Postal

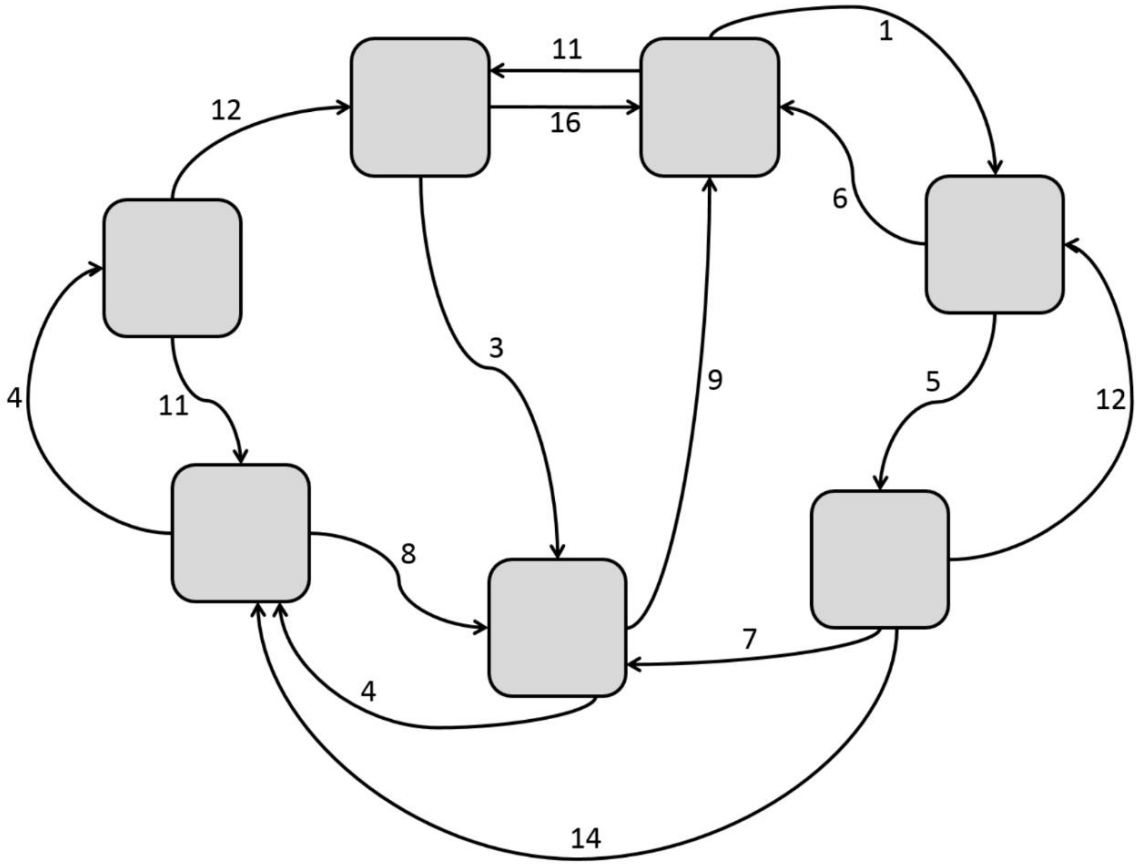
A Visual Studio Code extension to help web developers write better code

Postal

- Visualize Code bases
- In the past the Focus of the project was on web project supporting HTML, CSS, PHP and JavaScript
- We have now expanded the project to be able to handle almost any language as the user defines it.
- Errors

IFT Introduction

- Information Foraging Theory is the concept that our project revolves around
- Information Foraging Theory turns information into topologies



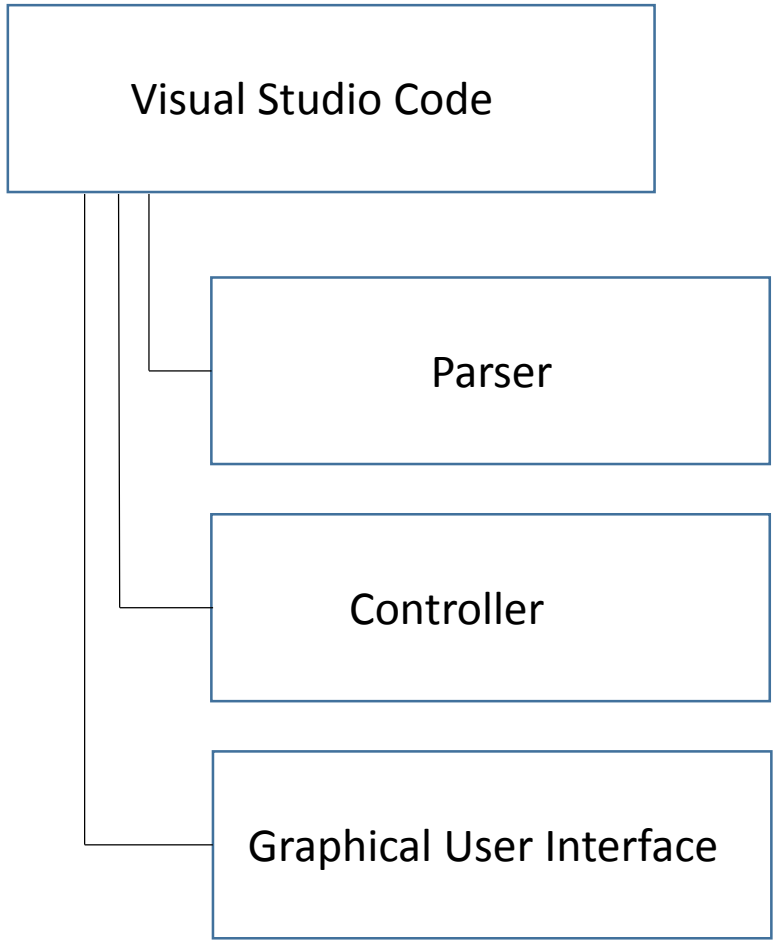
Design details

Visual Studio Code

- Free, cross platform, open-source text editor
- Potential support for any language through extensions
- Extension marketplace offers unlimited functionality expansion

Major Components

- The Major Parts of the Extension:
  - IDE
  - Controller
  - Parser
  - Data Structure
  - Graphical User Interface

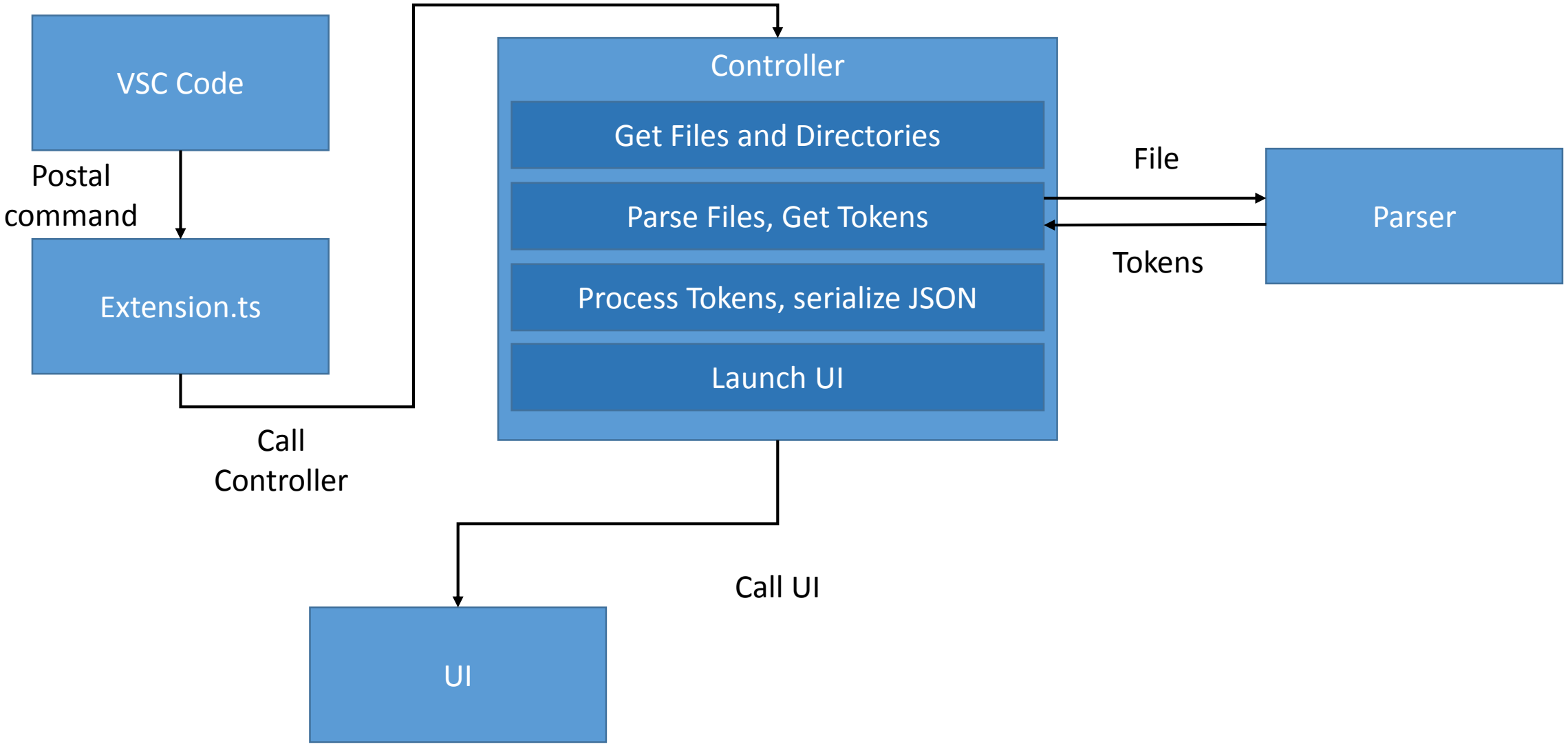


IDE: Visual Studio Code

- The Extension will work with VSCode using Event Handlers
- The Current Events:
  - On extension activate
- Planned Events to be implemented
  - On Click error box in the GUI

