App Architecture

Faster, Better... Simpler

Kirill Bubochkin

What is architecture?

What is not architecture?

- BLoC, Riverpod, MVVM
- Based on SOLID
- Clean Architecture*

Problems with "Clean Architecture"

- It doesn't tell much about your app
- It's often misinterpreted
- Doesn't work for mobile apps: https://bit.ly/3XBOpeU

What is architecture?

The goal of software architecture is to minimize the human resources required to build and maintain the required system.

Robert C. Martin "Clean Architecture"

The goal of software architecture is to minimize the human resources required to build and maintain the required system.

Robert C. Martin "Clean Architecture"

What is architecture?

Software architecture is the set of structures needed to reason about a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.

https://en.wikipedia.org/wiki/Software_architecture

Components

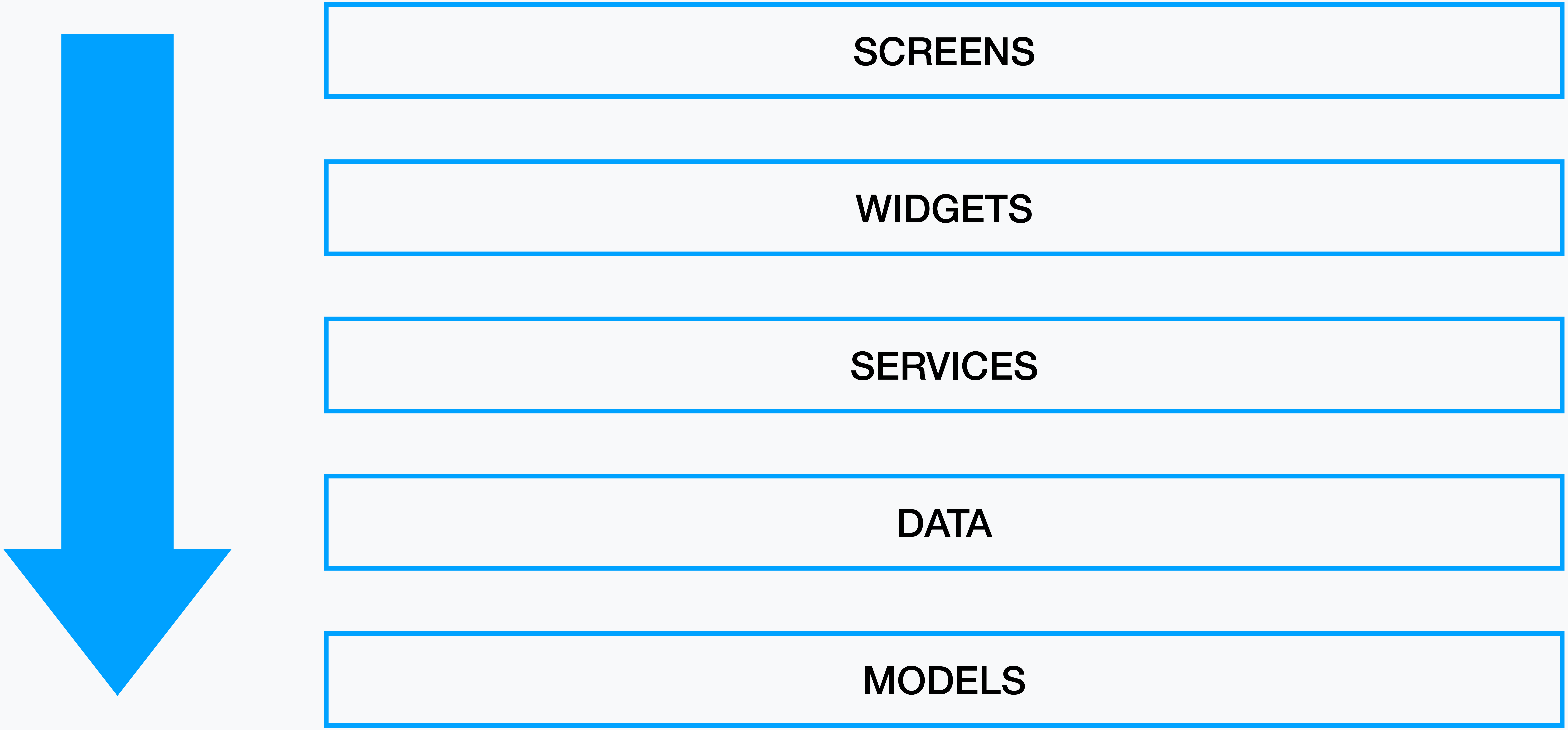
A grouping of related functionality behind a nice clean interface, which resides inside an execution environment like an application.

Simon Brown

Components

- Not screens
- Not (necessarily) user flows
- Not (necessarily) "global" responsibilities
- Loosely coupled, highly cohesive
- Encapsulated

Inside of the component



SCREENS

WIDGETS

SERVICES

DATA

MODELS

Inside of the component

- Widgets are not "dumb" views: https://bit.ly/45CuZbZ
- Not all layers are required
- Layer can communicate with any lower level
- No dependency inversion 😱
- Is BLoC a service?

Inside of the component

- Widgets are not "dumb" views: https://bit.ly/45CuZbZ
- Not all layers are required
- Layer can communicate with any lower level
- No dependency inversion 😱
- Is BLoC a service? It depends...

SOLID

- Single Responsibility Principle: useful when not misinterpreted
- Open/Closed Principle: brings overhead, not needed everywhere
- Liskov Substitution Principle: avoid inheritance
- Interface Segregation Principle: useful, but brings overhead
- Dependency Inversion Principle: useful in some scenarios

DRY

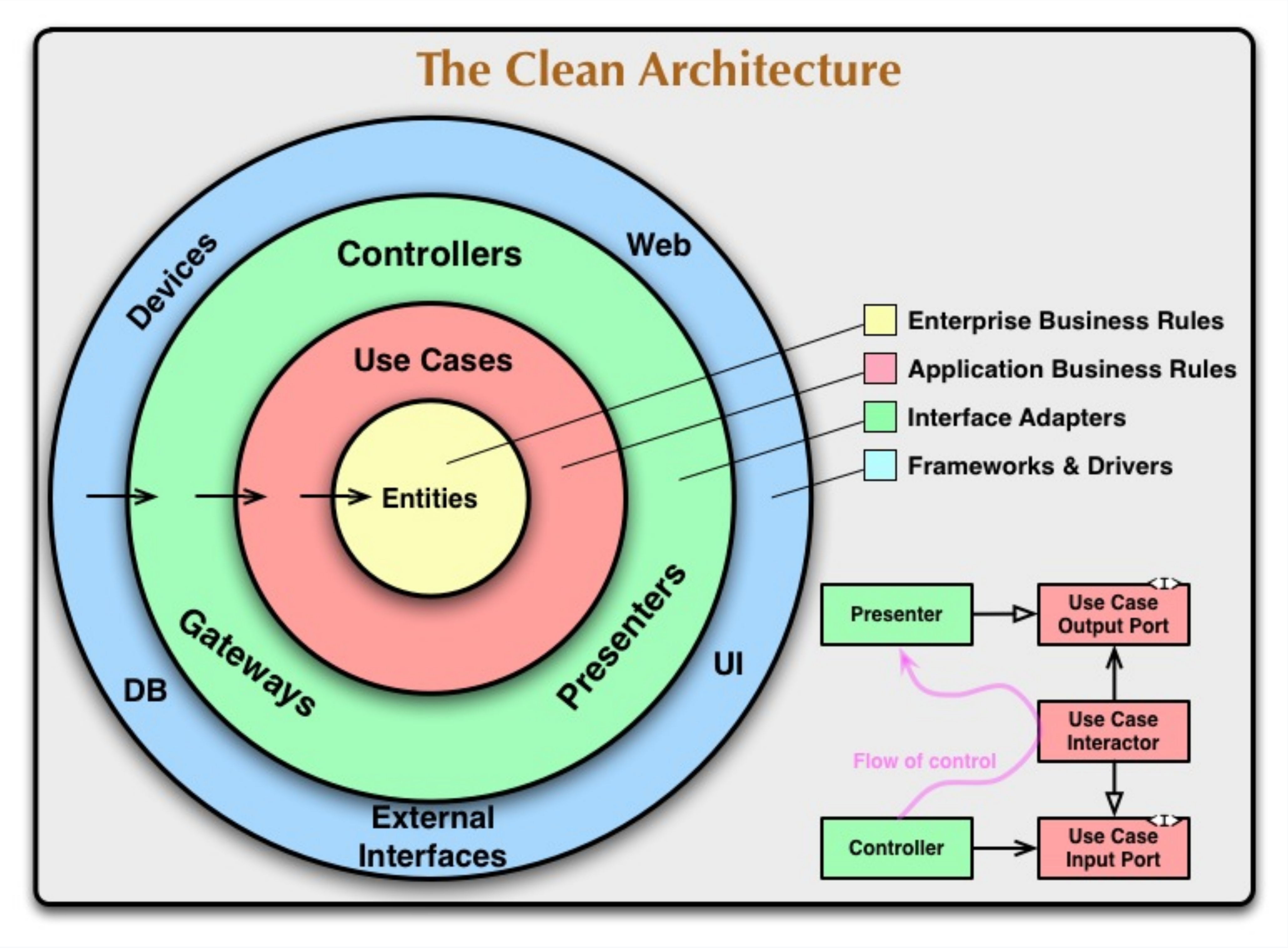
- Keep a single source of truth for your logic.
- Reduce repetition of information which is likely to change.
- Replace it with abstractions that are less likely to change.
- Logically related elements change predictably and uniformly.

