App Architecture

Faster, Better... Simpler

Kirill Bubochkin

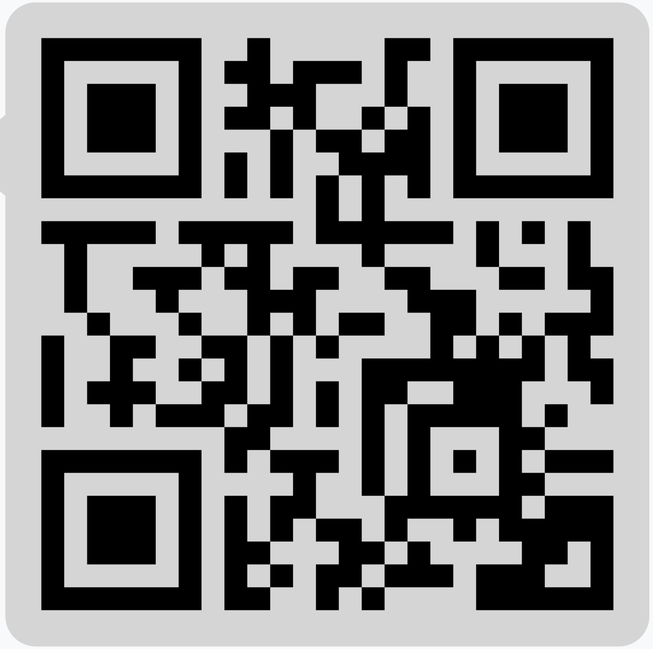
What is architecture?

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

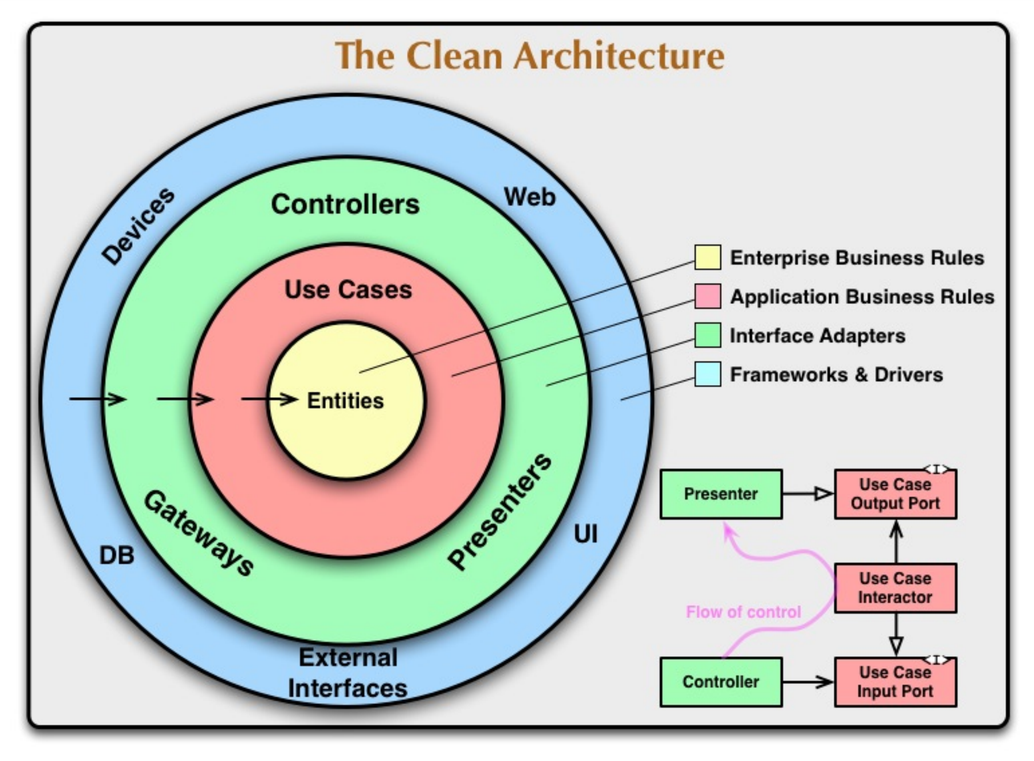
DRY

- 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.