Controller

- Manage subcomponents (Parser, UI, etc)
- On Extension Activation:
  - Recursively retrieve a list of all Files and Directories in the User’s open project directory.
  - Call the parser for each file.
  - Serialize returned tokens into JSON File
  - Launch the UI


VSC Code
Postal command
Extension.ts
Call Controller
Controller
Get Files and Directories
Parse Files, Get Tokens
Process Tokens, serialize JSON
Launch UI
File
Tokens
Parser
Call UI
UI

Parser

- Finding all useful information for the file it is provided
- What qualifies as useful?
  - Anything the user defines as such in the grammars file
- Return this information as a standardized token:
  - Line Number
  - Token Type – Link or a subNode
  - Type – Div, Class, ect. Null in the case of a Link
  - Value
  - Parent Token

Grammars

- Used to Parse for specific tokens
- Needs a type as the Hard coded parser behavior needs to be different
- Current types: Links and tagged
- Documentation will be made available to users

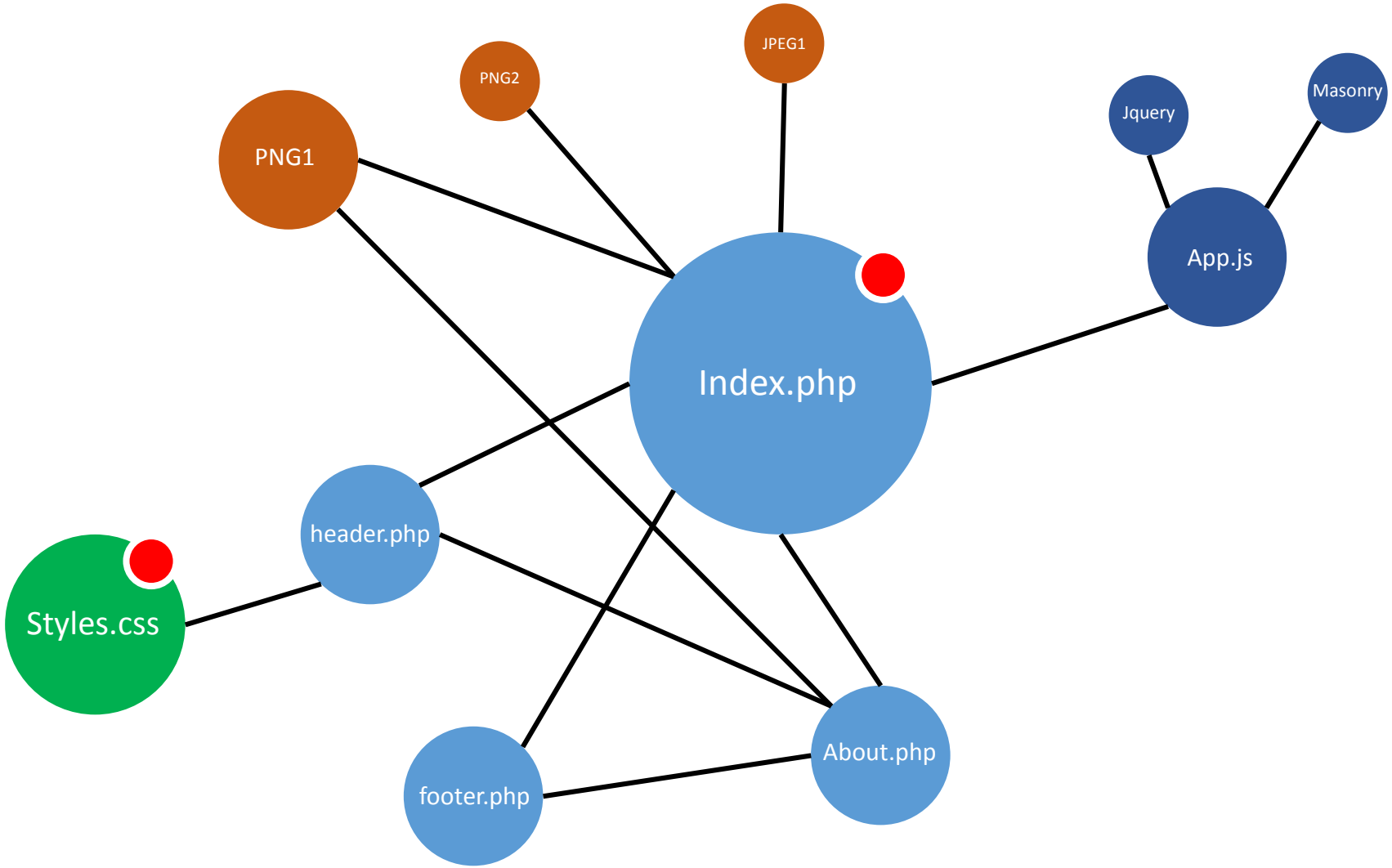
```
{
    "grammars" : [
        {
            "id" : 0,
            "title" : "html",
            "filetypes" : ["html", "php"],
            "rules" : [{
                    "title": "div",
                    "type": "tagged",
                    "options" : {
                        "tagStart": "<div",
                        "namedOption" : "id=\"(.+?)\"",
                        "tagEnd": ">",
                        "closingTag": "</div>",
                        "nodeColor": "blue"
                    }
                }, {
                    "title": "href link",
                    "type" : "link",
                    "options" : {
                        "link": "href=[\"](.+?)[\"]",
                        "nodeColor": "blue"
                    }
                }, {
                    "title": "includes link",
                    "type": "link",
                    "options": {
                        "link": "include=[\"](.+?)[\"]",
                        "nodeColor": "blue"
```

Parsers

- The Parser will now run only on activation of the extension.
- New Grammars can be created relatively quickly.
- Still in need of a few more behaviors (c-like functions)

User Interface -Refresh

- One GUI consisting of the File Map and the Error List.
- Displays all the data collected in the data structure JSON.
- The File Map Displays a visualization of the user's project directory in the form of a graph of interconnected nodes.
- The error list displays a list of the ‘Broken Rules’ in the project directory detected by the extension parser.



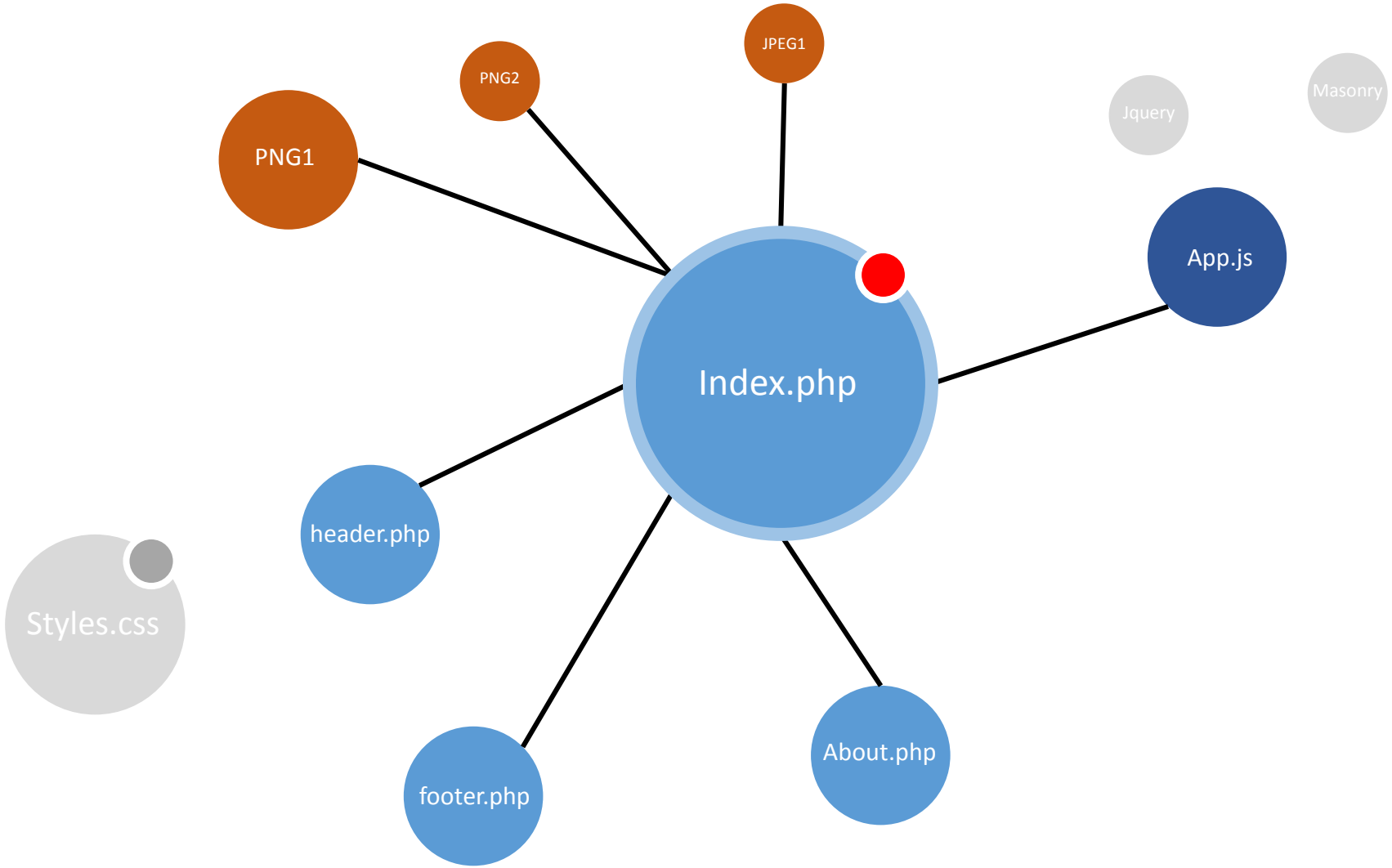
Interdum et malesuada fames ac ante ipsum primis in faucibus. Suspendisse ac efficitur quam. Vestibulum ante

ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nulla id tristique odio. Maecenas tempor justo eu odio mattis efficitur. Donec fermentum lectus nec

orci mollis pretium. Duis tempor sapien sed urna ultricies, ut porttitor ante rutrum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos

himenaeos. Vivamus tempus massa eget ante scelerisque, eu lobortis quam vestibulum. Pellentesque malesuada placerat convallis. Sed vehicula elementum tincidunt. Ut

at enim ac mi congue gravida feugiat non urna. Aliquam gravida, neque eu pellentesque sodales, augue nunc condimentum orci, in commodo urna enim vitae



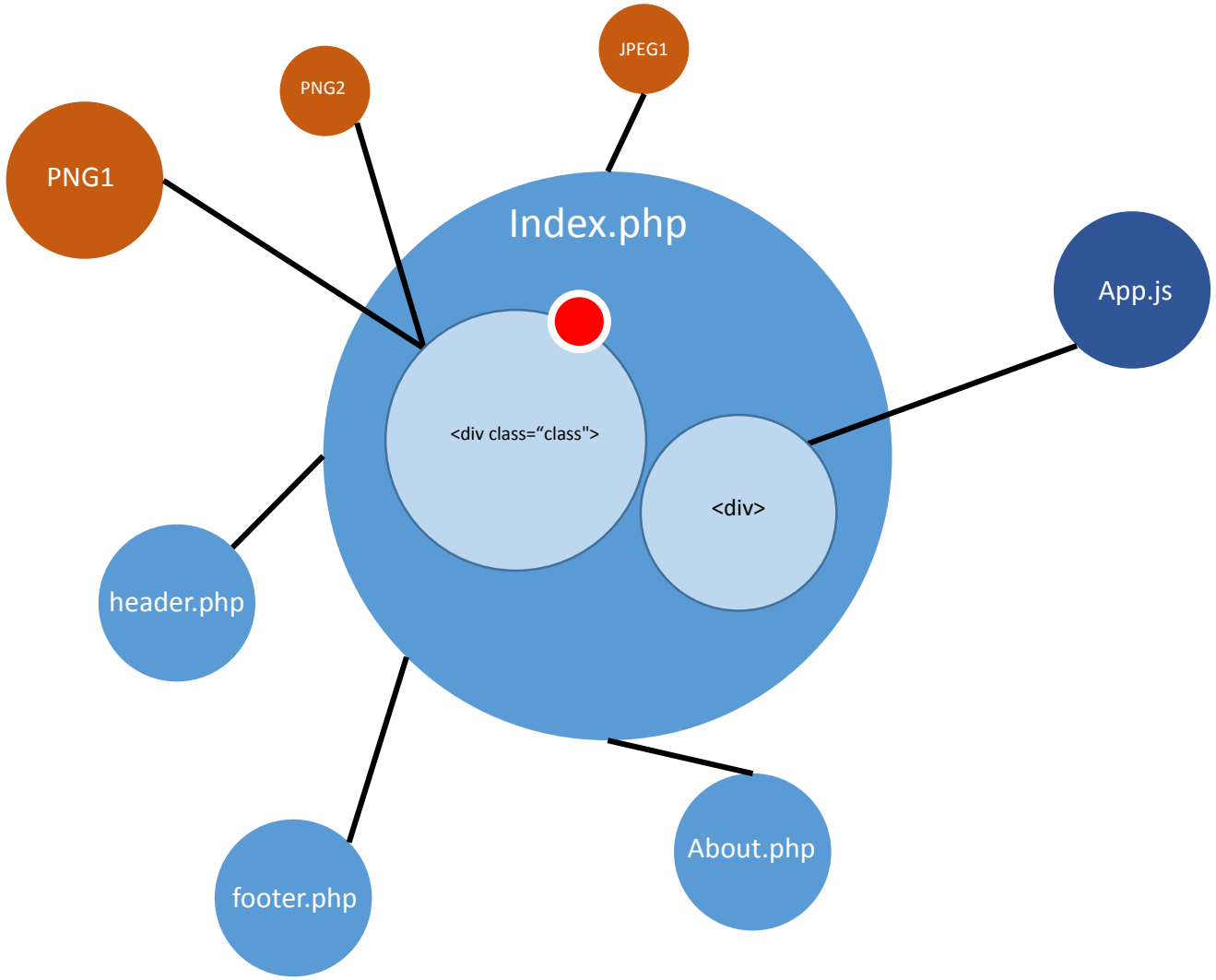
Interdum et malesuada fames ac ante ipsum primis in faucibus. Suspendisse ac efficitur quam. Vestibulum ante

ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nulla id tristique odio. Maecenas tempor justo eu odio mattis efficitur. Donec fermentum lectus nec

orci mollis pretium. Duis tempor sapien sed urna ultricies, ut porttitor ante rutrum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos

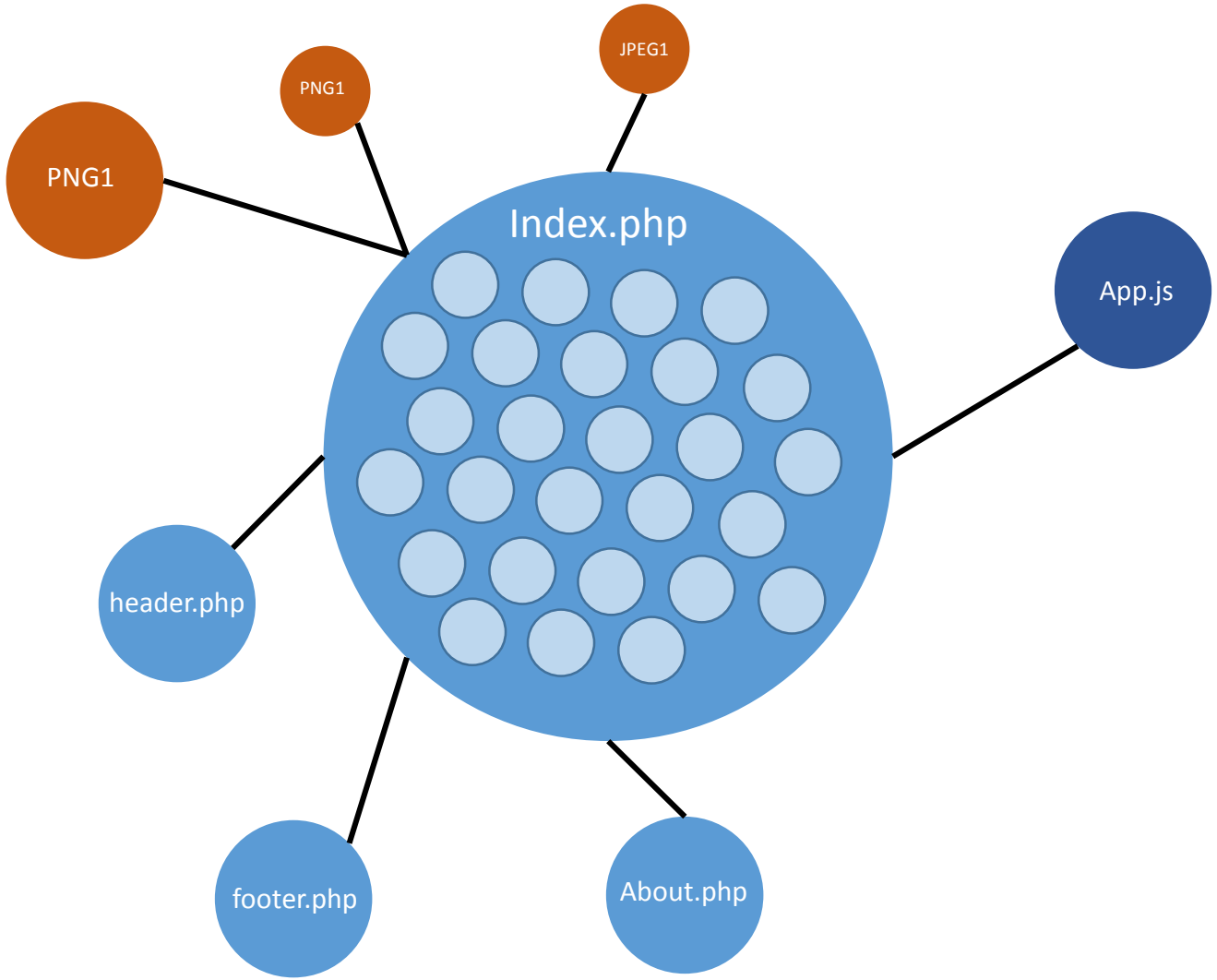
himenaeos. Vivamus tempus massa eget ante scelerisque, eu lobortis quam vestibulum. Pellentesque malesuada placerat convallis. Sed vehicula elementum tincidunt. Ut

