***Technical Vision: PHAROS***

**Document ID:** PHAROS-DOC-TECHNICAL\_VISION  
**Version:** 0.0.1  
**Status:** Draft  
**Author:** Alexandros P. Liaskos  
**Last Updated:** 16/9/2025

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Summary of Changes** |
| 0.0.1 | 16/9/2025 | Alexandros P. Liaskos | Initial draft of the document. |

**Implementation Strategy**

PHAROS will follow a modular architecture where each component will be developed as a standalone tool, ensuring maximum flexibility and reusability. The entire suite will be managed within a dedicated GitHub Organization to provide a centralized home for all components.

All source code will be maintained in public repositories within this organization under open-source licensing. Each component will have its own repository, allowing for independent development, issue tracking, and versioning.

**Component Architecture**

The PHAROS suite supports multiple implementation forms (software types) to accommodate different use cases:

* **CLI Applications:** Self-contained command-line tools built with Python or JavaScript/TypeScript. Distributed via PyPI or npm.
* **Libraries:** Reusable Python or JavaScript/TypeScript code packages. Distributed via PyPI or npm.
* **APIs:** Network-accessible REST endpoints built with Nest.js (TypeScript) and deployed as Docker containers.
* **Web Applications:** Full-stack browser-based interfaces using Next.js frontend with shadcn/ui components, Nest.js backend, and Python microservices for data processing.
* **Static Web Applications:** Client-side applications running entirely in the browser, using Next.js with IndexedDB for local data storage and deployed via GitHub Pages.

**Software Type Keys**

|  |  |  |
| --- | --- | --- |
| **Key/Value** | **Name** | **Description** |
| CLI | Command-Line Interface | A standalone executable program that runs in a terminal or command prompt. |
| LIB | Library | A collection of reusable code (functions, classes, modules) intended to be used by other independent programs. |
| API | Application Programming Interface | A backend service component that exposes data and functionality through defined endpoints for other applications to consume. |
| WEB | Web Application | A full stack server/client-side web application. |
| SPA | Single-Page Application | A serverless client-side web application. |