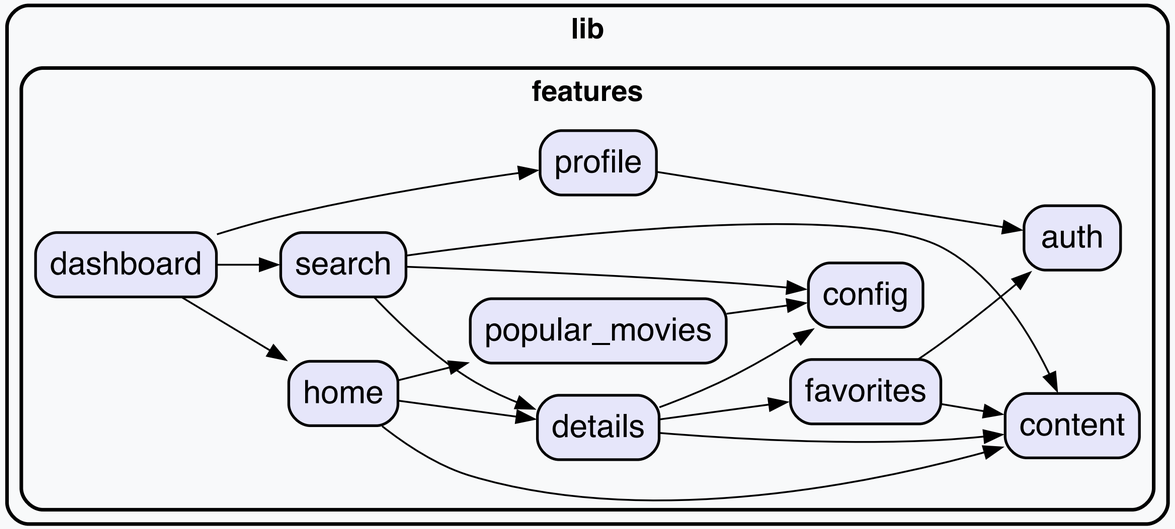
! DRY ≠ WET

Relations

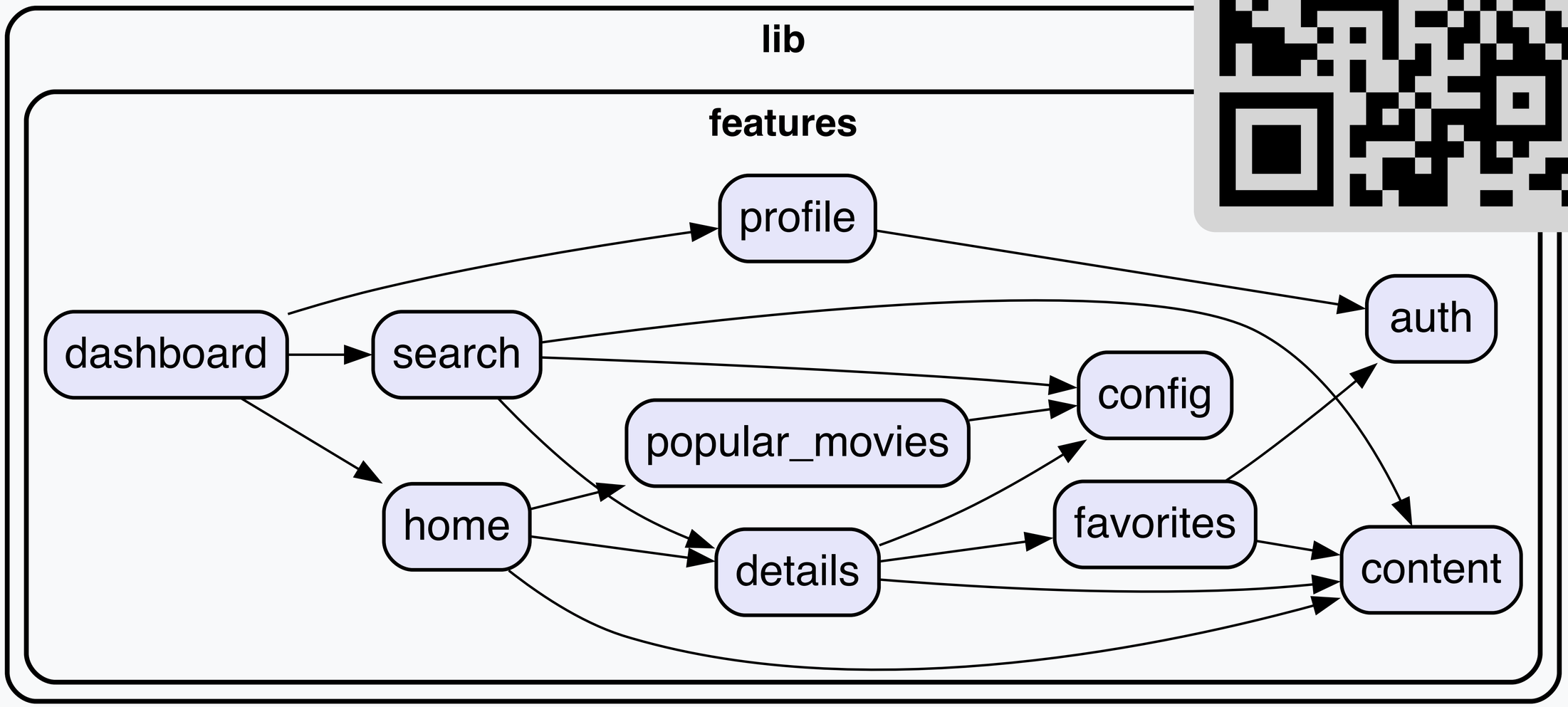
App Architecture



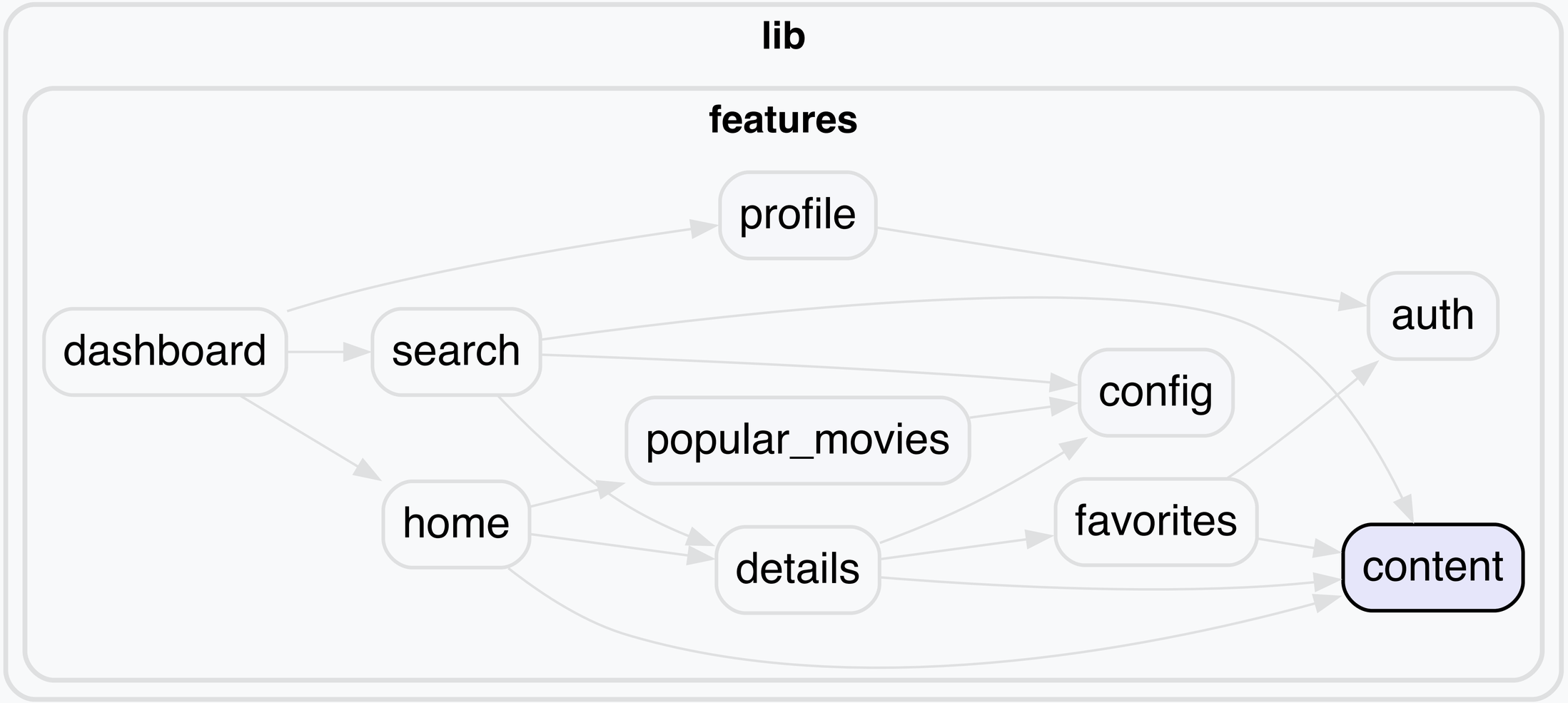
Your App Architecture



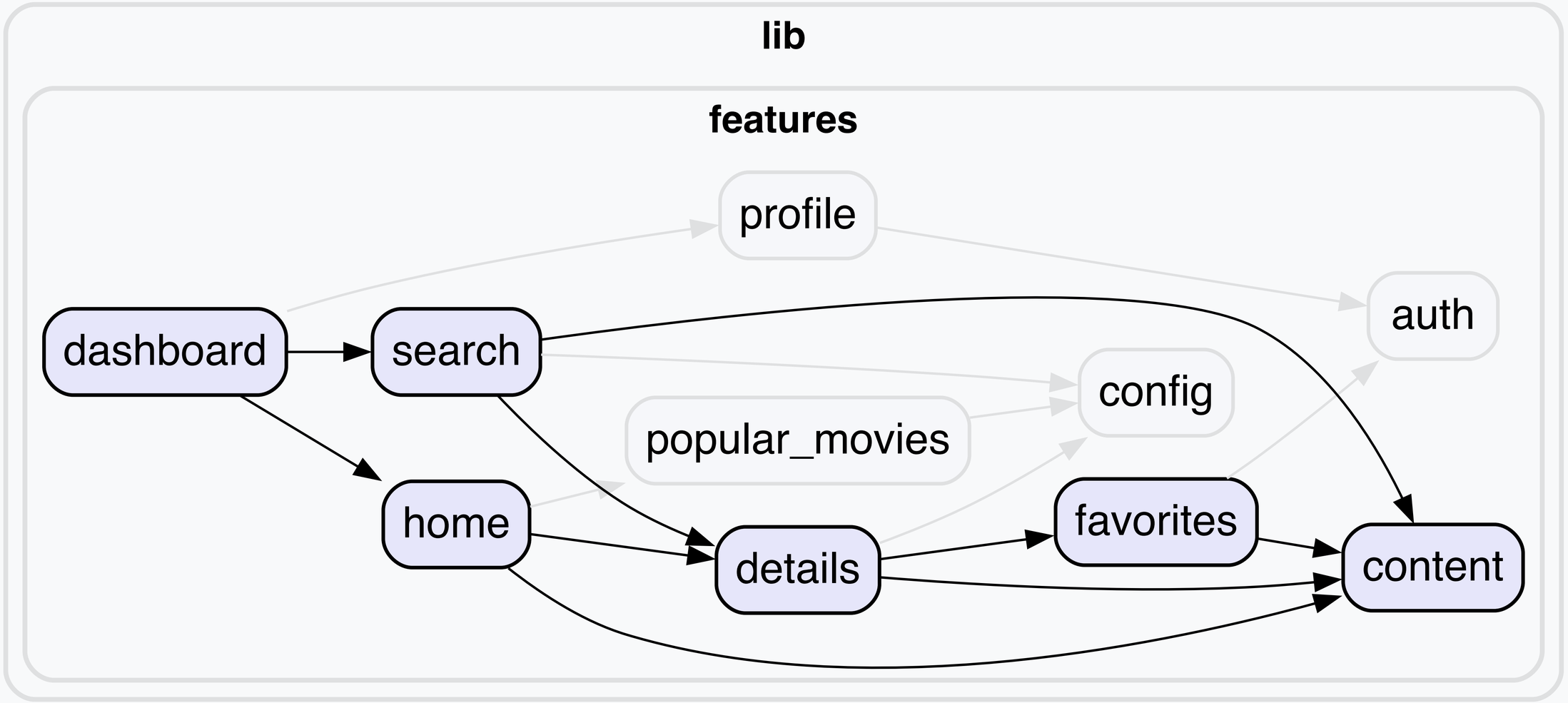
Your App Architecture



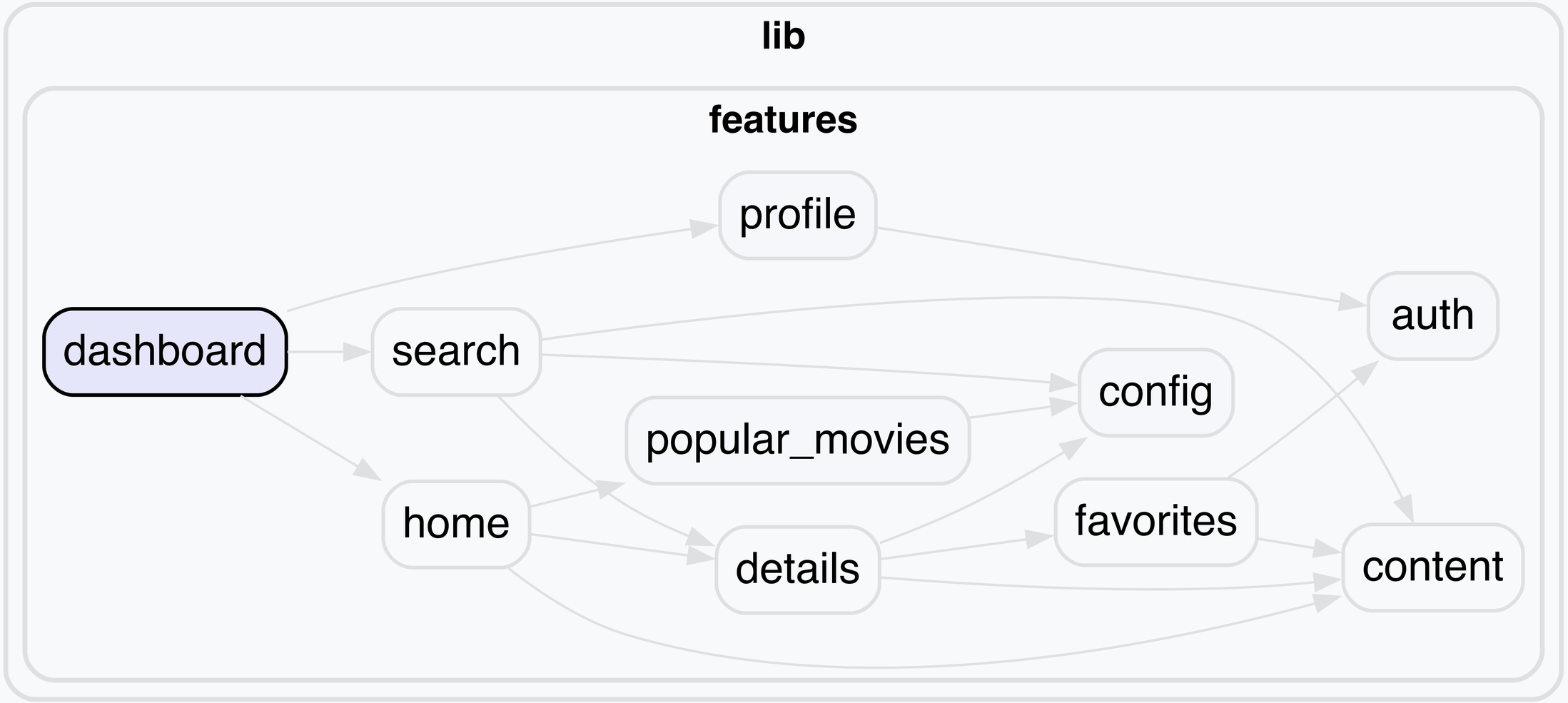
Stable components



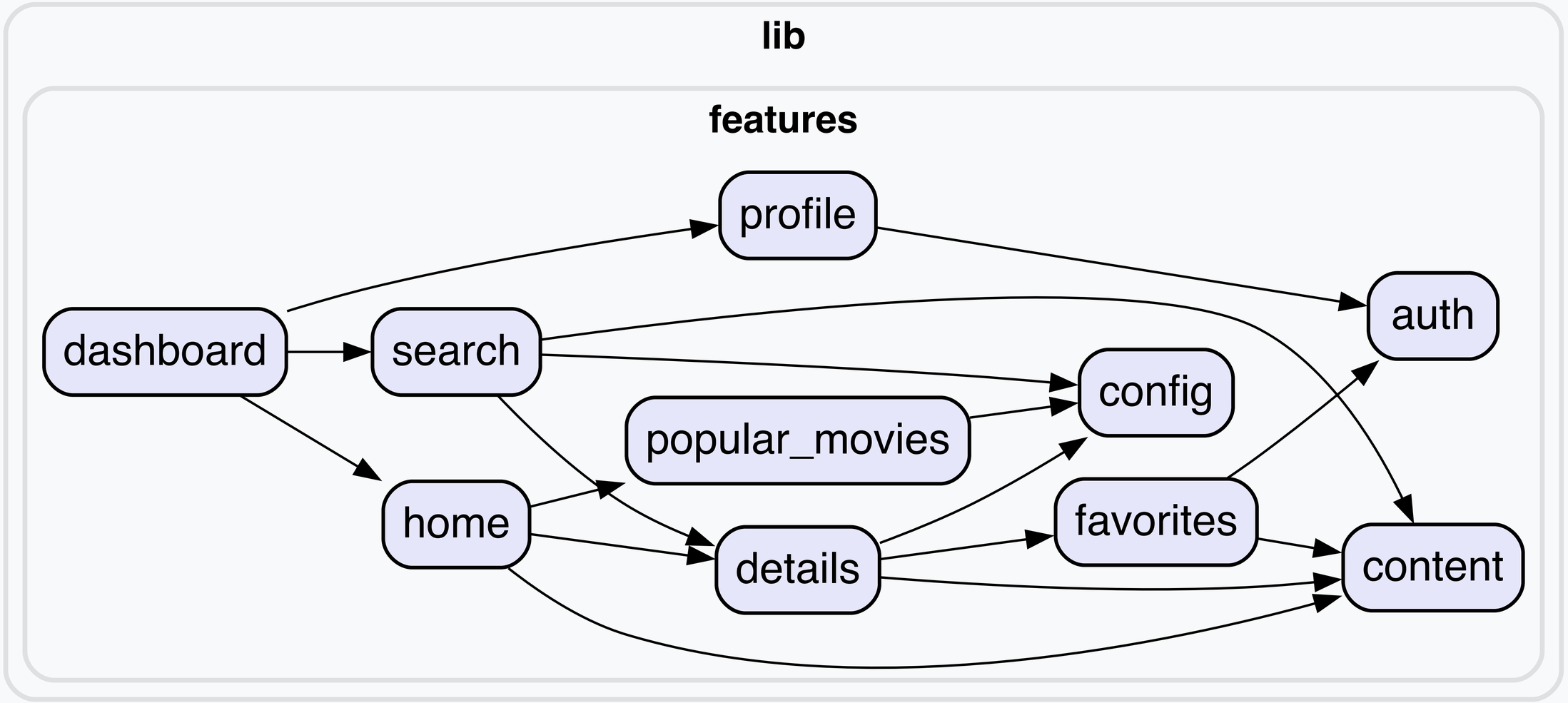
