Guidance Resources for Software Engineering in Geosoftware II -Winter 24/25

Brian Pondi
Christian Knoth

October 24, 2024

Project Focus

This guide is designed to help you understand and implement the necessary resources for developing a **Web Catalogue for user-friendly search and retrieval of machine learning models for EO (Earth Observation) datacubes**. The project involves both frontend and backend engineering, including database management and containerization. Below are curated resources and explanations to support your learning and development process.

1 Programming Resources

1.1 Frontend Engineering

Frontend engineering involves building the user interface (UI) and creating an engaging user experience (UX). The goal is to ensure a seamless interaction between the user and the application. You can choose from various tools and frameworks for this purpose:

- **Core Technologies**: If you're looking for a more hands-on approach that is also easier, consider starting with core web technologies like HTML, CSS, and JavaScript. These provide a solid foundation for building web interfaces.
- **Frameworks**: Modern frontend frameworks like ReactJS and Angular can significantly improve productivity, maintainability, and scalability. If you're a beginner but already have knowledge of Core Technologies(HTML, CSS, JavaScript), ReactJS might be a good starting point. Angular, while powerful, is more complex and may require prior experience.

Recommended Learning Resources:

1. HTML and CSS Basics - https://www.youtube.com/watch?v=G3e-cpL7ofc&t=62s
2. JavaScript Crash Course - https://www.youtube.com/watch?v=PkZNo7MFNFg
3. ReactJS Tutorial for Beginners - https://www.youtube.com/watch?v=LDB4uaJ87e0
4. Angular Crash Course - https://www.youtube.com/watch?v=3dHNOWTI7H8
5. Typescript Overview (Superset of JavaScript) - https://www.youtube.com/watch?v=d56mG7DezGs

1.1.1 Map Visualization for Geospatial Data

For the geospatial aspect of this project, you will need libraries capable of rendering interactive maps. These include:

- **LeafletJS**: Lightweight and beginner-friendly for adding interactive maps.
- **OpenLayers**: More advanced (but more complex!!!), with support for complex mapping applications.

1.2 Backend Engineering

Backend engineering involves setting up the server, managing databases, and creating APIs that allow the frontend to communicate with the backend. The following are key areas you should focus on.

1.2.1 REST API Design

Understanding REST (Representational State Transfer) is fundamental for backend development. REST APIs allow the frontend to communicate with the backend by sending requests and receiving data, often in the form of JSON.

Recommended Learning Resource:

- REST API Crash Course - https://www.youtube.com/watch?v=qbLc5a9jdXo&t=1085s

1.2.2 Backend with JavaScript (Node.js)

JavaScript is not limited to the frontend. You can also use it on the backend with Node.js, which is a runtime environment that allows you to run JavaScript on the server. Express.js is a popular framework for building REST APIs with Node.js.

Recommended Learning Resources:

- Node.js Basics - https://www.youtube.com/watch?v=32M1al-Y6Ag
- Express.js Tutorial - https://www.youtube.com/watch?v=L72fhGm1tfE

1.2.3 Backend with Python

Python is another powerful option for backend development. It offers several frameworks, such as:

- **Flask**: Lightweight and flexible for small to medium-sized applications.
- **Django**: Feature-rich with built-in admin interfaces and authentication systems, ideal for larger applications. (!!! Can be overwhelming complex for beginners to understand the whys of things)
- **FastAPI**: Modern and efficient, particularly useful when performance is a priority.

NB: Look into SQLAlchemy - https://www.sqlalchemy.org/

1.2.4 Database Management

For database management, PostgreSQL is highly recommended, especially for geospatial applications because it supports PostGIS, an extension that enables advanced geospatial querying. Other options include MySQL and SQLite, which may be more suitable for smaller projects.

1.2.5 Database Interaction Client

For interacting with databases, it is recommended to use the DBeaver Community Edition, an open-source database management tool that supports various database types, including PostgreSQL, MySQL, and SQLite. DBeaver provides a user-friendly interface for database management, querying, and visualization.

- **DBeaver** - https://dbeaver.io/

1.2.6 Tools for Testing REST APIs

Testing your APIs is essential for ensuring that they work as intended. Postman and Insomnia are popular tools for sending requests to your API endpoints and validating the responses.