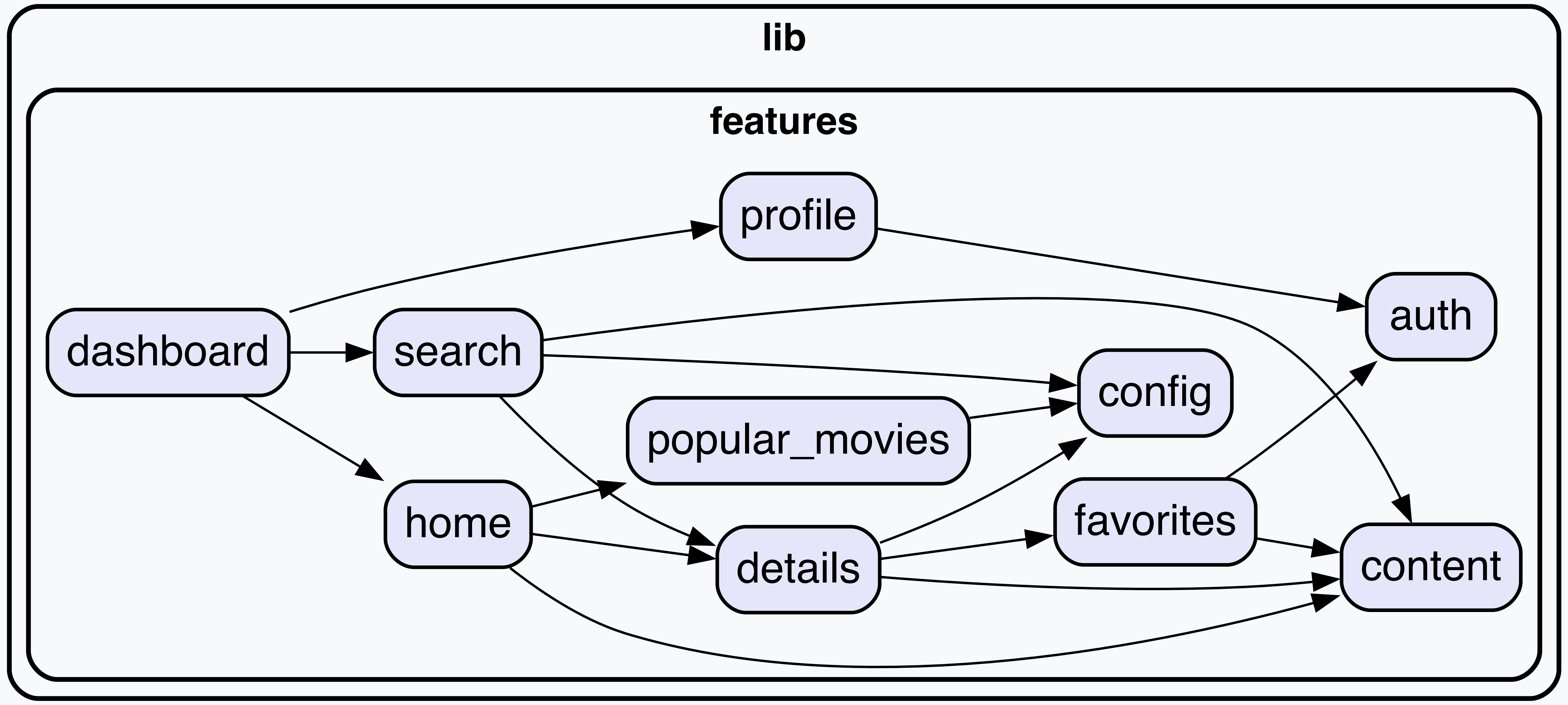
! DRY ≠ WET

Relations

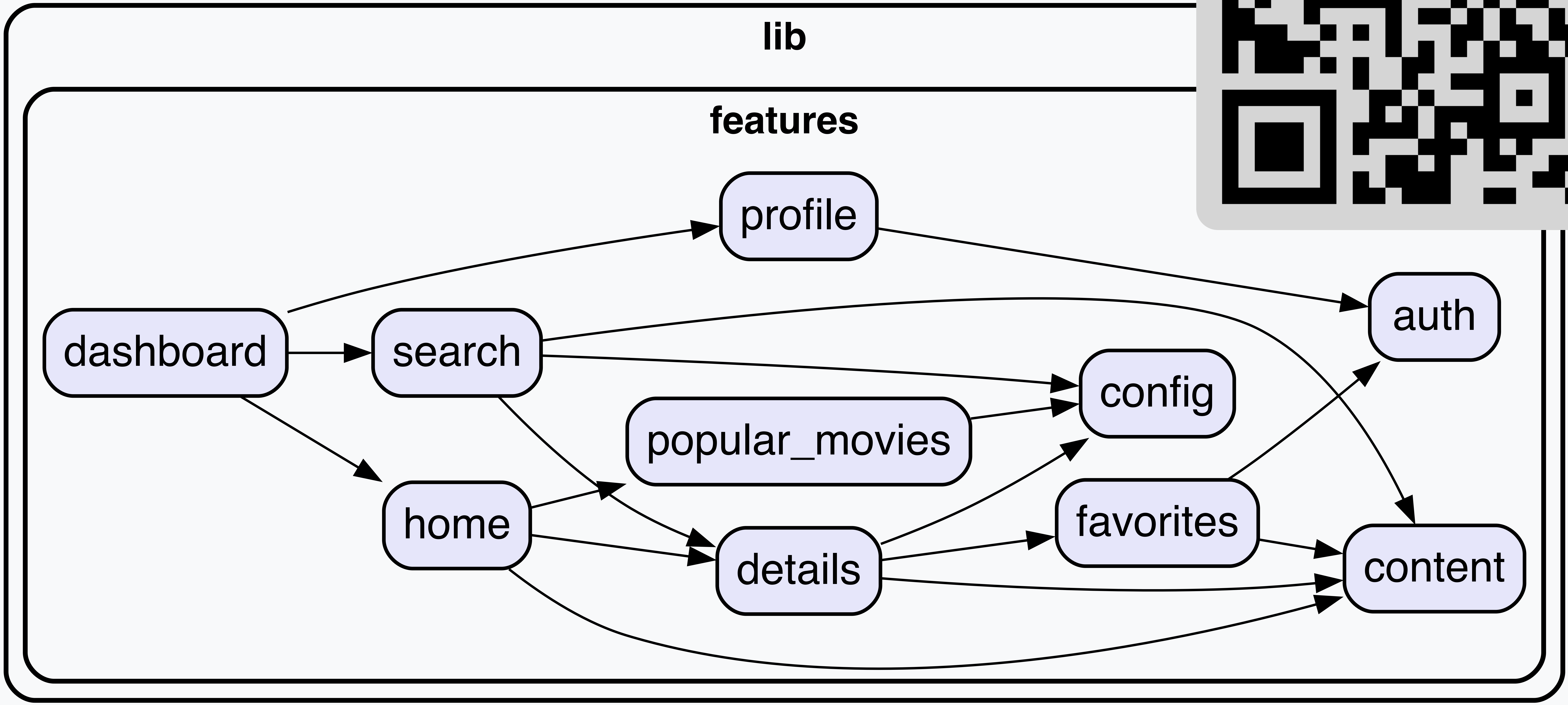
App Architecture

Your App Architecture

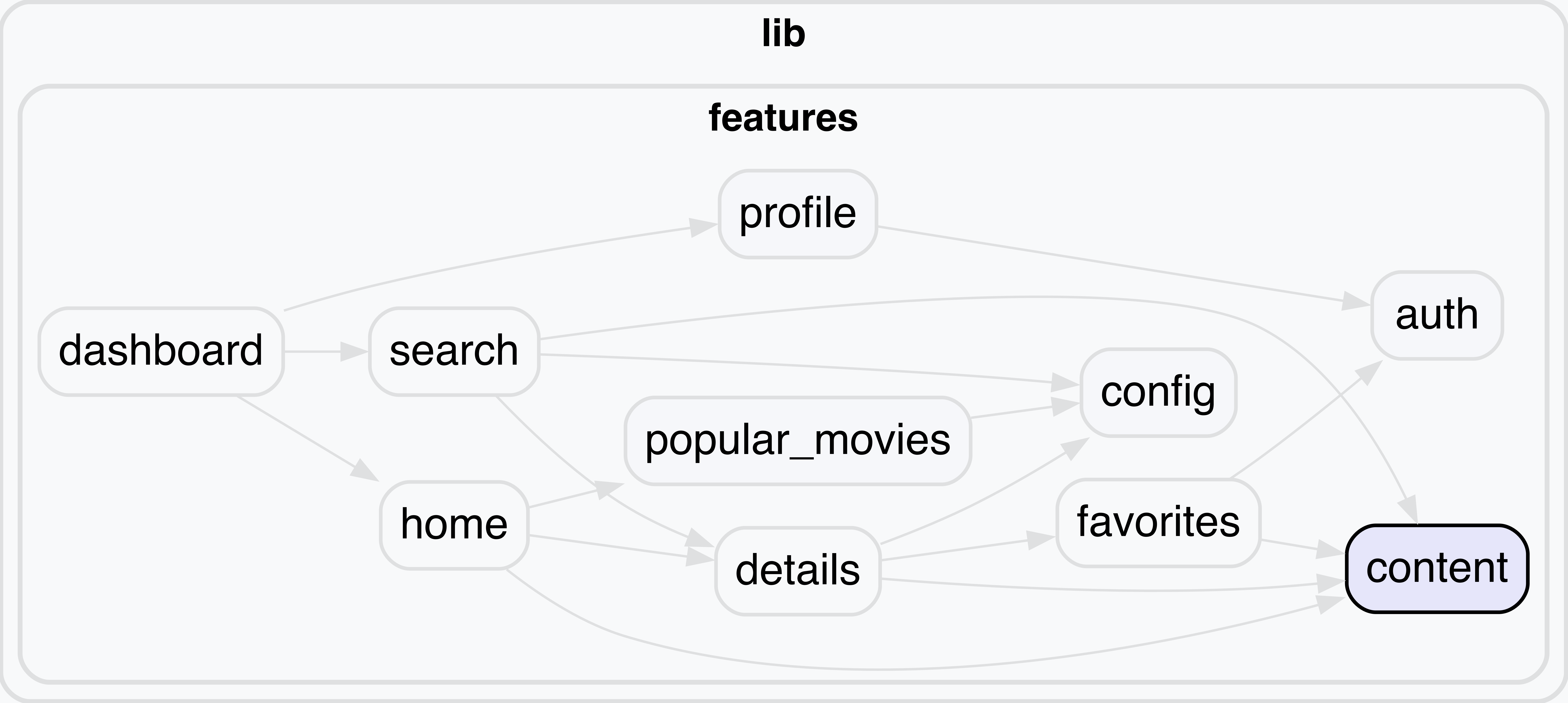