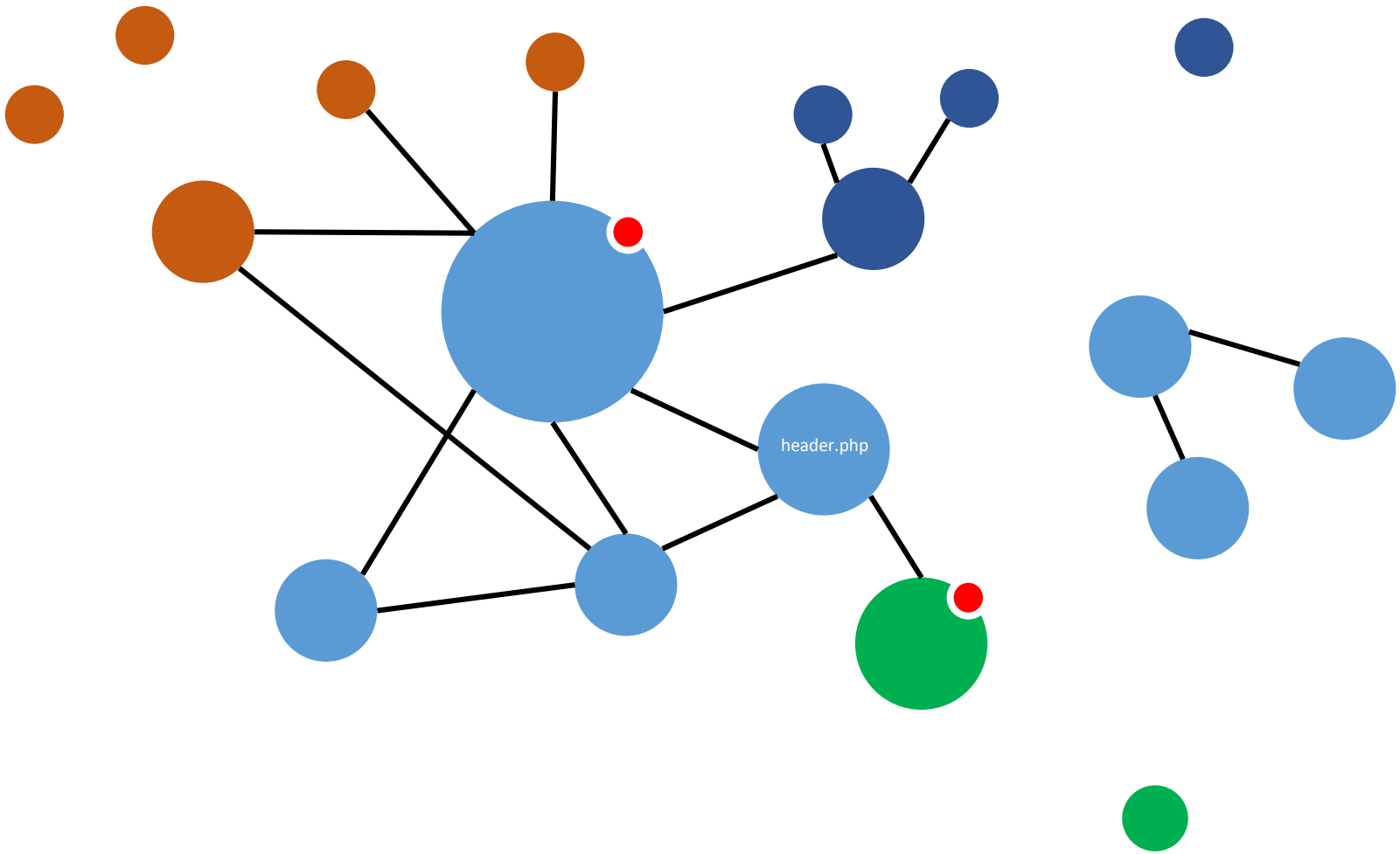
at enim ac mi congue gravida feugiat non urna. Aliquam gravida, neque eu pellentesque sodales, augue nunc condimentum orci, in commodo urna enim vitae



ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nulla id tristique odio. Maecenas tempor justo eu odio mattis efficitur. Donec fermentum lectus nec

orci mollis pretium. Duis tempor sapien sed urna ultricies, ut porttitor ante rutrum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos


JPEG1
PNG1
PNG1
Index.php
App.js
header.php
footer.php
About.php



Interdum et malesuada fames ac ante ipsum primis in faucibus. Suspendisse ac efficitur quam. Vestibulum ante

ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nulla id tristique odio. Maecenas tempor justo eu odio mattis efficitur. Donec fermentum lectus nec

orci mollis pretium. Duis tempor sapien sed urna ultricies, ut porttitor ante rutrum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos

himenaeos. Vivamus tempus massa eget ante scelerisque, eu lobortis quam vestibulum. Pellentesque malesuada placerat convallis. Sed vehicula elementum tincidunt. Ut

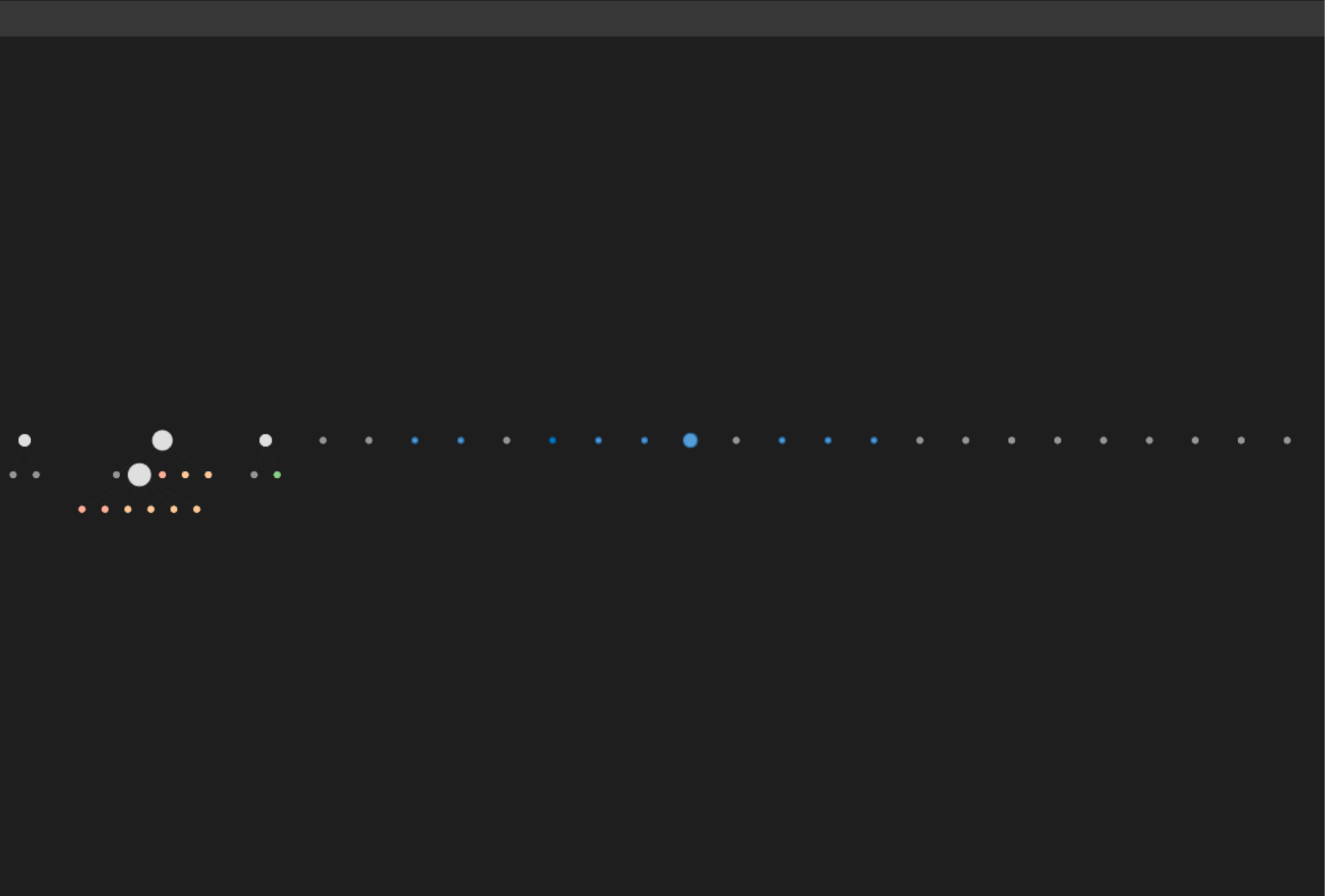
at enim ac mi congue gravida feugiat non urna. Aliquam gravida, neque eu pellentesque sodales, augue nunc condimentum orci, in commodo urna enim vitae

