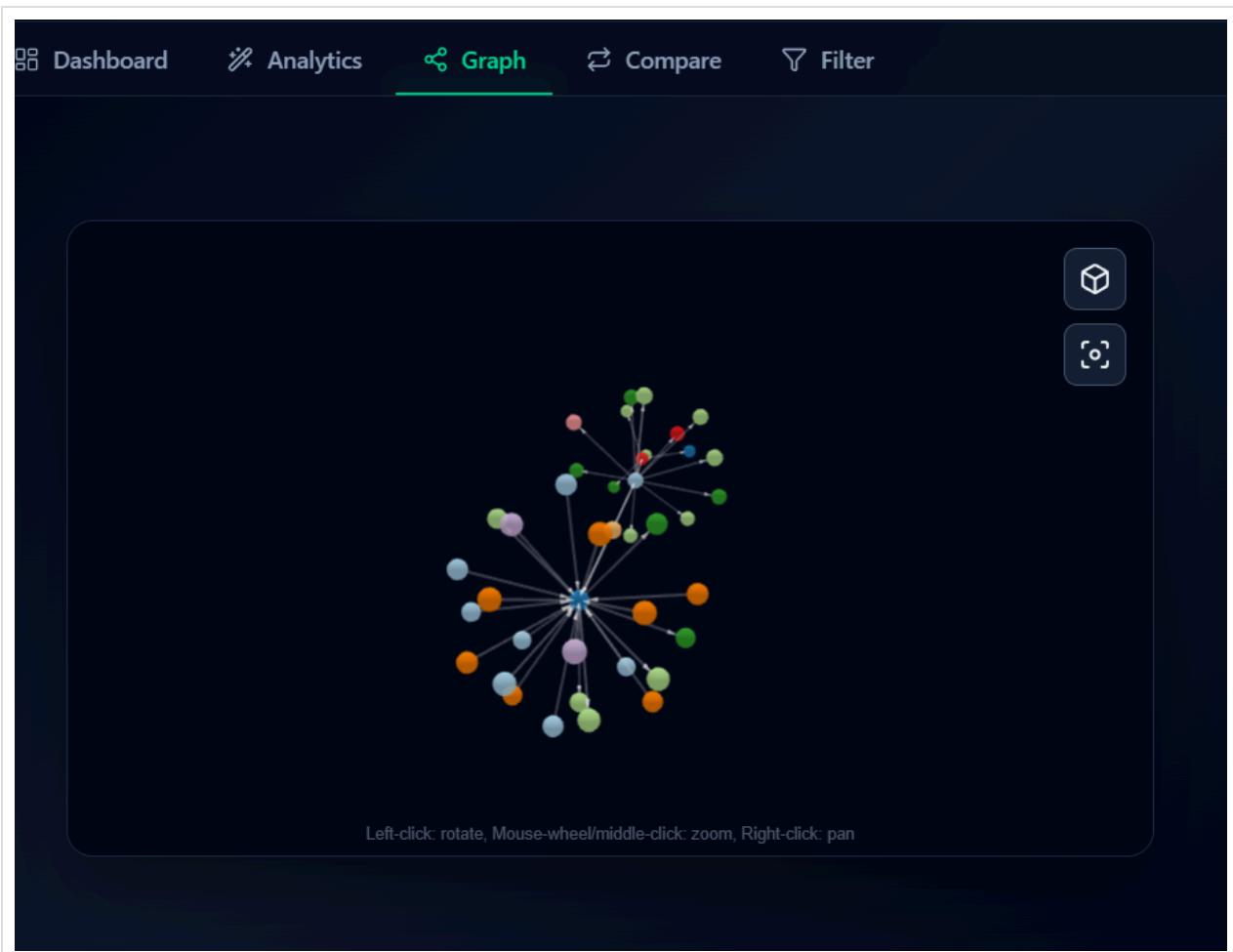


Figure 4 – 3D Graph Builder.

#### 4. Filter Data

The screenshot shows the Semantic Data Explorer interface for the MovieLens 20M Dataset, specifically the 'Movies' section. The top navigation bar includes 'Dashboard', 'Analytics', 'Graph', 'Compare', and 'Filter' (which is currently active). The main area is titled 'Filter Movies' with a 'Run' button. Two filter conditions are defined:

- Condition 1: `http://schema.org/name` Contains `toy` (Path (Optional))
- Condition 2: `http://schema.org/genre` Contains `toy` (Path (Optional))

A dropdown menu is open over the second condition, showing the following options:

- Contains
- Not contains (highlighted)
- Equals (=)
- Not equals (!=)
- Greater >
- Less <
- Transitive (Tree)

The 'Results (3)' section lists the following items:

- `Toy Story (1995)` (`http://schema.org/Movie`)  
`http://schema.org/name: Toy Story (1995)`
- `Toy Story 2 (1999)` (`http://schema.org/Movie`)  
`http://schema.org/name: Toy Story 2 (1999)`
- `Toy Story 3 (2010)` (`http://schema.org/Movie`)  
`http://schema.org/name: Toy Story 3 (2010)`

At the bottom, it says 'Showing 3 items (Offset: 0)' with 'Previous' and 'Next' buttons.

Figure 5 – Filter Data.

For issues or further customizations, open an issue in the repository or ask the authors.

## 6. References