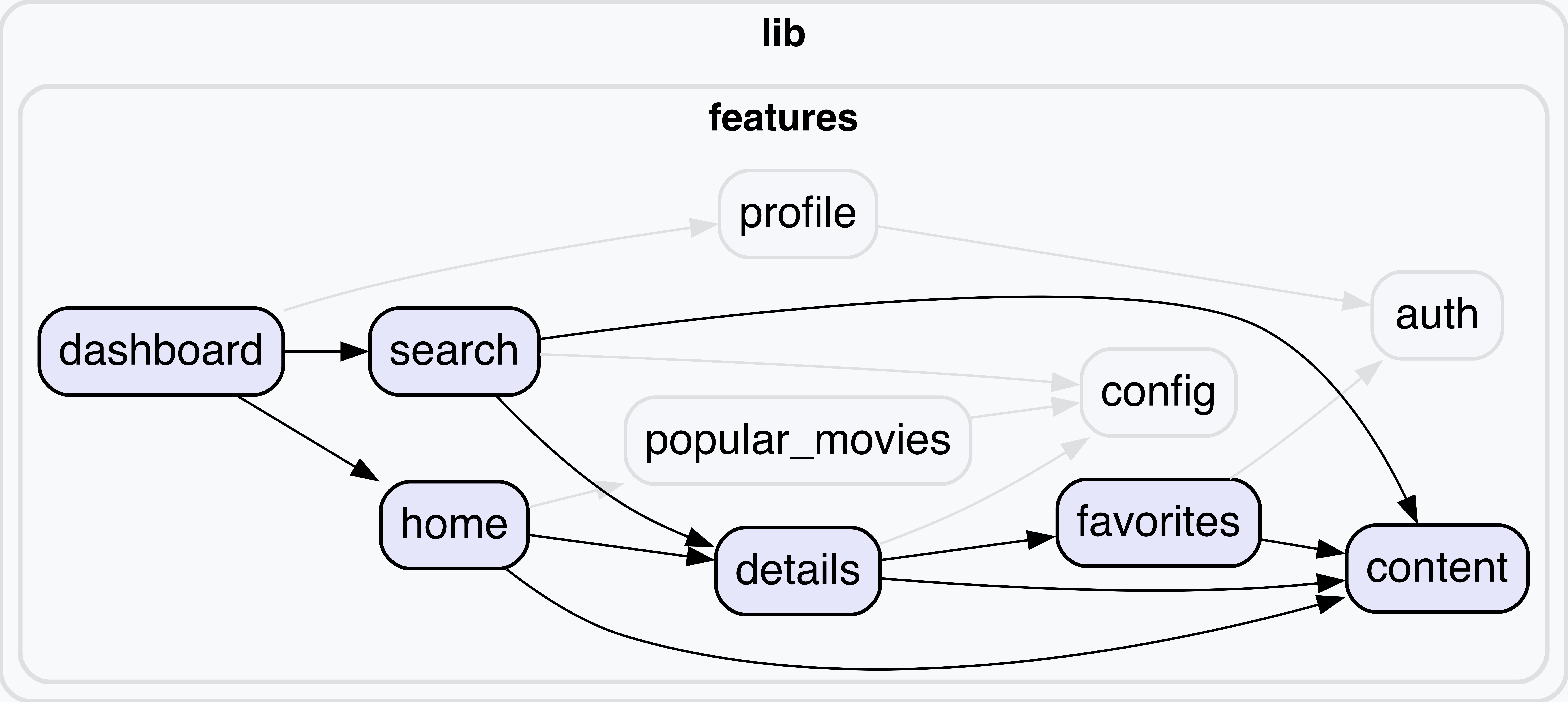
Your App Architecture



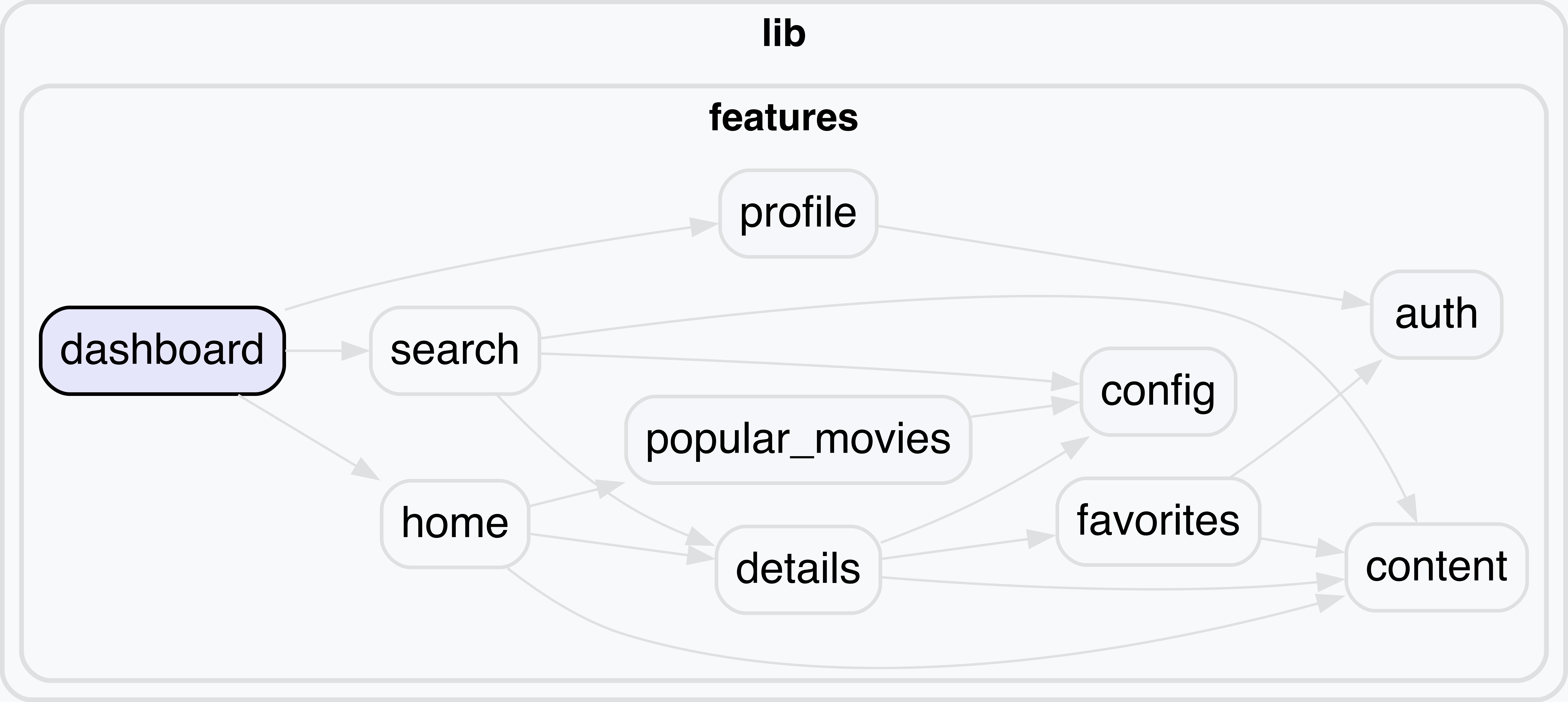
Stable components



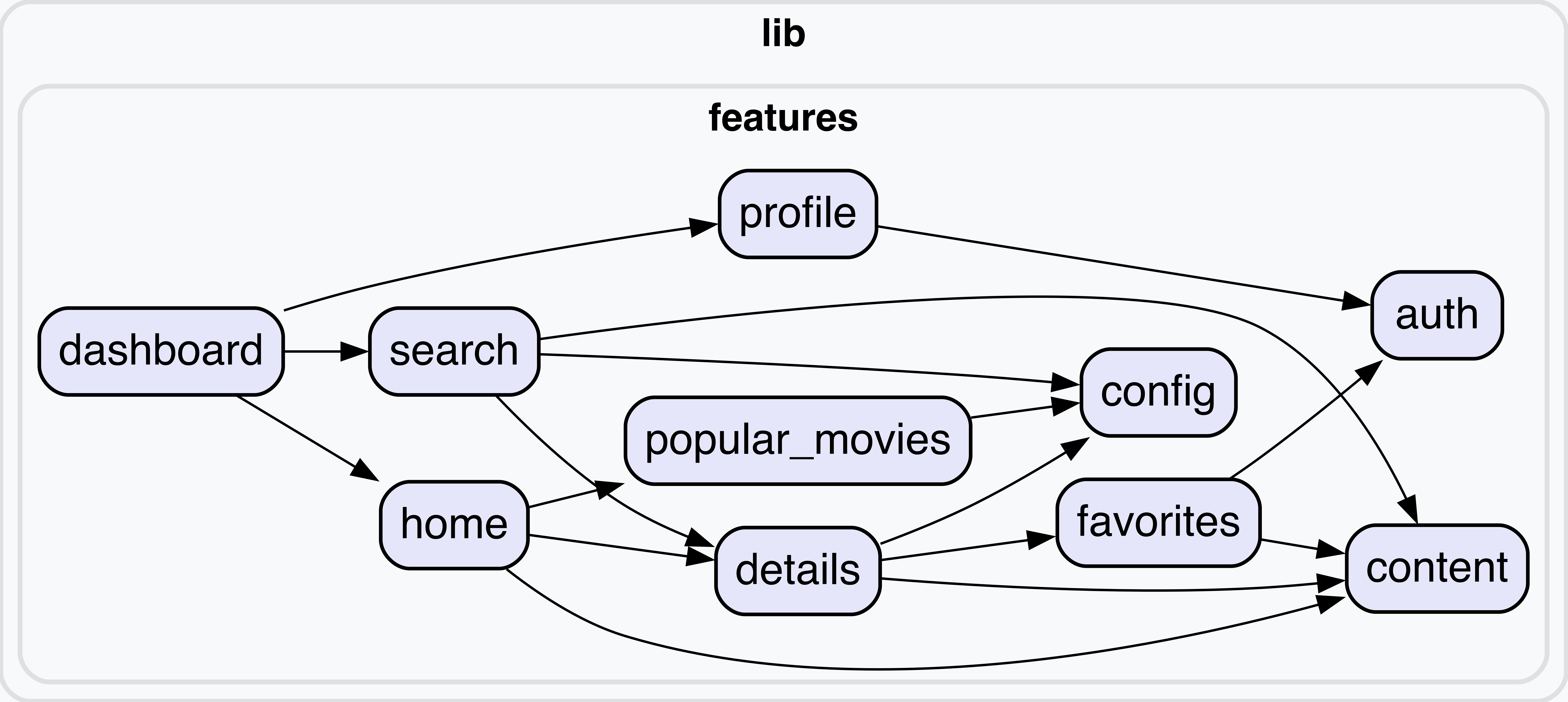
Stable components



