Minard Architecture

System Overview

```
┌──────────────────────────────────────────────────────────────────────────
┐
│                              MINARD
│
├──────────────────────────────────────────────────────────────────────────
┤
│
│
│  ┌──────────────┐   ┌──────────────┐   ┌──────────────────────────────┐
│  │   Frontend   │   │Site Explorer│   │      VS Code Extension       │
│  │  (Halogen)   │   │  (Halogen)   │   │        (TypeScript)          │
│  └──────┬───────┘   └──────┬───────┘   └──────────────┬───────────────┘
│         │                  │                          │
│         └────────┬─────────┴──────────────────────────┘
│                  │
│                  ▼
│         ┌────────────────┐
│         │     Server     │  REST API
│         │  (HTTPurple)   │  /api/v2/*
│         └───────┬────────┘
│                 │
│                 ▼
│         ┌────────────────┐
│         │    Database    │  DuckDB
│         │ (ce-unified)   │  Unified Schema v3
│         └───────┬────────┘
│                 ▲
│
```

```
│                 │
│                 │
│        ┌────────┴────────┐
│        │     Loader      │  Data Ingestion
│        │  (Node.js →     │  spago, git, docs.json, corefn.json
│        │   Rust/Go)      │
│        └─────────────────┘
│
│
└──────────────────────────────────────────────────────────────┘
```

Data Flow

1. Ingestion (Loader → Database)

```
Source Files          Compilation Artifacts     Version Control
     │                        │                         │
     ▼                        ▼                         ▼
┌──────────┐          ┌──────────────┐          ┌──────────┐
│ *.purs   │          │  docs.json   │          │ git log  │
│ imports  │          │ corefn.json  │          │ git diff │
└────┬─────┘          │ spago.lock   │          └────┬─────┘
     │                └──────┬───────┘               │
     │                       │                       │
     └───────────────────────┼───────────────────────┘
             │
             │
             ▼
        ┌──────────────┐
        │    Loader    │
        │              │
        │ • Parse JSON │
        │ • Build graphs│
        │ • Compute metrics│
        │ • Generate SQL│
        └──────┬───────┘
             │
             │
             ▼
        ┌──────────────┐
        │   DuckDB     │
        │              │
        │ • packages   │
        │ • modules    │
        │ • declarations│
        │ • imports    │
        │ • calls      │
```

```
              │ • git metrics│
              └──────────────┘
```

Data Sources:

Source	What We Extract	Tables Populated
`spago.lock`	Package versions, dependencies	`package_versions`, `package_dependencies`
`output/*/docs.json`	Module names, declarations, types	`modules`, `declarations`, `child_declarations`
`output/*/corefn.json`	Function calls, imports	`function_calls`, `module_imports`
`git log`	Commits, authors, dates	`commits`, `module_commits`
`*.purs` (src)	LOC, raw source	`modules.loc`, `declarations.source_code`

2. Query (Server → Database)

The server provides a REST API that translates HTTP requests to SQL queries:

```
GET /api/v2/packages
    ↓
SELECT * FROM package_versions WHERE source IN ('workspace', 'extra')
    ↓
JSON response with moduleCount, declarationCount computed via subquery
```

Key API Patterns:

- **List endpoints** return summary data (counts, not full objects)
- **Detail endpoints** return nested data (package → modules → declarations)
- **Bulk endpoints** (`/all-imports`) return denormalized data for graph building
- **Search endpoints** support type signature queries

3. Visualization (Frontend ← Server)

```
┌─────────────────────────────────────────────────────────────────┐
│                            Frontend                             │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│                                                                 │
│  ┌───────────────────────────────────────────────────────────┐  │
│  │                         AppShell                          │  │
│  │ • Project selector                                        │  │
│  │ • Scene coordinator                                       │  │
│  │ • Theme/settings                                          │  │
│  └─────────────────────────┬─────────────────────────────────┘  │
│
```

```
│                               │                                 │
│                 ┌─────────────┼─────────────┐                   │
│                 ▼             ▼             ▼                   │
│  ┌────────────┐   ┌────────────┐   ┌────────────┐   ...         │
│  │ ForceGraph │   │  Treemap   │   │  Beeswarm  │               │
│  │   Scene    │   │   Scene    │   │   Scene    │               │
│  └────────────┘   └────────────┘   └────────────┘               │
│                                                                 │
│  Built with Hylograph (HATS):                                   │
│  • Declarative visualization trees                              │
│  • D3 for layout algorithms                                     │
│  • HATS interpreter for DOM operations                          │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Scene Architecture:

Each visualization scene follows this pattern:

1. **Data Loading**: Fetch from server, transform to visualization model
2. **Layout Computation**: Use D3 algorithms (force, hierarchy, etc.)
3. **HATS Tree Building**: Declarative specification of SVG structure
4. **Rendering**: HATS interpreter produces DOM operations
5. **Interaction**: Event handlers update state, trigger re-render

4. Site Explorer Data Flow