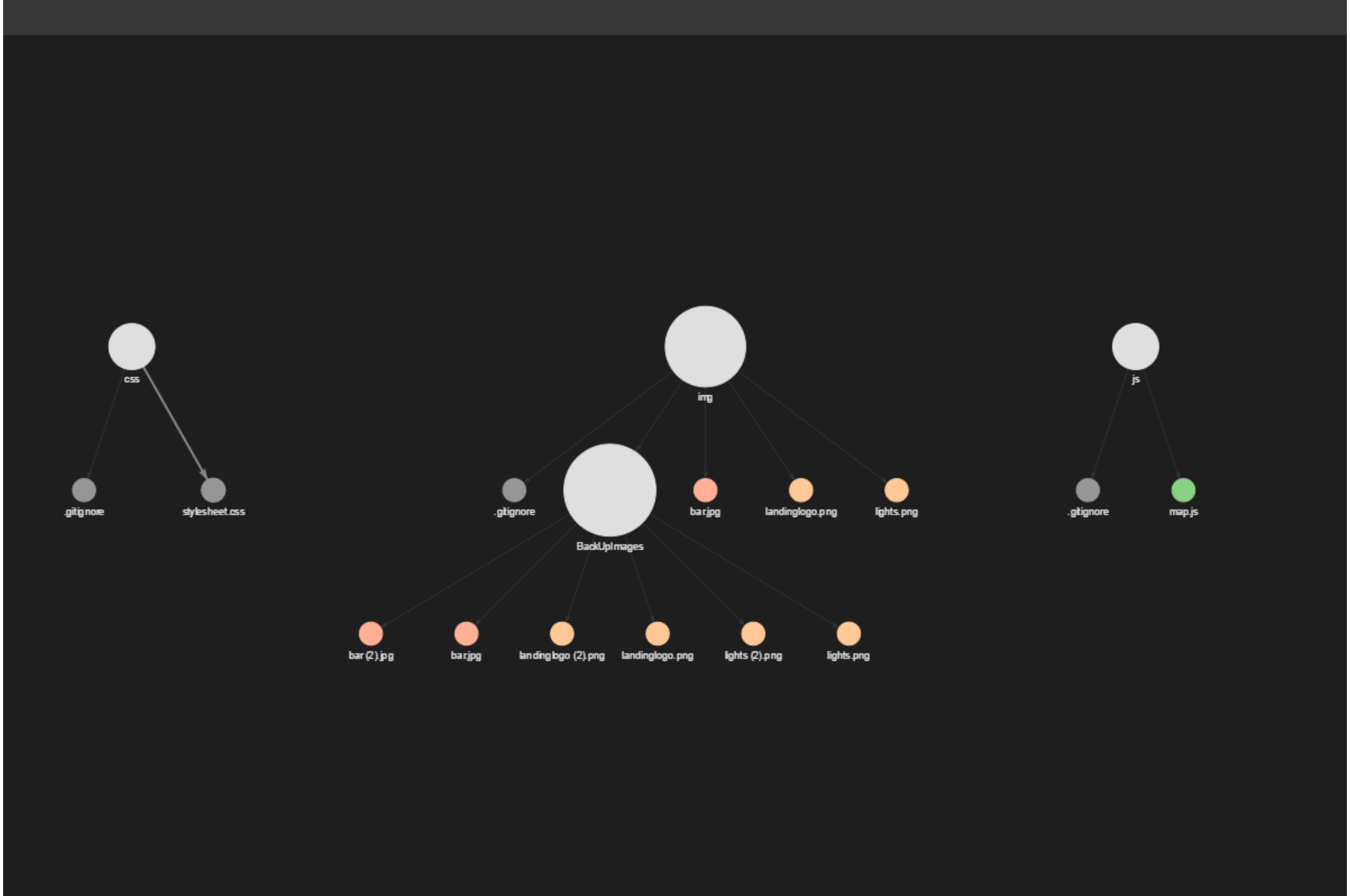
Problems with this approach:

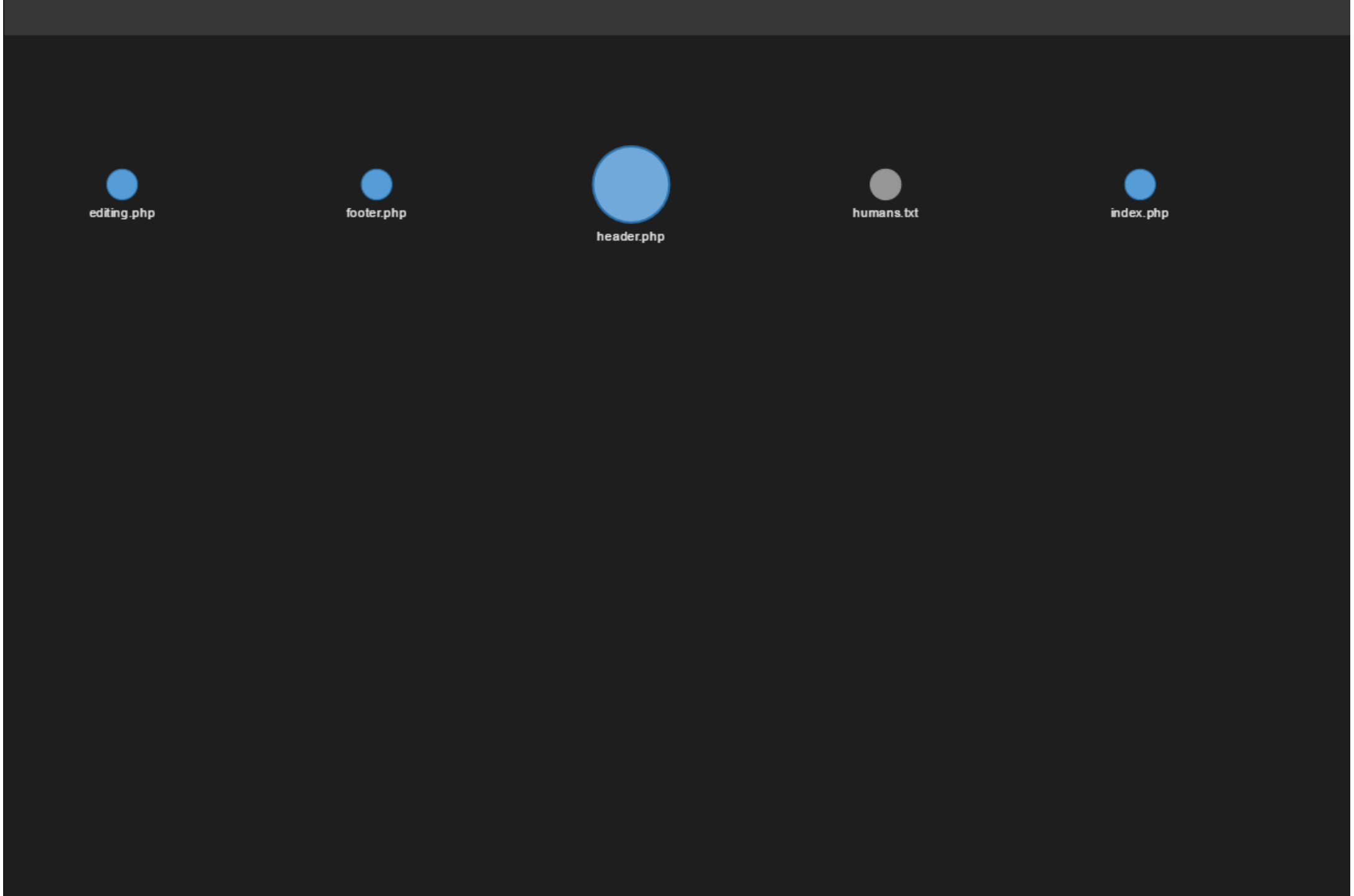
- Large Projects (about 100 files) are overwhelming and don’t provide useful information.
- No way to intuitively display directors.
- Too many links between files. Hard to distinguish what the links mean.