Stable components



Unstable components

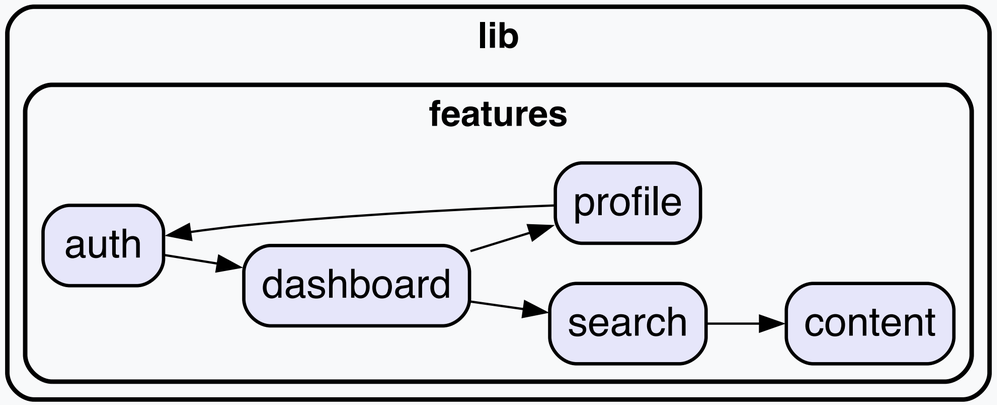


Unstable components

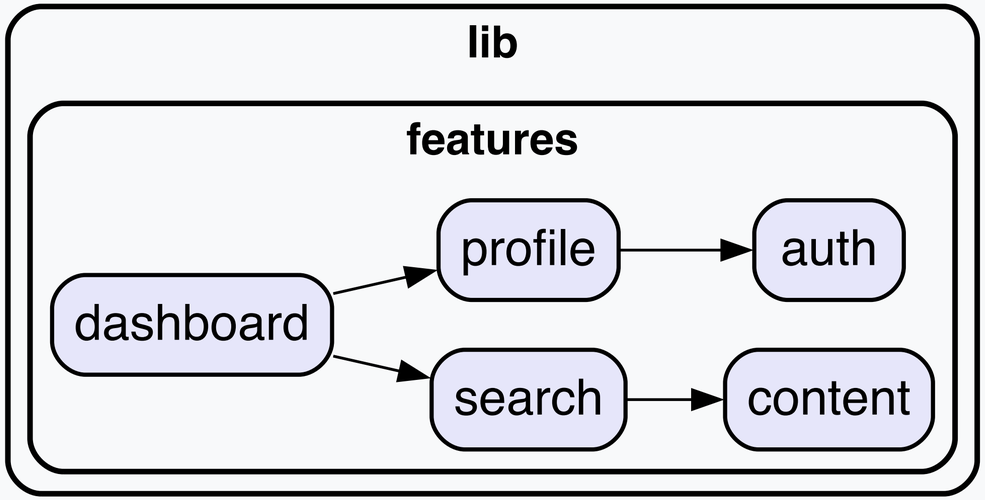


Directed Acyclic Graph

❌ Cycles

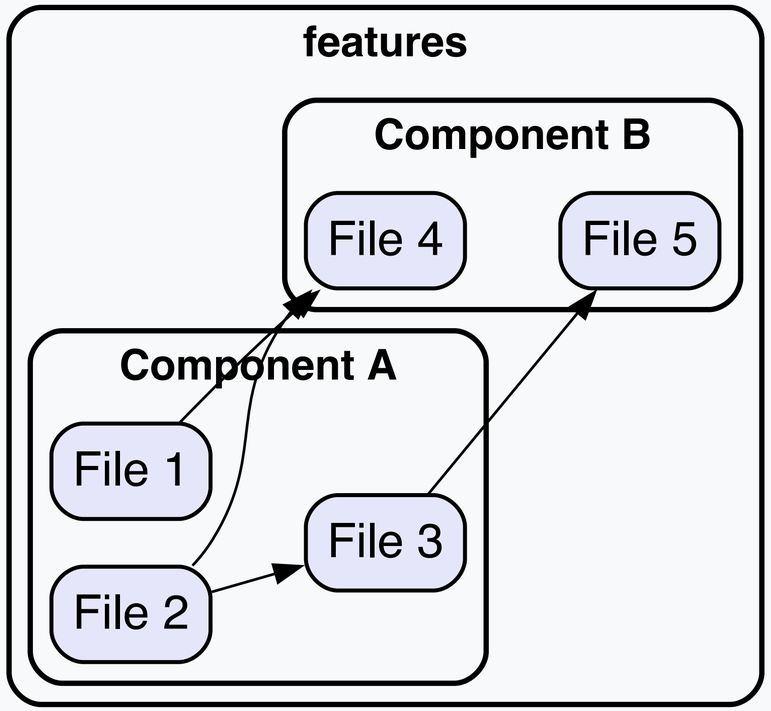


✅ No cycles

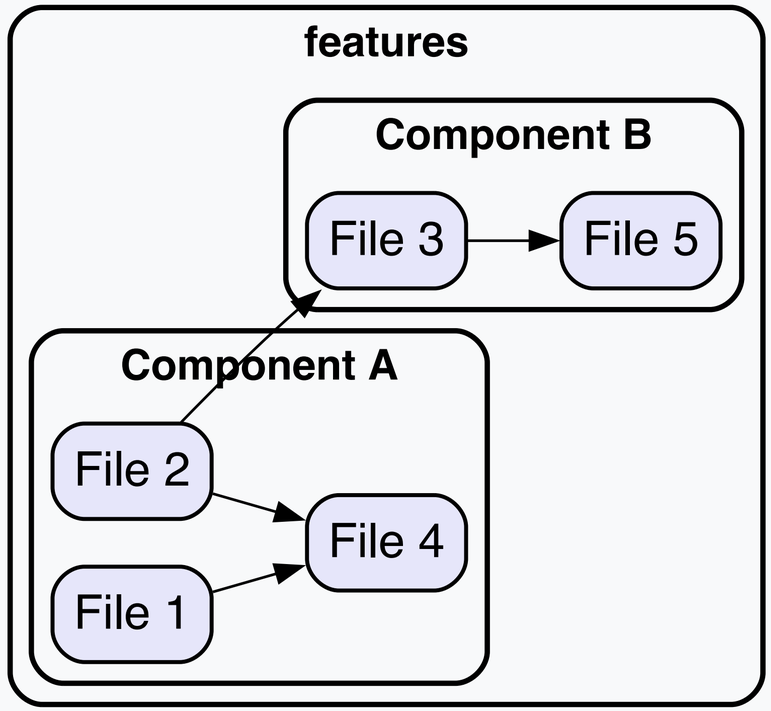


Loose coupling, high cohesion

❌ Coupling ↑ cohesion ↓



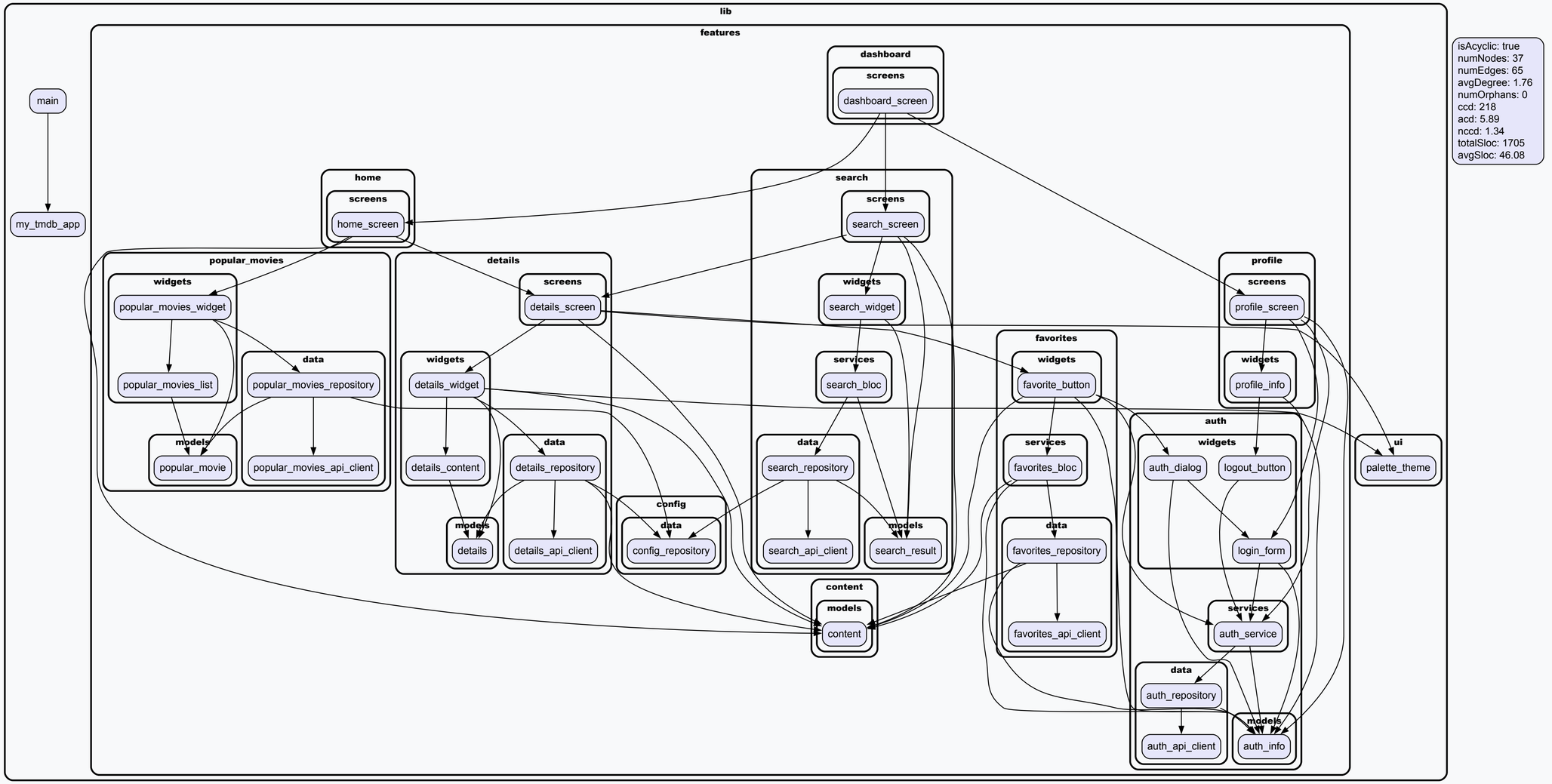