Unstable components

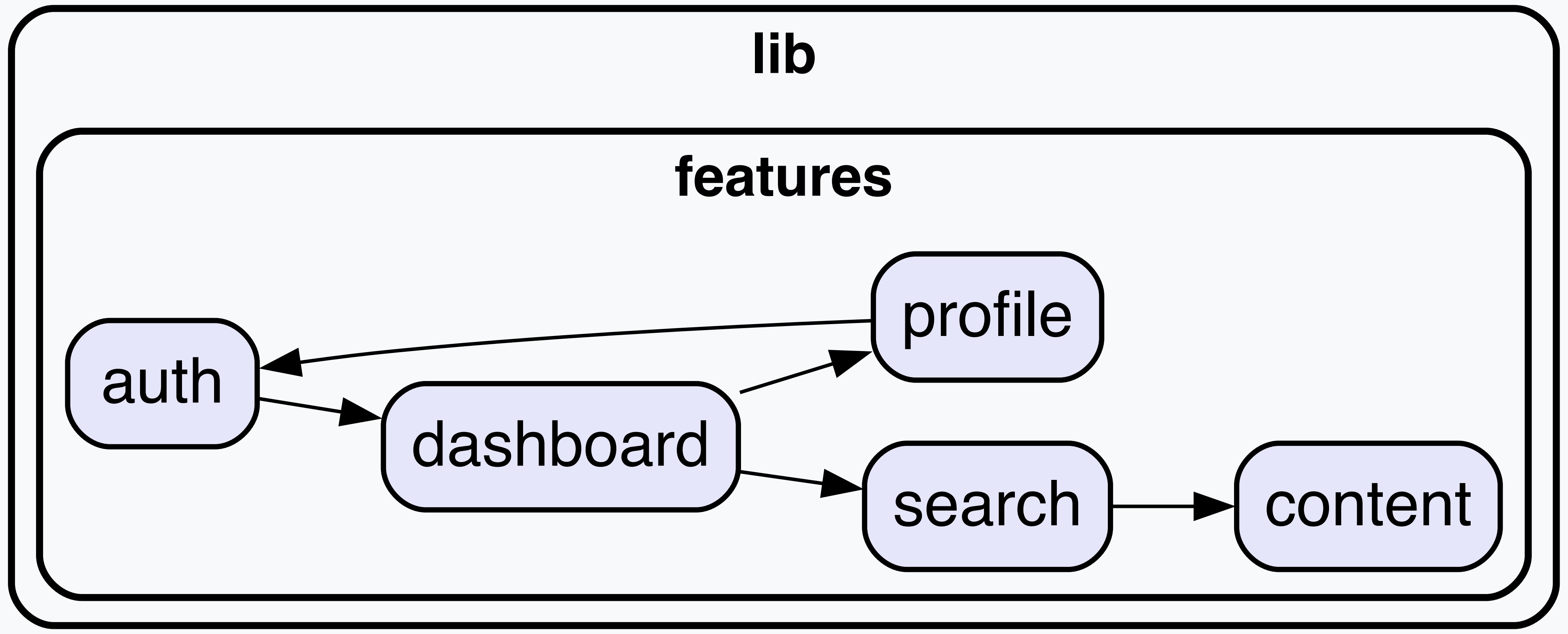


Unstable components

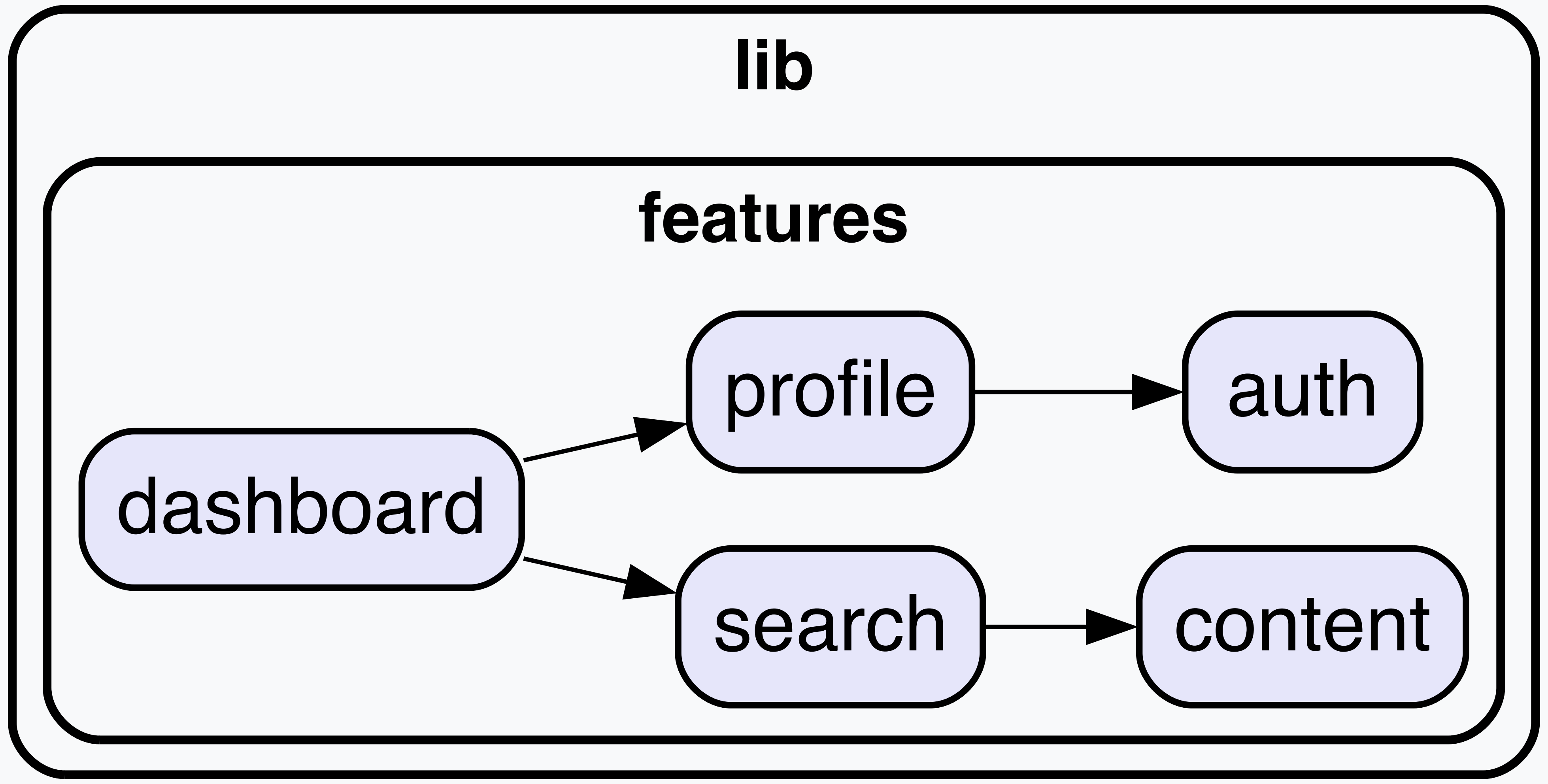


Directed Acyclic Graph

❌ Cycles

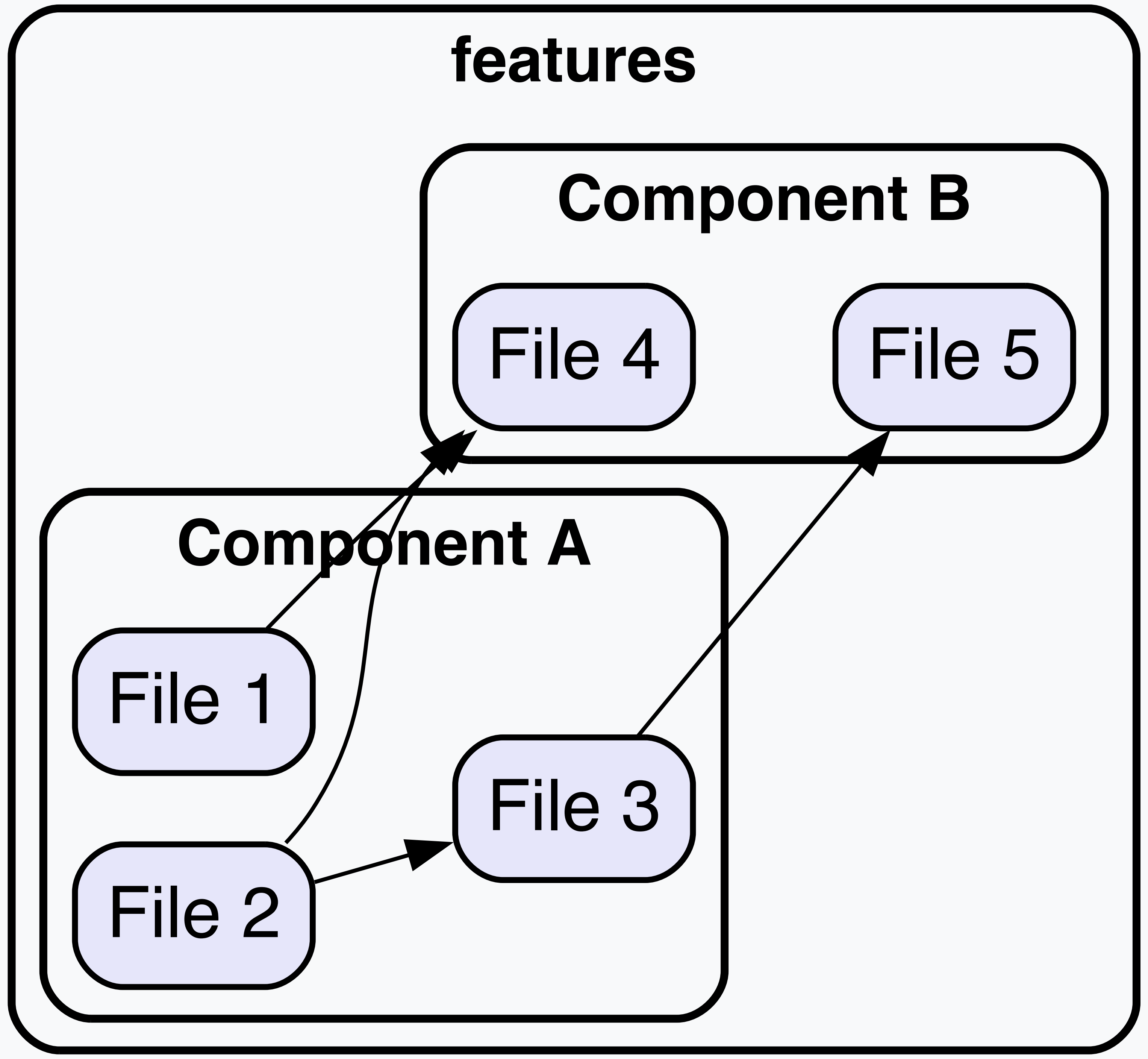