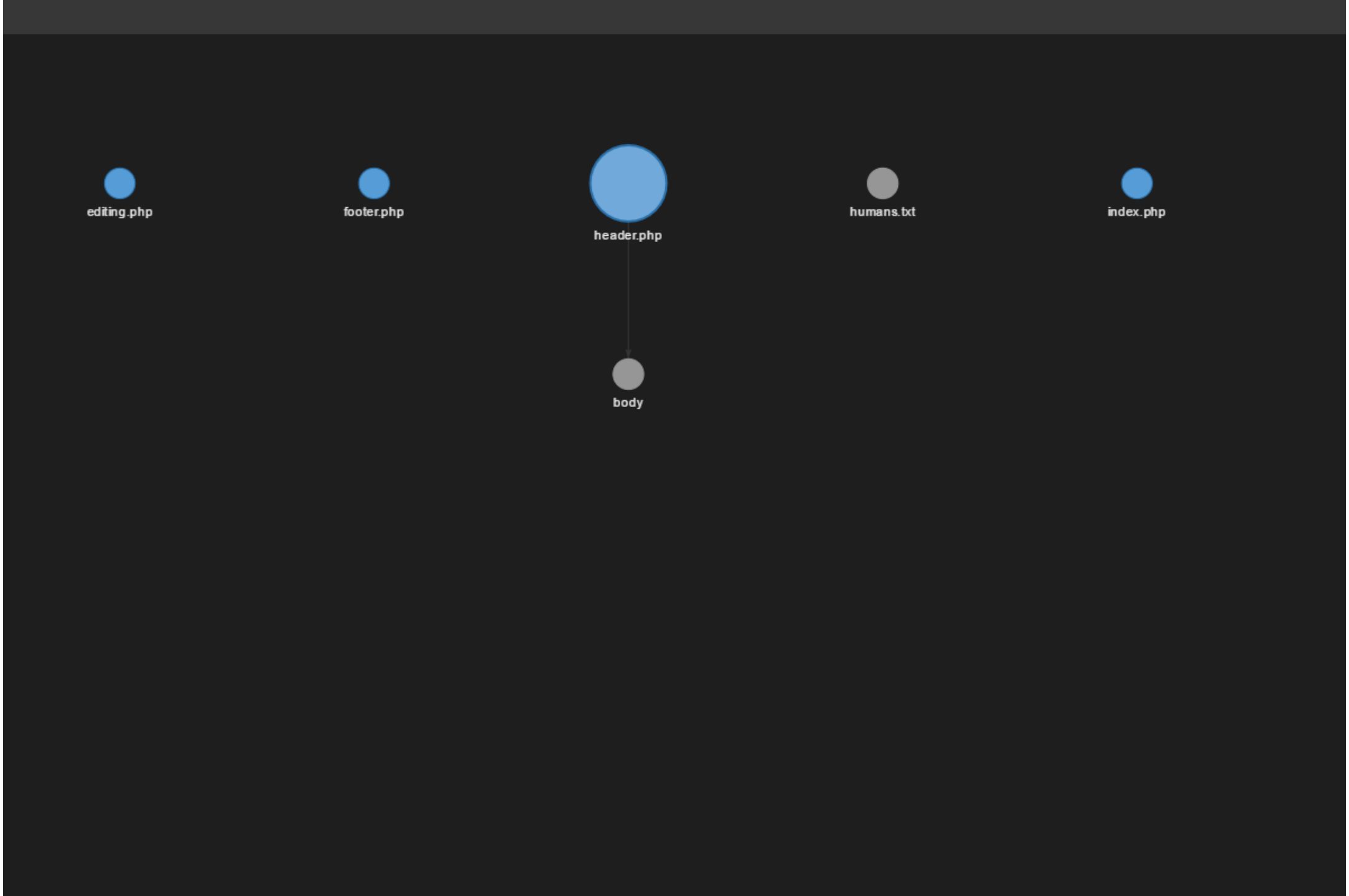
New Approach:

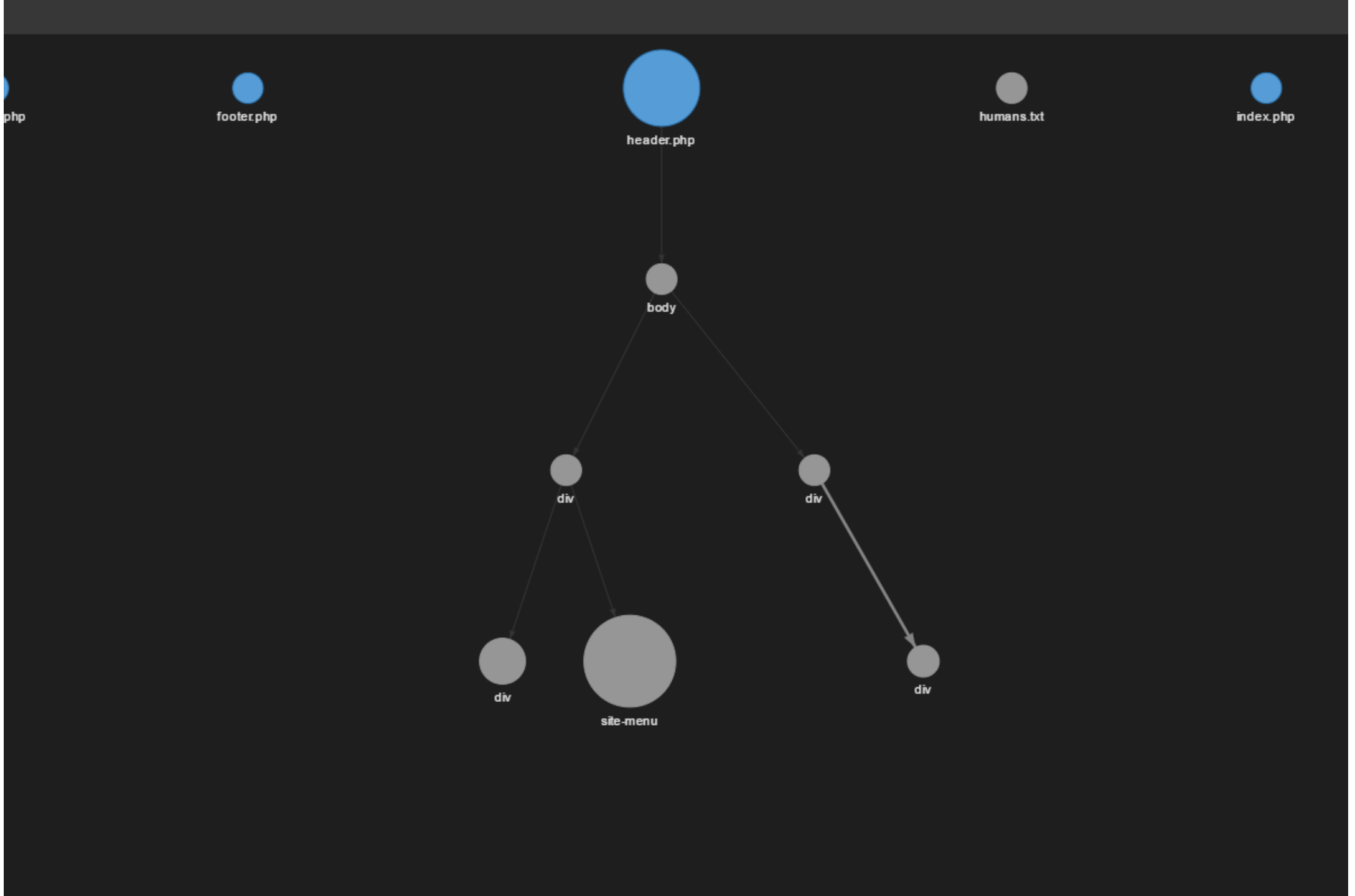
- Render nodes in a hierarchy by file location.
- Show Links between files only when a node is selected, highlight those links in a bright color
- Expanding the nodes into subcomponents produces two new nodes below the parent, maintaining the hierarchy.