✅ Coupling ↓ cohesion ↑



Tools

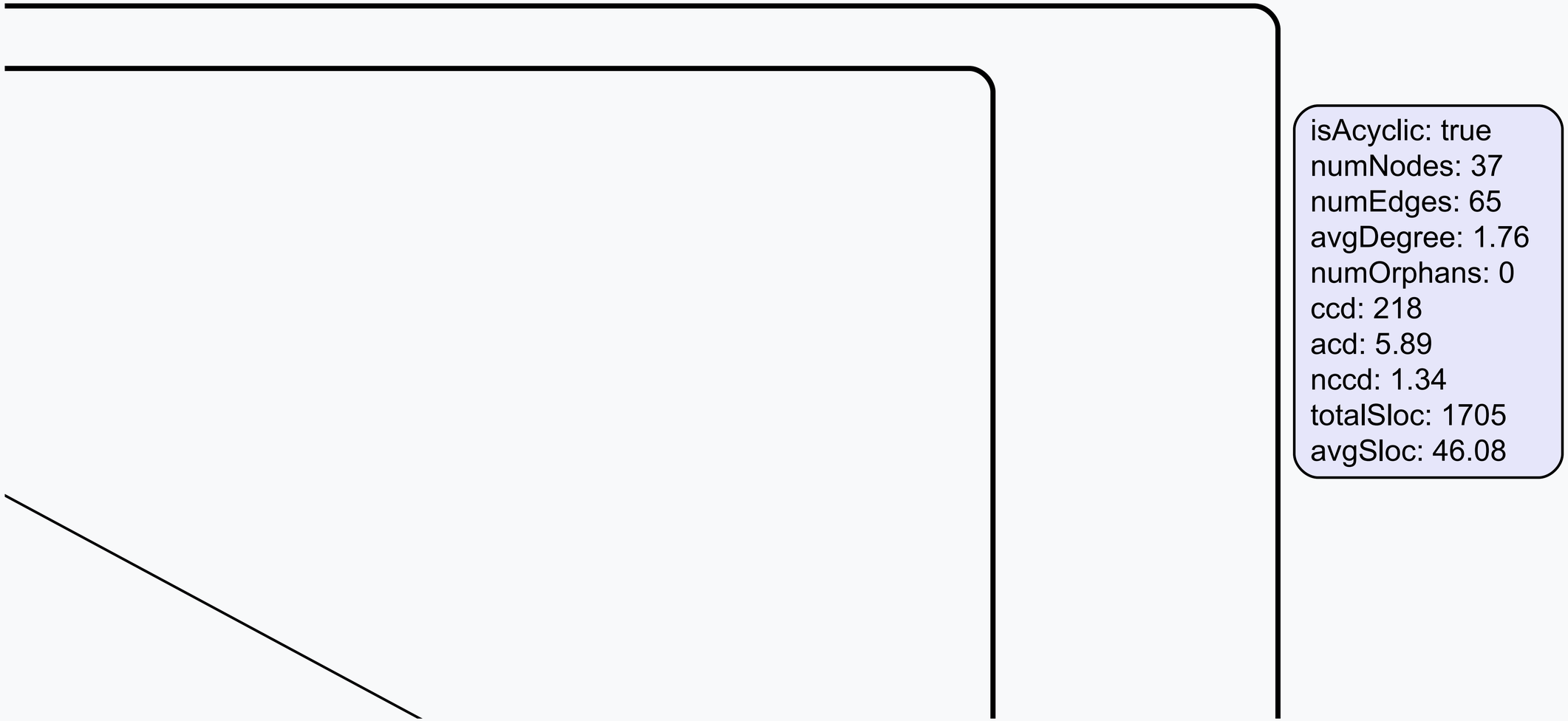
Tools

lakos


lib
features
dashboard
screens
dashboard_screen
main
my_tmdb_app
home
screens
home_screen
search
screens
search_screen
popular_movies
widgets
popular_movies_widget
popular_movies_list
data
popular_movies_repository
models
popular_movie
popular_movies_api_client
details
screens
details_screen
widgets
details_widget
details_content
data
details_repository
details_api_client
models
details
config
data
config_repository
widgets
search_widget
services
search_bloc
data
search_repository
search_api_client
models
search_result
favorites
widgets
favorite_button
services
favorites_bloc
data
favorites_repository
favorites_api_client
profile
screens
profile_screen
widgets
profile_info
auth
widgets
auth_dialog
logout_button
login_form
services
auth_service
data
auth_repository
auth_api_client
models
auth_info
ui
palette_theme
content
models
content
isAcyclic: true
numNodes: 37
numEdges: 65
avgDegree: 1.76
numOrphans: 0
ccd: 218
acd: 5.89
nccd: 1.34
totalSloc: 1705
avgSloc: 46.08