✅ No cycles

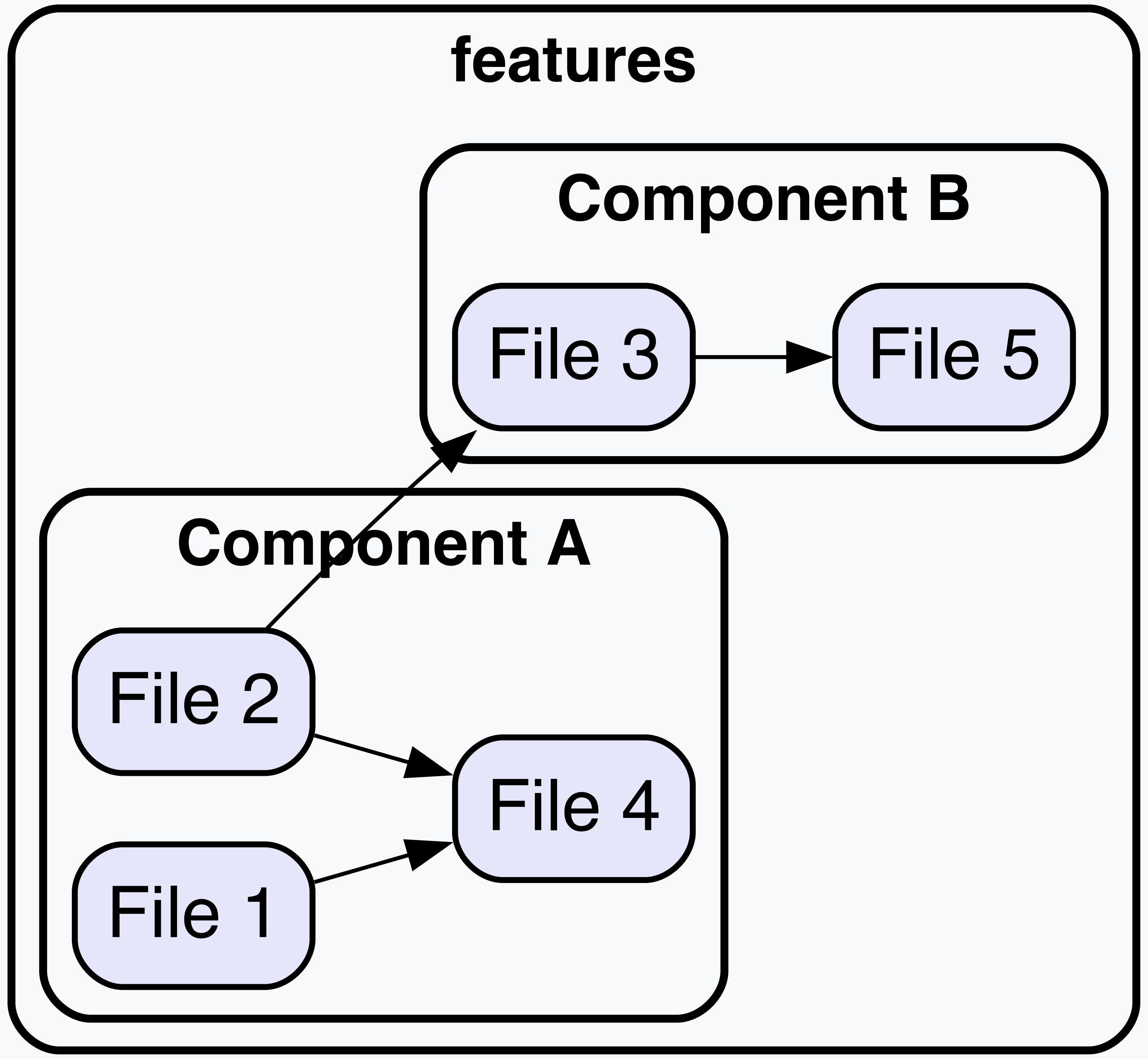


Loose coupling, high cohesion

❌ Coupling ↑ cohesion ↓



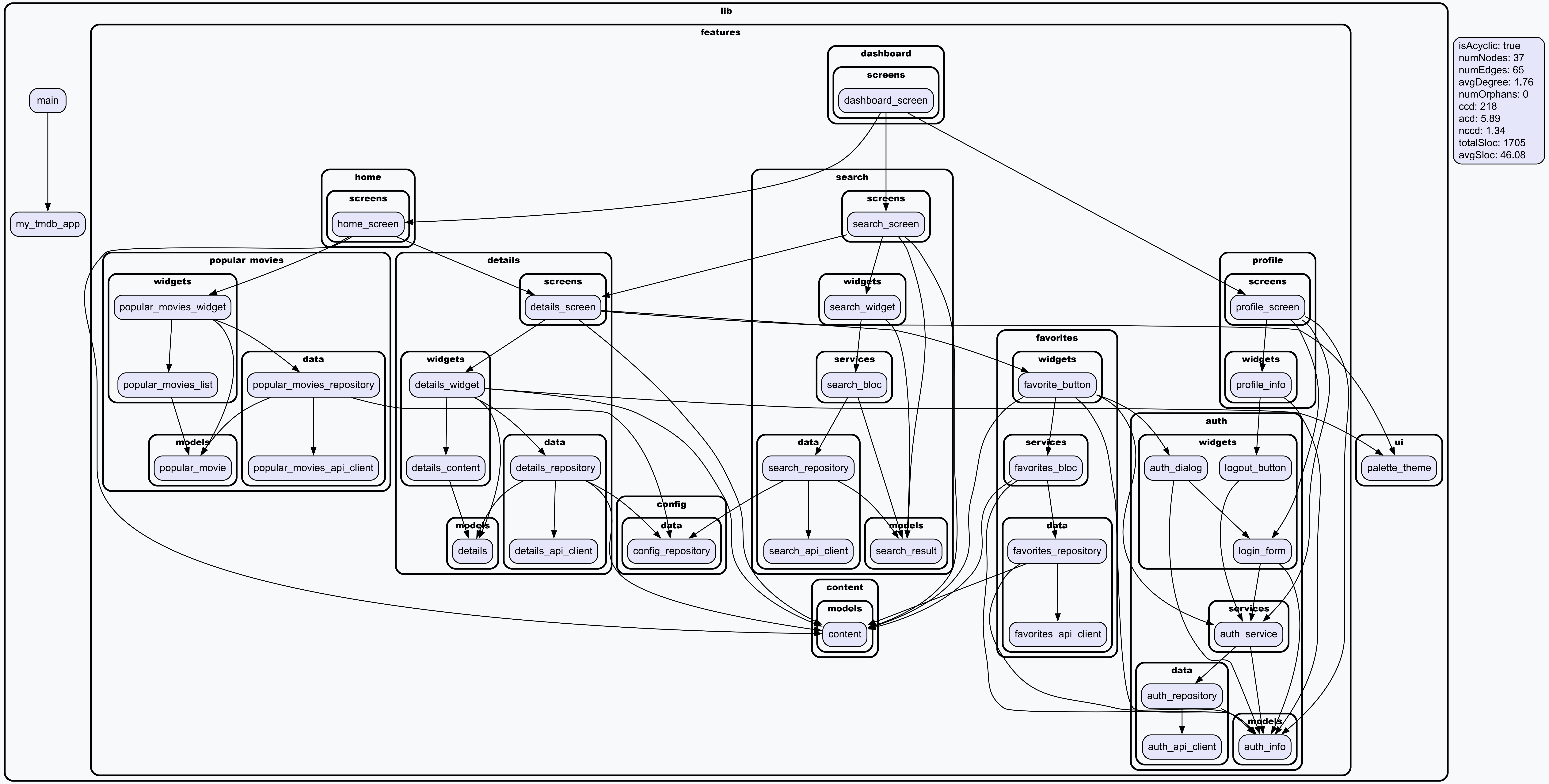
✅ Coupling ↓ cohesion ↑



Tools

Tools