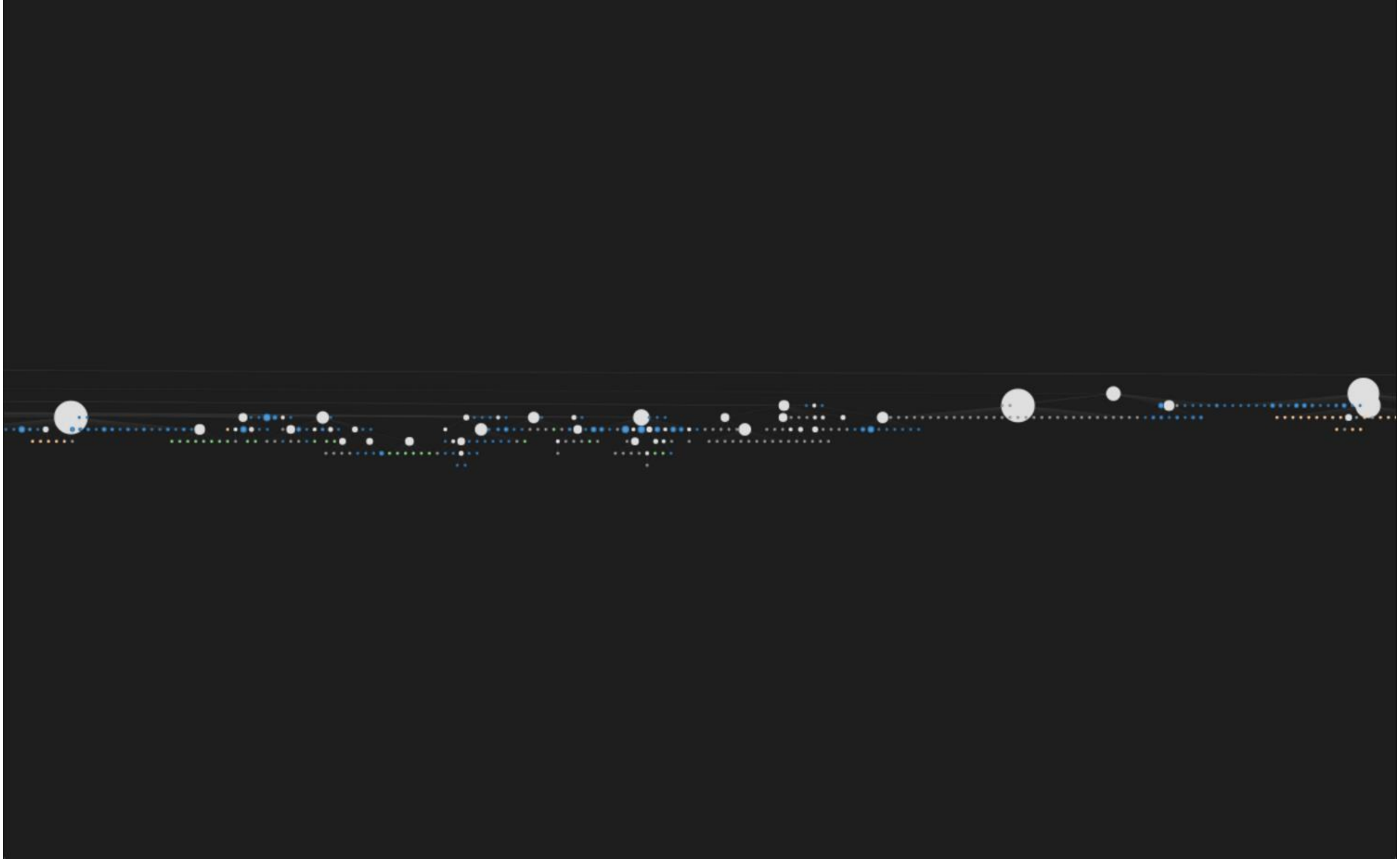


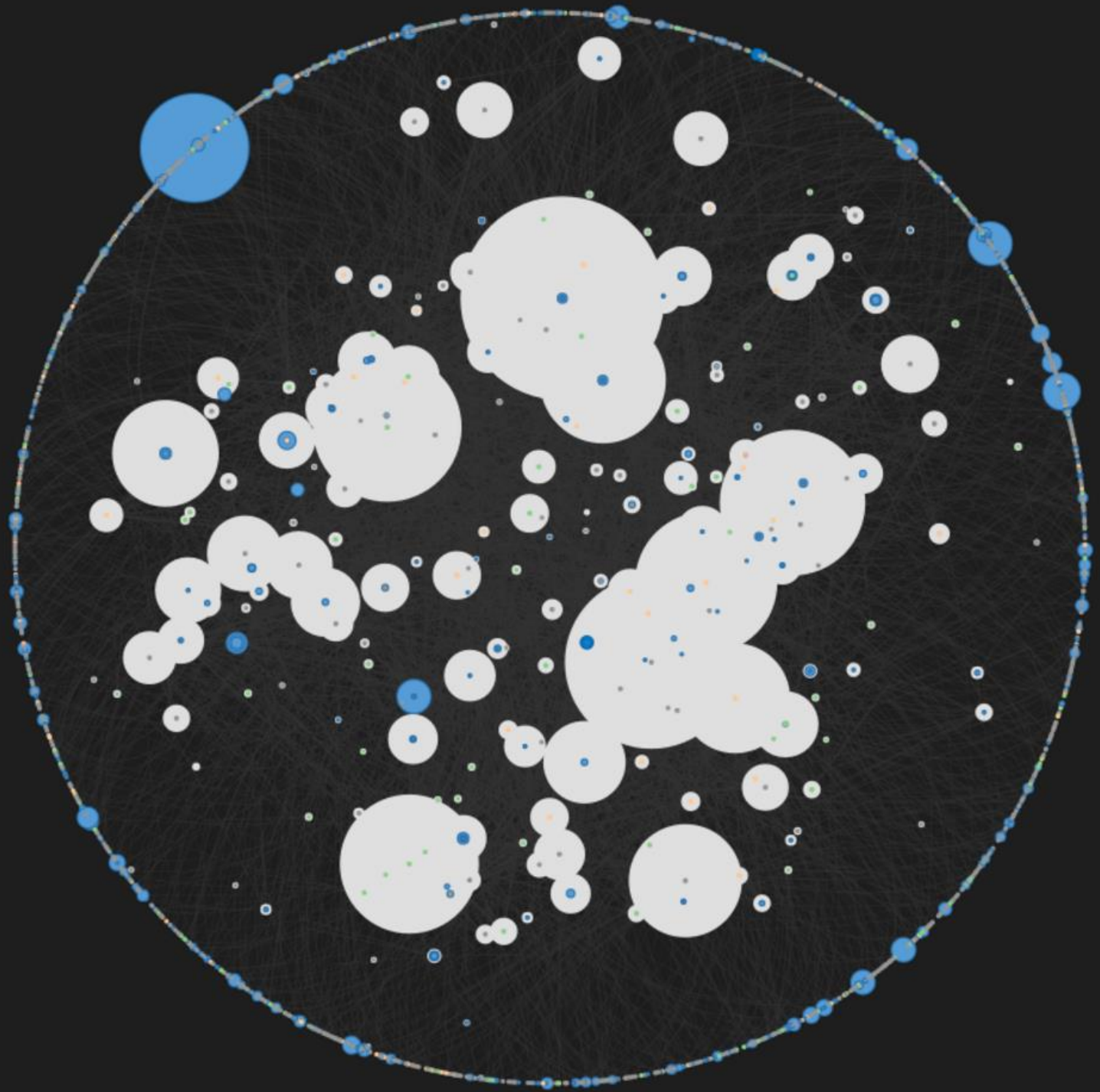

css
.gitignore
stylesheet.css
img
.gitignore
BackUpImages
bar.jpg
landinglogo.png
lights.png
bar (2).jpg
bar.jpg
landinglogo (2).png
landinglogo.png
lights (2).png
lights.png
js
.gitignore
map.js


editing.php
footer.php
header.php
humans.txt
index.php


editing.php
footer.php
header.php
humans.txt
index.php
body


.php
footer.php
header.php
humans.txt
index.php
body
div
div
div
site-menu
div





GUI Implementation missing features

- Single click links appearing
- Error list
  - Process bridge to communicated between the UI, Parser and VSCode

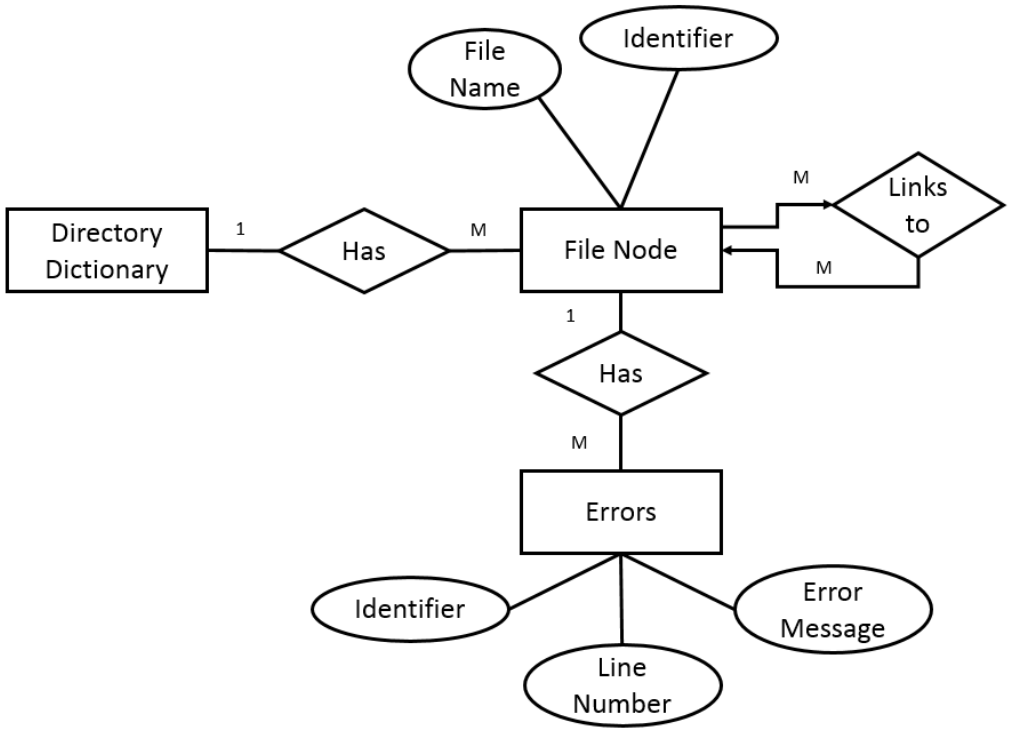
GUI Implementation Details

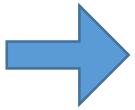
- File Map -> vis.js
- Error list -> Advanced News Ticker
- GUI window -> Electron application