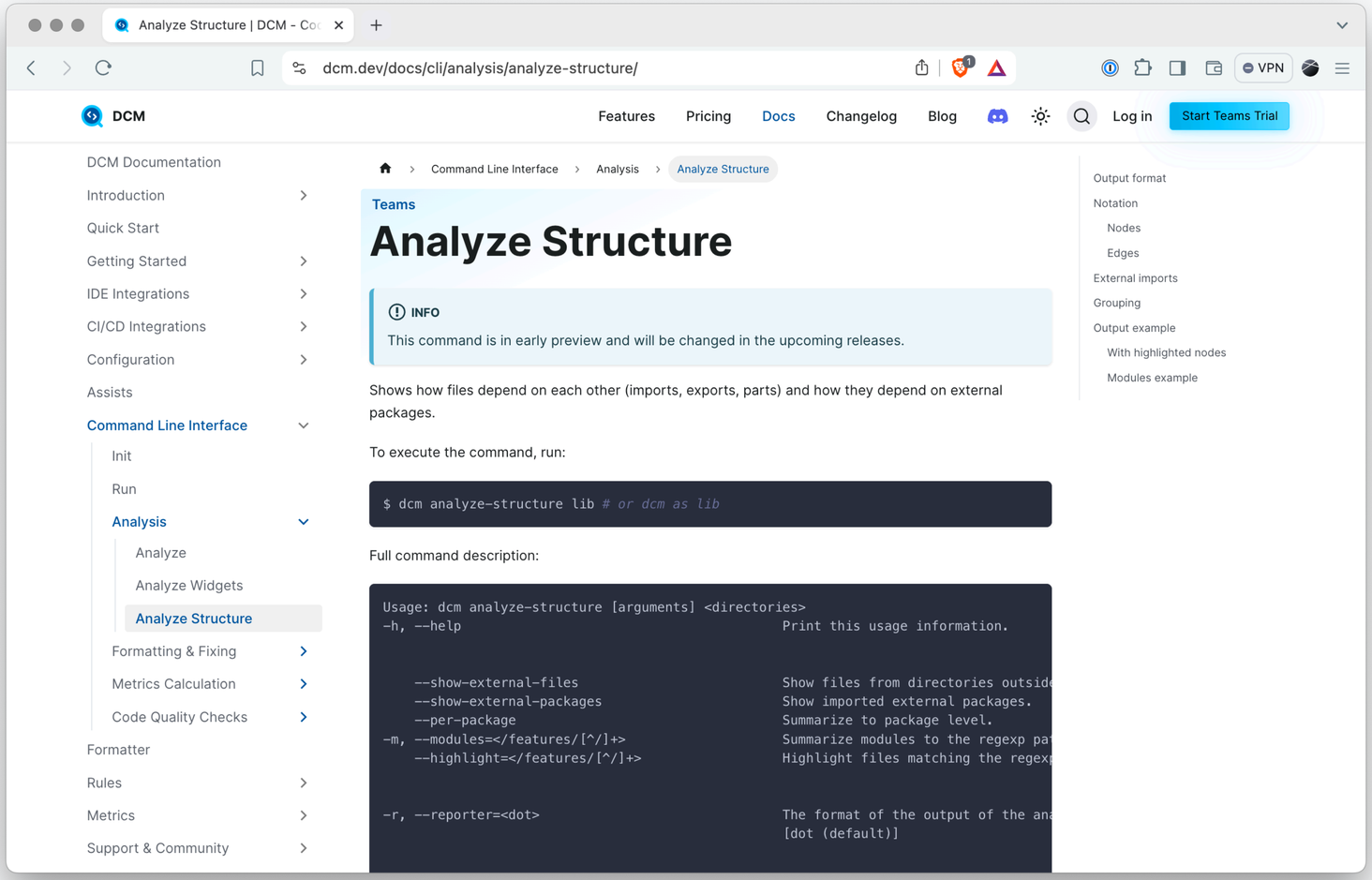
Tools

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

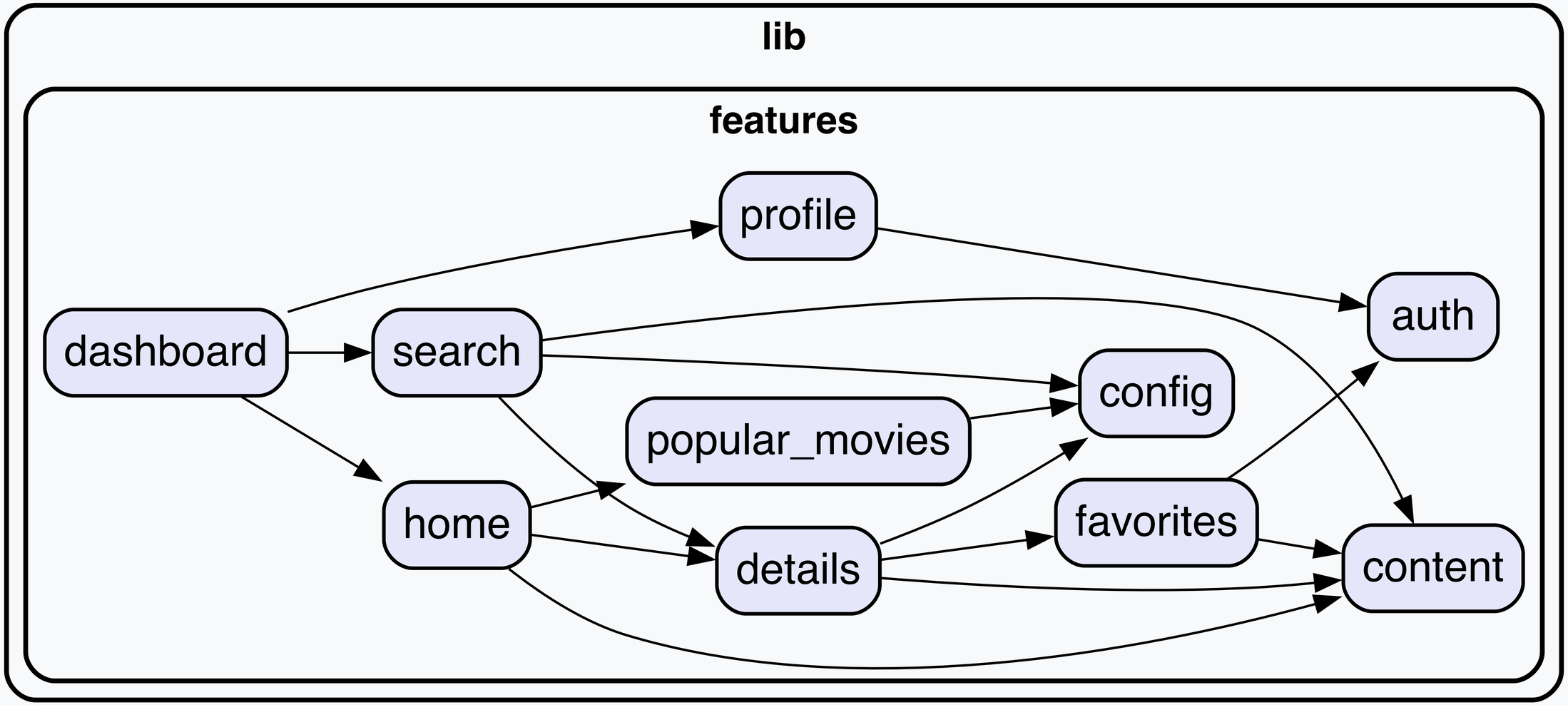
Tools

DCM


Analyze Structure | DCM - Co
dcm.dev/docs/cli/analysis/analyze-structure/
VPN
DCM
Features
Pricing
Docs
Changelog
Blog
Log in
Start Teams Trial
DCM Documentation
Introduction
Quick Start
Getting Started
IDE Integrations
CI/CD Integrations
Configuration
Assists
Command Line Interface
Init
Run
Analysis
Analyze
Analyze Widgets
Analyze Structure
Formatting & Fixing
Metrics Calculation
Code Quality Checks
Formatter
Rules
Metrics
Support & Community
Command Line Interface
Analysis
Analyze Structure
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 outside
--show-external-packages Show imported external packages.
--per-package Summarize to package level.