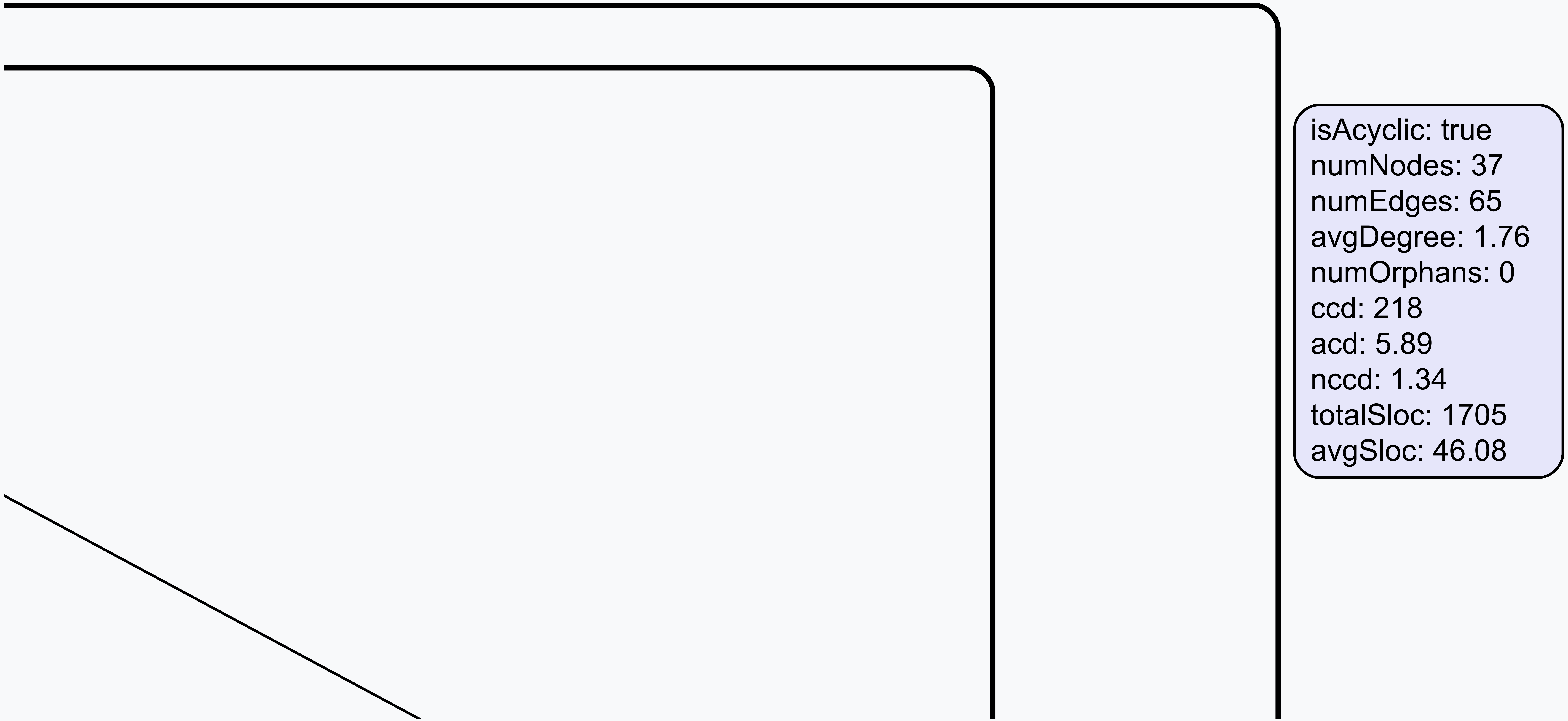
lakos



```
isAcyclic: true
numNodes: 37
numEdges: 65
avgDegree: 1.76
numOrphans: 0
ccd: 218
acd: 5.89
nccd: 1.34
totalSloc: 1705
avgSloc: 46.08
```