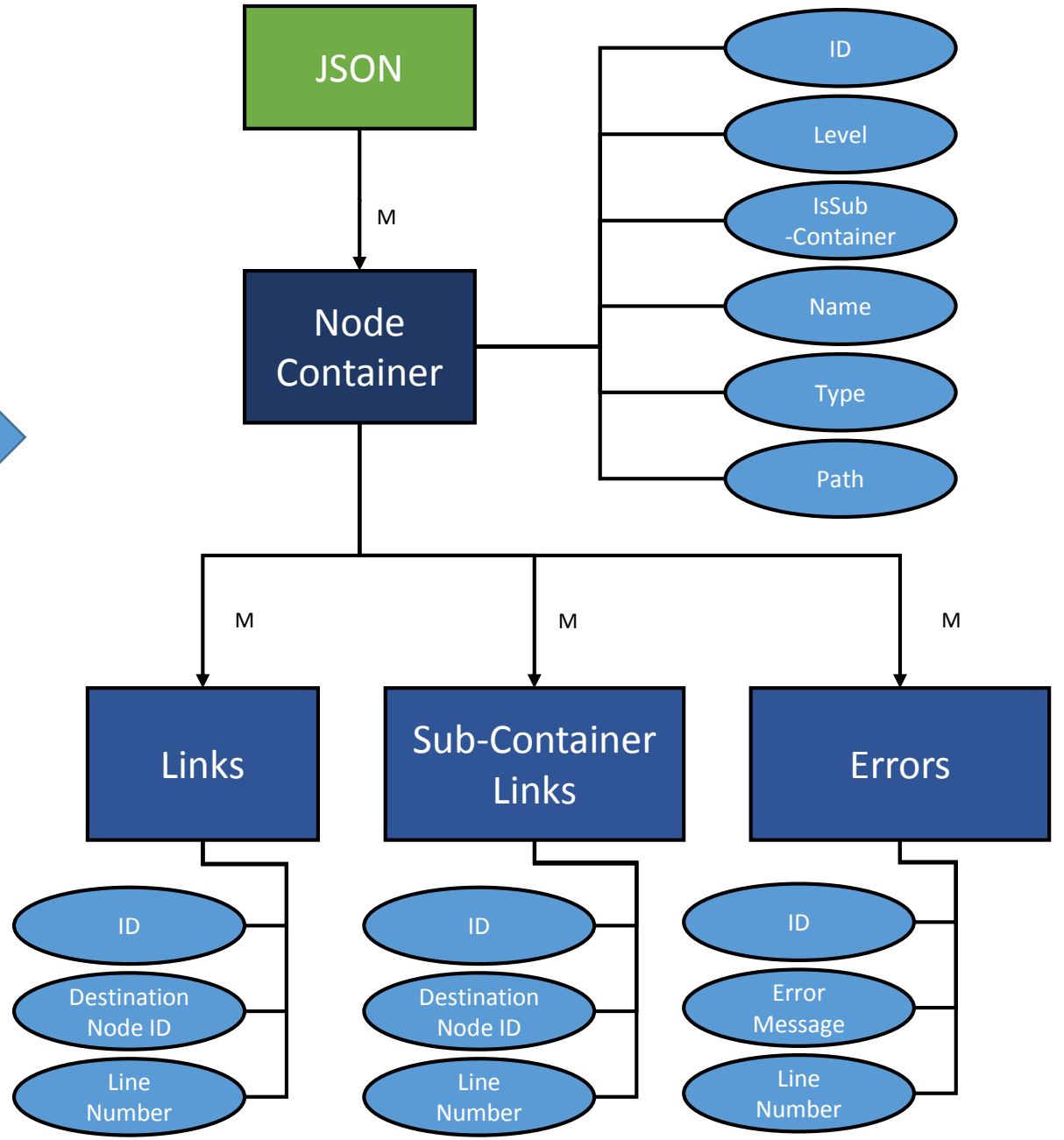
Data Layer

- Dictionary of file nodes and links collected by the parser
- Nodes in data layer contain info on all the files in the project, as well as error data detected by parser
- Node and error info stored in a file is used by the UI

Updated Schema





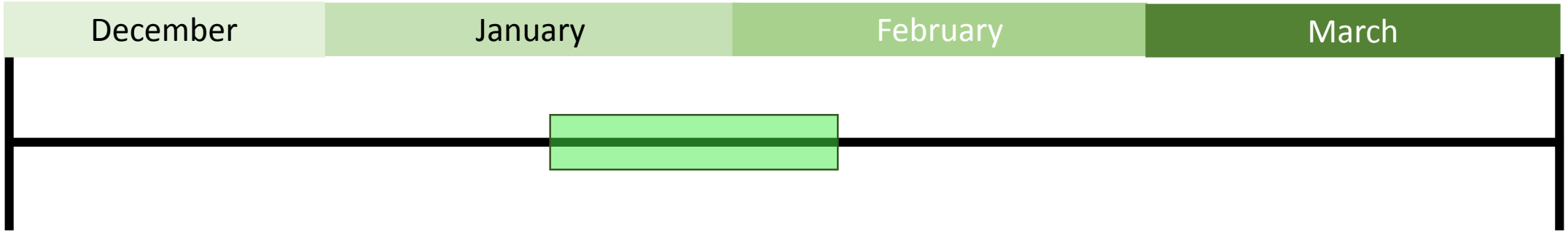


Term in Review

Timeline

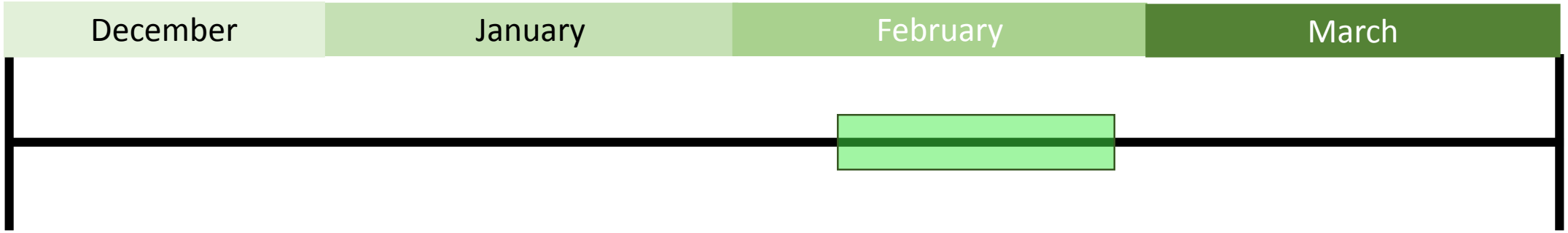
- Winter Break- Team got together to discuss distribution of work, got started on individual parts
- Week 1- Worked to develop very early prototype, client satisfied with progress, set end of February beta deadline

Timeline



- Week 2- Continued to tweak early prototype before moving on to major features
- Week 3- Worked on small UI changes
- Week 4- Began updating documentation, created OneNote “engineering log”, work continued on individual components

Timeline



- Week 5- Began combining project components, continued to update Technology Review and Design Document
- Week 6- Conducted major project rework, rewriting parser to read files line-by-line, switched to MVC design, updated data layer and UI to reflect new parser data

Team Performance: Positives

- Roles have been more clearly defined
  - Sam in charge of client communication
  - Zach in charge of documentation and submission
  - Cramer in charge of extension publishing
  - Eric in charge of overall extension design and tasks
- Client has been satisfied with out progress thus far
- Conducted code reviews to understand all parts of code
- Team has continued to work ahead of class schedule

Team Performance: What Needs Improvement

- Deadlines concerning research and user testing need to be clarified with our client
- Individual accountability for project tasks needs to be improved
- We need to reference and update our documentation more frequently

Changes To Be Made

- We will meet with our client to solidify user testing details
- We will communicate amongst each other better on what it expected and when it needs to be done
- Biweekly documentation checks/meetings will be conducted to evaluate if anything needs to be changed

Future Plans

Future Plans

- Bug Fixes
- Implement a Process Bridge
  - Send information between the UI and the extension logic
- C-Like Function support for the parser
  - Change parser behavior
  - Build in default grammar support
- Errors
  - Change parser behavior
  - Build in default grammar support
- User Testing