-m, --modules=</features/[^/]+> Summarize modules to the regexp pat
--highlight=</features/[^/]+> Highlight files matching the regex
-r, --reporter=<dot> The format of the output of the ana
[dot (default)]
Output format
Notation
Nodes
Edges
External imports
Grouping
Output example
With highlighted nodes
Modules example

Tools

DCM


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

Benefits

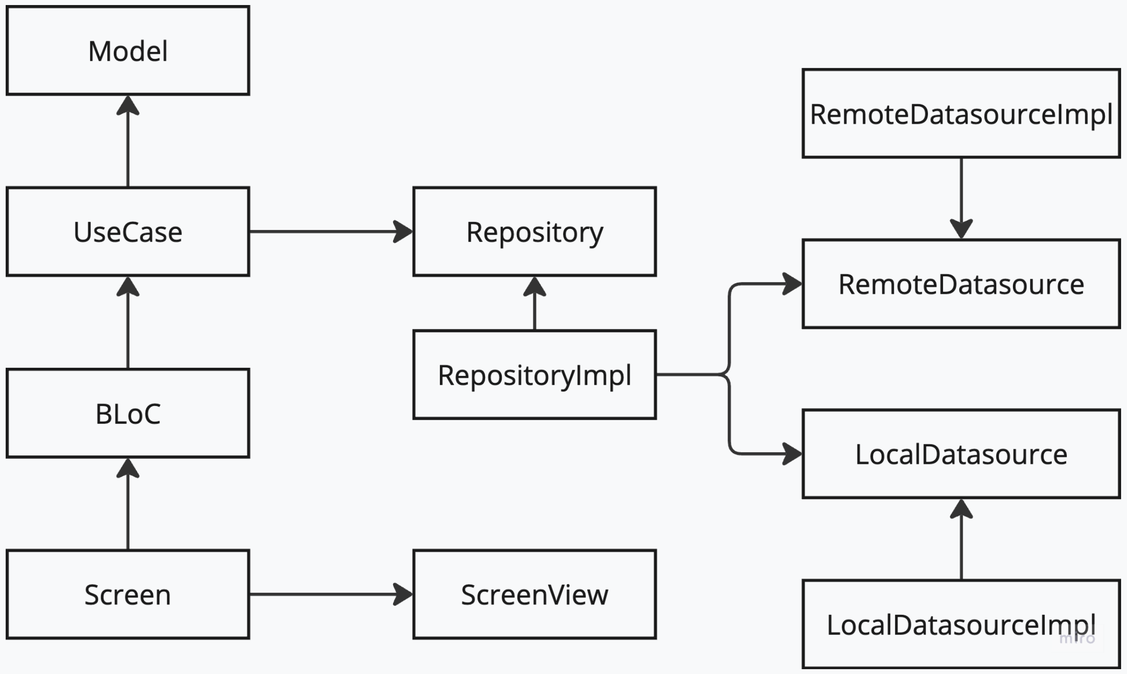
Benefits

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

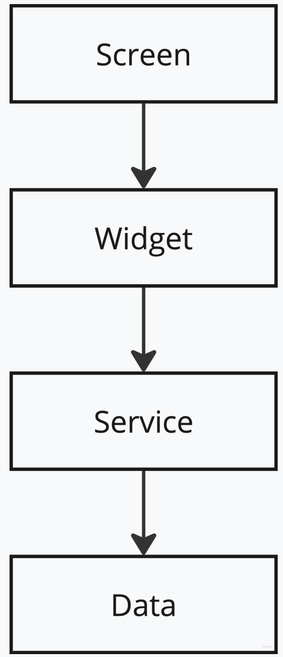
P.S. Is it really different from Clean Architecture?

Simplification

❌ "Traditional"

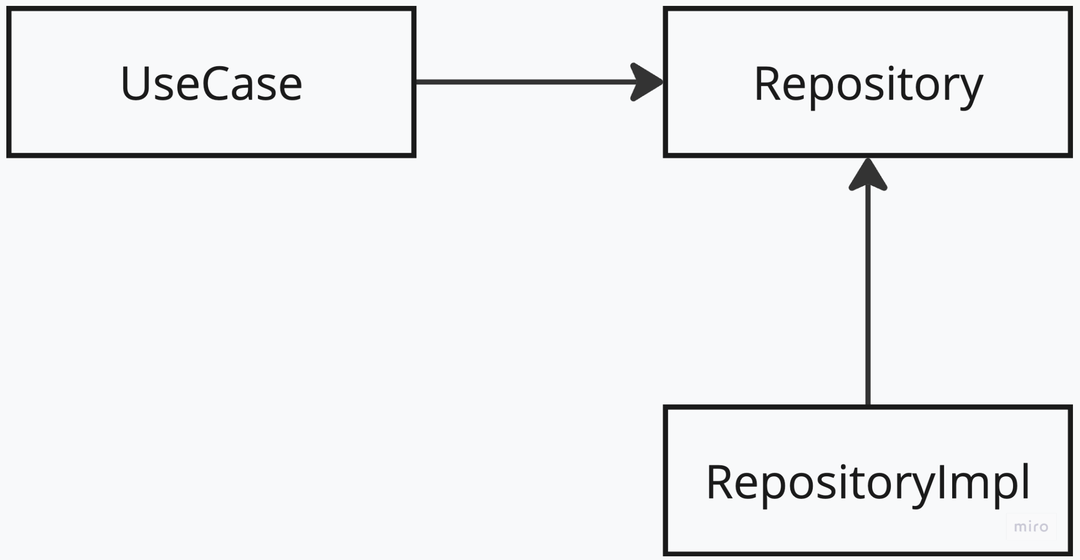


✅ Pragmatic



Inversion of Dependency Inversion

❌ "Traditional"



✅ Pragmatic

Also

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

Thank you!

linkedin.com/in/ookamikb/