Recommended Tools:

- Postman - https://www.postman.com/
- Insomnia - https://insomnia.rest/

1.3 Containerization and Collaborative Programming

1.3.1 Containerization with Docker

Containerization allows you to package your application with all its dependencies, ensuring that it runs consistently across different environments. Docker is the leading tool for this, and learning how to use Docker will streamline both development and deployment.

Recommended Learning Resource:

- Docker Tutorial for Beginners - https://www.youtube.com/watch?v=pg19Z8LL06w

1.3.2 Collaborative Development with Git and GitHub

Collaboration is a crucial part of software development. Git allows version control, and GitHub facilitates collaborative development by enabling pull requests, code reviews, and issue tracking.

Key Concepts:

- **Git Commands**: Learn basic commands like `git clone`, `git commit`, `git push`, and `git pull`.
- **GitHub**: Explore how to create pull requests, manage branches, resolve conflicts, and merge changes.

NB: There are many YouTube tutorials for this.
Recommended Learning Resource:

- Git and GitHub Crash Course - https://www.youtube.com/watch?v=tRZGeaHPoaw

2 SpatioTemporal Asset Catalog (STAC) Ecosystem

The SpatioTemporal Asset Catalog (STAC) specification provides a standardized language for describing geospatial data, making it easier to work with, index, and discover. It allows for the efficient organization of spatiotemporal assets, such as satellite imagery and Earth Observation (EO) data.[1]

2.1 STAC-MLM Extension

The STAC-MLM (Machine Learning Models) extension is designed to represent metadata related to machine learning models, particularly those trained or applied to spatiotemporal data. This extension defines how models and their parameters are described, making them discoverable within STAC catalogs.

For details on implementing the necessary filter parameters, refer to the **ReadMe** in the GitHub repository.

- Paper: https://doi.org/10.1145/3681769.3698586
- GitHub: https://github.com/stac-extensions/mlm

2.2 STAC Node Server

The STAC Node Server is a Node.js based implementation of the STAC API, designed for serverless environments like AWS Lambda. It also supports integration with OpenSearch for enhanced search functionality.

- STAC Node Server - https://github.com/stac-utils/stac-server

[1] https://stacspec.org/en

2.3 STAC FastAPI

STAC FastAPI is a Python implementation of the STAC API using the FastAPI framework. It provides a fast and efficient way to deploy a STAC-compliant API, enabling users to search, retrieve, and manage spatiotemporal assets in real time.

- STAC FastAPI - https://github.com/stac-utils/stac-fastapi

2.4 STAC with PostgreSQL

STAC-Postgres integrates STAC metadata storage and querying with PostgreSQL, making it an ideal solution for managing large-scale catalogs. Combining it with PostGIS allows advanced geospatial queries, enhancing the ability to search and analyze spatiotemporal data.

- STAC FastAPI with PostgreSQL (stac-fastapi-pgstac) - https://github.com/stac-utils/stac-fastapi-pgstac
- Docker Compose file for STAC FastAPI PostgreSQL - https://tinyurl.com/4sut3naf

2.5 STAC Clients

STAC clients are libraries and tools that enable users to interact with STAC-compliant APIs and catalogs. These clients support queries for specific spatiotemporal ranges and facilitate the download of geospatial assets, such as satellite imagery. Notable clients include **PySTAC** for Python and **RSTAC** for R.

Here's an example of Terradue using PySTAC and the STAC MLM extension:

- Blog - https://tinyurl.com/34j6nrmc

2.6 STAC Examples of Machine Learning Models Metadata

Below are examples of machine learning models that include metadata using the STAC-MLM extension:

Examples:

- Water Bodies Model - https://ai-extensions-stac.terradue.com/collections/ML-Models/items
- Landcover Eurosat Sentinel-2 Model - https://huggingface.co/wherobots/mlm-stac/blob/main/classification/landcover-eurosat-sentinel2/model-metadata.json
- Solar Satlas Sentinel-2 Model - https://huggingface.co/wherobots/mlm-stac/blob/main/semantic-segmentation/solar-satlas-sentinel2/model-metadata.json

2.7 STAC Utilities

A comprehensive collection of tools for working with SpatioTemporal Asset Catalogs (STAC) can be found on the official STAC Utilities GitHub repository. These tools simplify creating, validating, and searching STAC catalogs.

- STAC Utilities - https://github.com/stac-utils

Conclusion

These resources are just recommendations to guide your implementation. Feel free to design a software architecture that meets your specific project requirements, as long as it adheres to the requirements.