Tools

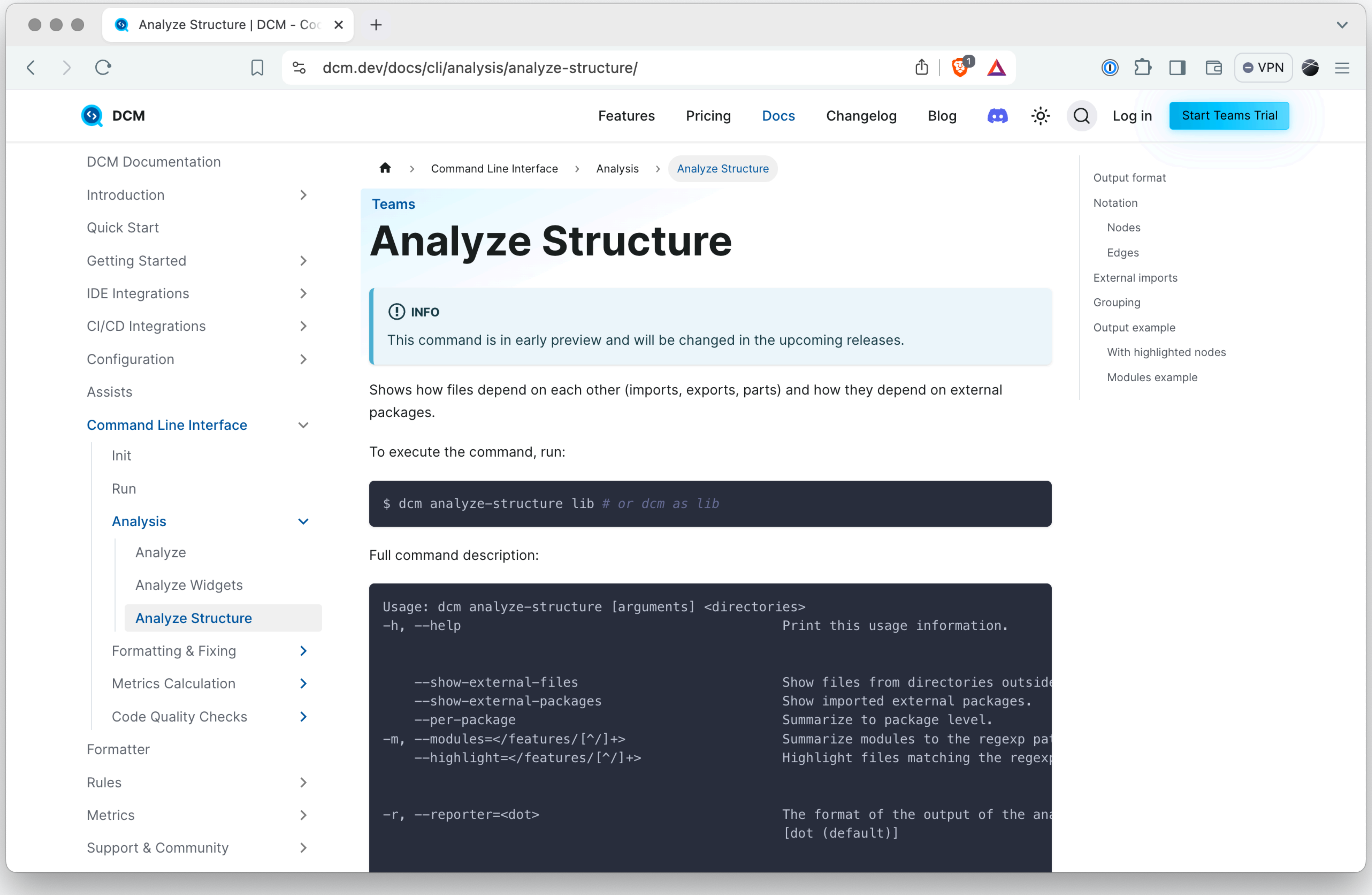
lakos



isAcyclic: true
numNodes: 37
numEdges: 65
avgDegree: 1.76
numOrphans: 0
ccd: 218
acd: 5.89
nccd: 1.34
totalSloc: 1705
avgSloc: 46.08

Tools

DCM



Teams

Analyze Structure

INFO

This command is in early preview and will be changed in the upcoming releases.

Shows how files depend on each other (imports, exports, parts) and how they depend on external packages.

To execute the command, run:

```
$ dcm analyze-structure lib # or dcm as lib
```

Full command description:

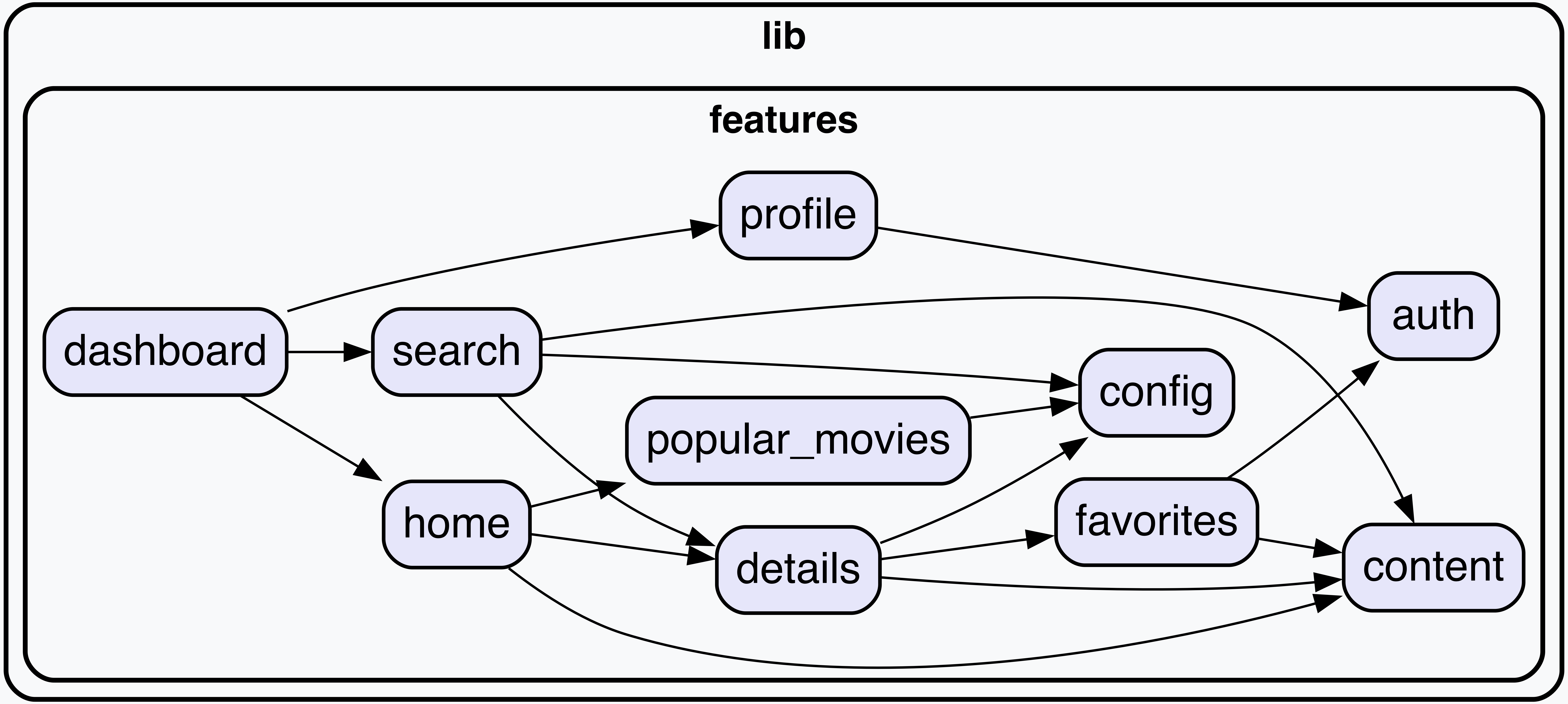
```
Usage: dcm analyze-structure [arguments] <directories>
-h, --help                                         Print this usage information.


    --show-external-files                          Show files from directories outsid
    --show-external-packages                       Show imported external packages.
    --per-package                                  Summarize to package level.
-m, --modules=</features/[^/]+>                    Summarize modules to the regexp pa
    --highlight=</features/[^/]+>                  Highlight files matching the regex


-r, --reporter=<dot>                               The format of the output of the ana
                                                   [dot (default)]
```

Tools

DCM


lib
features
profile
auth
dashboard
search
config
popular_movies
home
favorites
details
content