```
┌───────────────────┐      ┌───────────────────┐
│  Static Analysis  │      │ Dynamic Spidering │
│                   │      │                   │
│  • Parse Route    │      │  • Puppeteer      │
│    ADT            │      │  • BFS from /     │
│  • Extract        │      │  • Follow #links  │
│    mappings       │      │  • Record depth   │
└─────────┬─────────┘      └─────────┬─────────┘
          │                          │
          └────────────┬─────────────┘
                       │
                       ▼
          ┌────────────────────────┐
          │       Comparison       │
          │                        │
          │  • Defined routes      │
          │  • Reachable routes    │
          │  • Unreachable         │
          │  • Extra (orphan)      │
          └────────────┬───────────┘
                       │
                       ▼
          ┌────────────────────────┐
          │      Database          │
```

```
                  │                        │
                  │  • routes              │
                  │  • spider_runs         │
                  │  • discovered_pages    │
                  │  • annotations         │
                  └────────────────────────┘
```

Database Schema (Simplified)

```
┌─────────────────┐      ┌─────────────────┐      ┌─────────────────┐
│ package_versions│────<│     modules     │────<│  declarations   │
│                 │      │                 │      │                 │
│  • name         │      │  • name         │      │  • name         │
│  • version      │      │  • path         │      │  • kind         │
│  • source       │      │  • loc          │      │  • type_signature│
│  • description  │      │  • namespace_id │      │  • type_ast     │
└─────────────────┘      └────────┬────────┘      └─────────────────┘
                                  │
                     ┌────────────┼────────────┐
                     ▼            ▼            ▼
          ┌──────────────┐ ┌──────────────┐ ┌──────────────┐
          │module_imports│ │function_calls│ │module_commits│
          │              │ │              │ │              │
          │  • from_id   │ │  • caller    │ │  • commit_id │
          │  • to_name   │ │  • callee    │ │  • module_id │
          └──────────────┘ └──────────────┘ └──────────────┘
```

Design Principles:

1. **Package versions are identity**: `halogen@7.0.0` ≠ `halogen@8.0.0`
2. **spago.lock is truth**: Versions come from lock file, not guessing
3. **Types are first-class**: Both rendered (`String -> Int`) and AST (queryable JSON)
4. **Namespaces independent**: `Data.Array` namespace exists even without that module

Technology Stack

Component	Technology	Why
Frontend	PureScript + Halogen	Type safety, declarative UI
Visualization	Hylograph (HATS)	Declarative, composable
Server	PureScript + HTTPurple	Type-safe HTTP, same language as frontend
Database	DuckDB	Fast analytics, embedded, SQL
Loader	Node.js (→ Rust/Go)	Easy start, performance rewrite planned
Site Spider	Puppeteer	Headless Chrome, reliable SPA crawling
VS Code	TypeScript	Required by VS Code extension API

Deployment

```
┌──────────────────────────────────────────────────────────────┐
│                        Docker Compose                        │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│  ┌─────────────┐     ┌─────────────┐     ┌─────────────┐     │
│  │ ce-frontend │     │ ce-backend  │     │    edge     │     │
│  │   (nginx)   │     │   (node)    │     │ (openresty) │     │
│  │             │     │             │     │             │     │
│  │ /code/*.html│◄────│ :3000/api/* │◄────│    :80/*    │     │
│  │ /code/*.js  │     │             │     │             │     │
│  └─────────────┘     └─────────────┘     └─────────────┘     │
│                             │                                │
│                             ▼                                │
│                      ┌─────────────┐                         │
│                      │ ce-unified  │                         │
│                      │   .duckdb   │                         │
│                      │ (baked in)  │                         │
│                      └─────────────┘                         │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

Edge Router:

- OpenResty (nginx + Lua)
- Routes `/code/api/*` → ce-backend
- Routes `/code/*` → ce-frontend
- URL rewriting: strips `/code` prefix

Future: Loader Rewrite

The current loader (`ce-loader.js`, ~800 lines Node.js) works but is slow. Plan:

```
┌──────────────────────────────────────────────────────────────────┐
│                       minard-loader (Rust/Go)                    │
├──────────────────────────────────────────────────────────────────┤
│                                                                  │
│  Inputs:                                                         │
│  • Path to PureScript project                                    │
│  • Path to DuckDB database                                       │
│  • Options: --incremental, --project-name, --snapshot-label      │
│                                                                  │
│  Responsibilities:                                               │
│  1. Discover files (*.purs, output/*/docs.json, etc.)            │
│  2. Parse JSON (docs.json, corefn.json, spago.lock)              │
│  3. Build in-memory graphs (imports, calls)                      │
│  4. Compute derived metrics (coupling, reachability)             │
│  5. Generate SQL inserts                                         │
│  6. Execute against DuckDB                                       │
│                                                                    │
│  Performance target: <5s for 1000 modules (currently ~30s)         │
│                                                                    │
│  Key constraint: Run on every compile without friction             │
│                                                                    │
└───────────────────────────────────────────────────────────────────┘
```

See `docs/LOADER-SPEC.md` for full specification.