Benefits

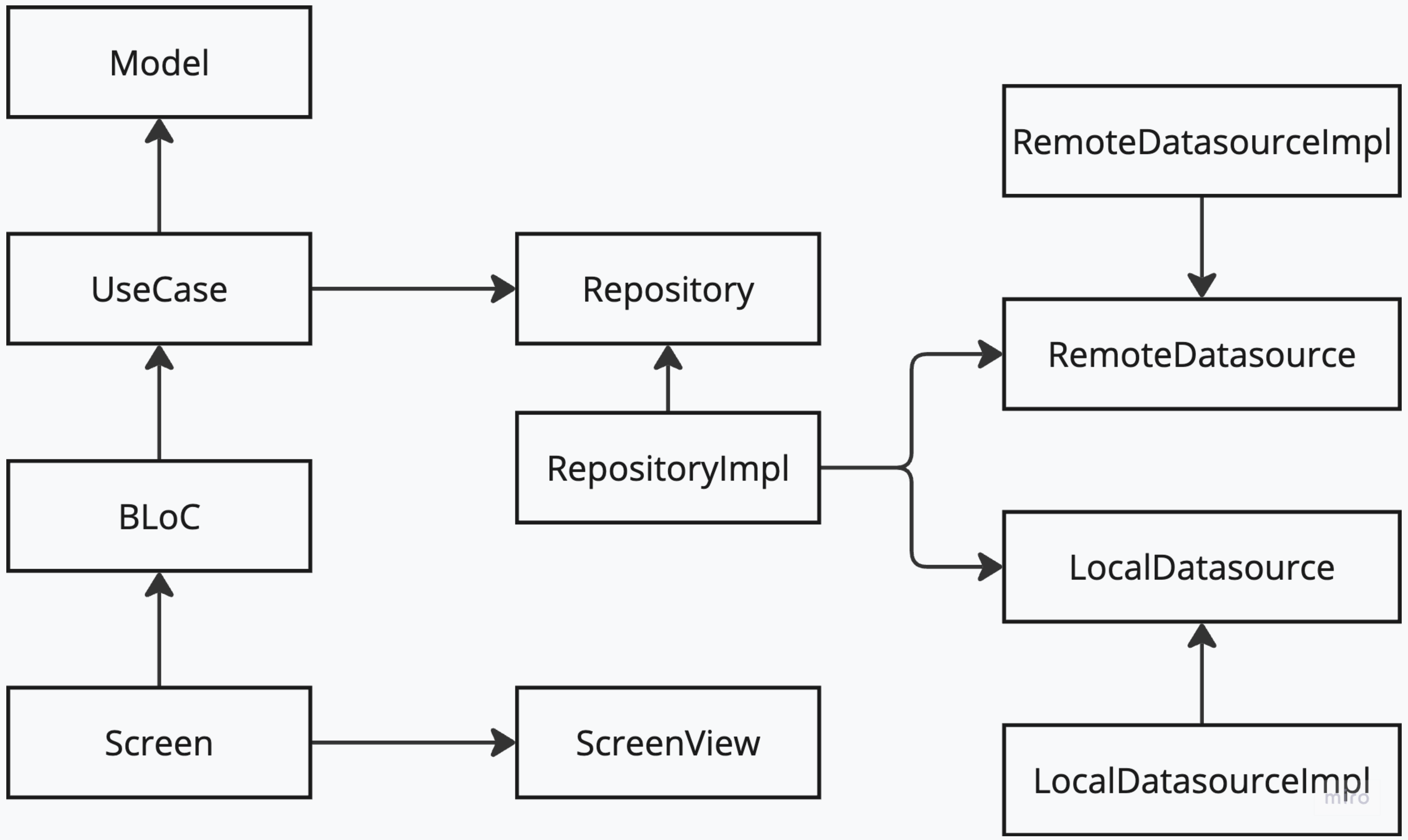
Benefits

- Flexibility: independent modules
- Transparency: overview of "stable" and "unstable" components
- Maintainability: chaos is local
- Simplicity: no unnecessary abstractions
- Scalability: decomposition

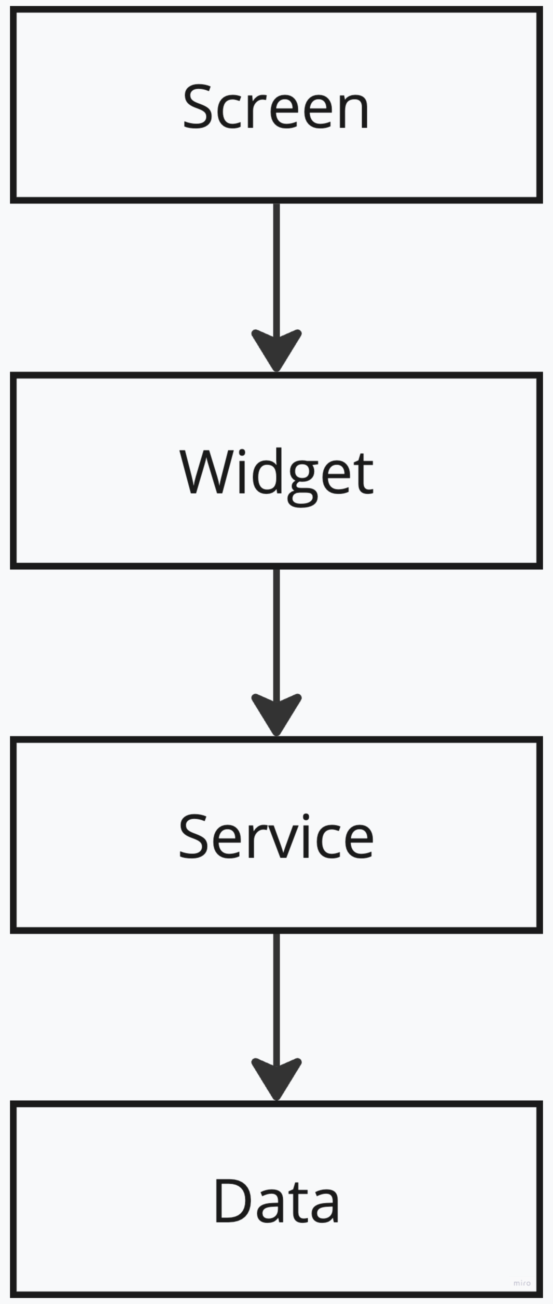
P.S. Is it really different from Clean Architecture?

Simplification

❌ "Traditional"

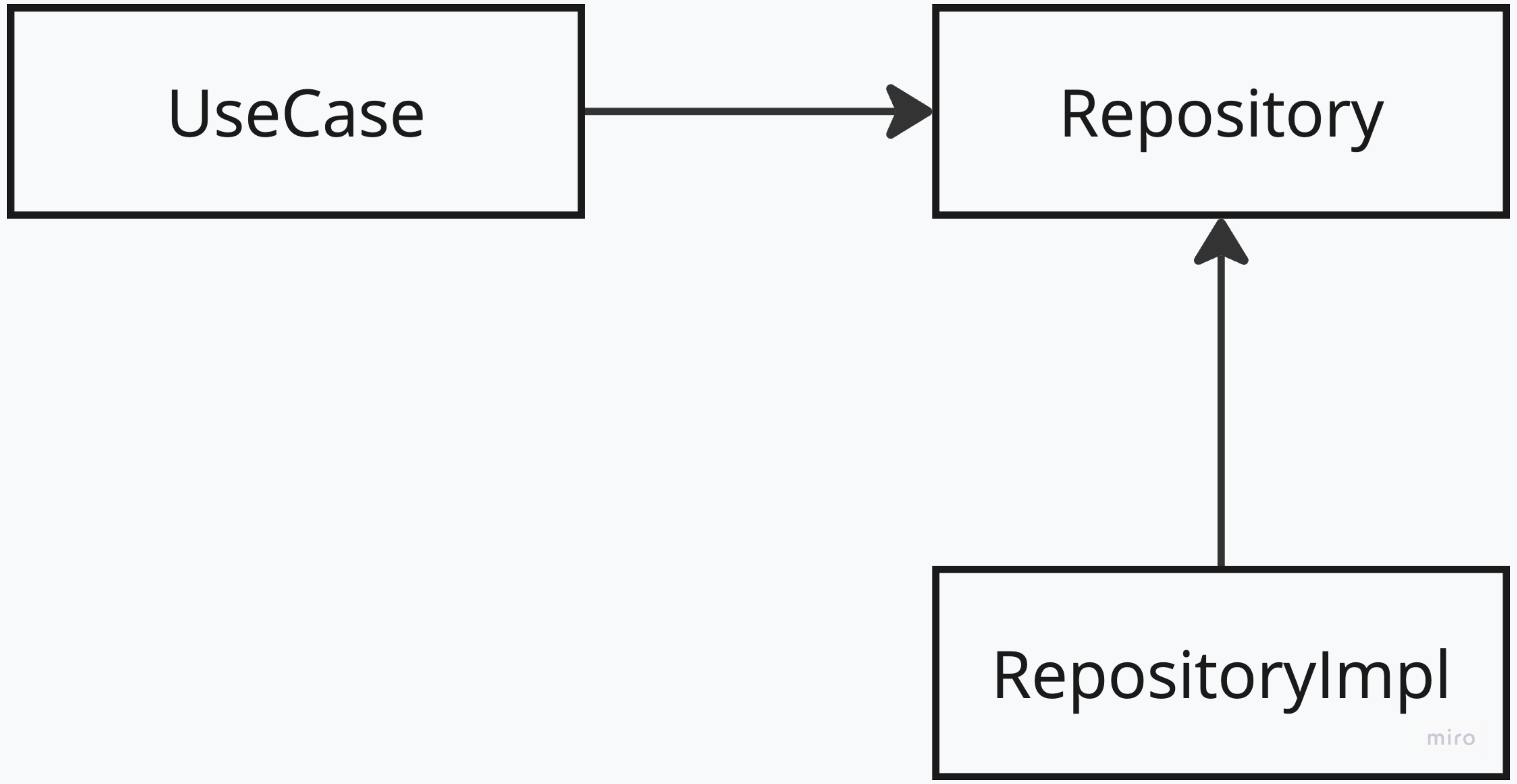


✅ Pragmatic



Inversion of Dependency Inversion

❌ "Traditional"

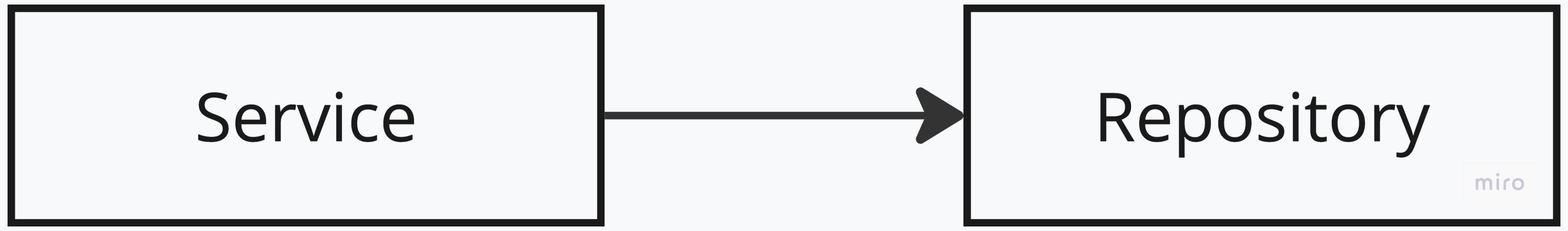


UseCase

Repository

RepositoryImpl

✅ Pragmatic



Service

Repository

Also

- Structure by components
- Keep related functionality close
- Keep components small
- Focus on components relations

Thank you!

linkedin.com/in/ookamikb/