



Global Knowledge®



# Angular Extended Using Multiple Modules

Peter Kassenaar –

[info@kassenaar.com](mailto:info@kassenaar.com)

WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR  
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA



# Multiple modules

Splitting your application into separate, reusable modules

# Default application – 1 module

The image shows the Angular CLI welcome screen on the left and a file explorer on the right, illustrating the default application structure.

**Angular CLI Welcome Screen:**

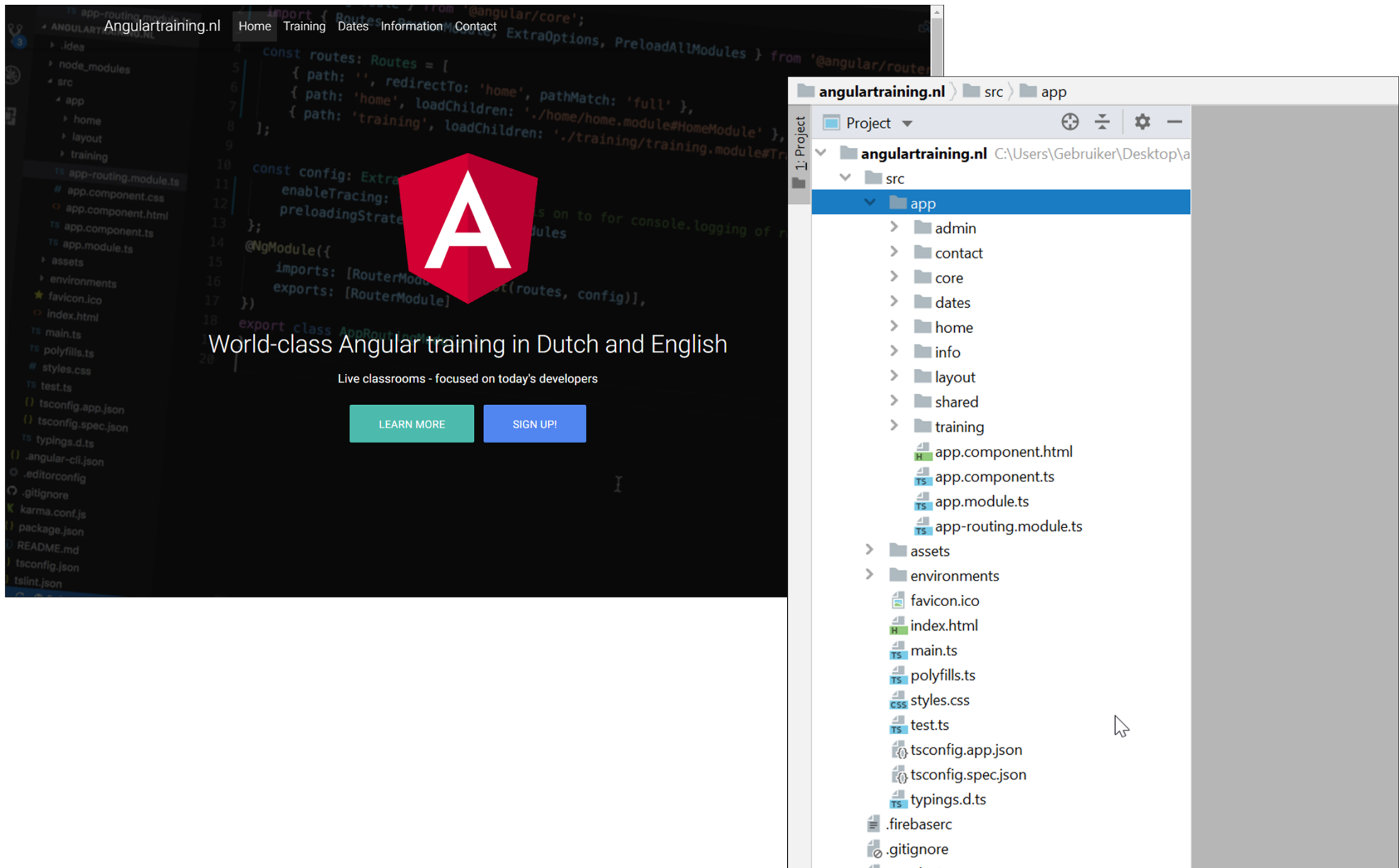
- Header: Welcome
- Notification: multiple-modules app is running!
- Resources: Here are some links to help you get started:
  - Learn Angular >
  - CLI Documentation >
  - Angular B...
- Next Steps: What do you want to do next with your app?
  - + New Component
  - + Angular Material
  - + Add Dependency
  - + Build for Production
- Terminal: `ng generate component xyz`
- Footer: Love Angular? Give our repo a star. ★ Star >

**File Explorer (customProject):**

- Project: customProject C:\Users\Peter Kassenaar\Desktop\custo
- Structure:
  - e2e
  - node\_modules library root
  - src
    - app
      - app.component.css
      - app.component.html
      - app.component.spec.ts
      - app.component.ts
      - app.module.ts
    - assets
      - .gitkeep
    - environments
      - environment.prod.ts
      - environment.ts
    - favicon.ico
    - index.html
    - main.ts
    - polyfills.ts
    - styles.css
    - test.ts
    - tsconfig.app.json
    - tsconfig.spec.json
    - typings.d.ts
  - .angular-cli.json
  - .editorconfig
  - .gitignore
  - karma.conf.js
  - package.json
  - protractor.conf.js
  - README.md
  - tsconfig.json
  - tslint.json
  - yarn.lock
  - External Libraries

(228 MB)

# Bigger applications – multiple modules



The image displays a screenshot of an Angular application project structure and a code editor. The code editor on the left shows the routing module configuration for 'angulartraining.nl'. The project structure on the right shows the 'src' directory with an 'app' subdirectory containing various modules and components.

**Code Editor (Left):**

```
import { Routes, RouterModule, ExtraOptions, PreloadAllModules } from '@angular/router';

const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: 'home', loadChildren: './home/home.module#HomeModule' },
  { path: 'training', loadChildren: './training/training.module#TrainingModule' }
];

const config: ExtraOptions = {
  enableTracing: true,
  preloadingStrategy: PreloadAllModules
};

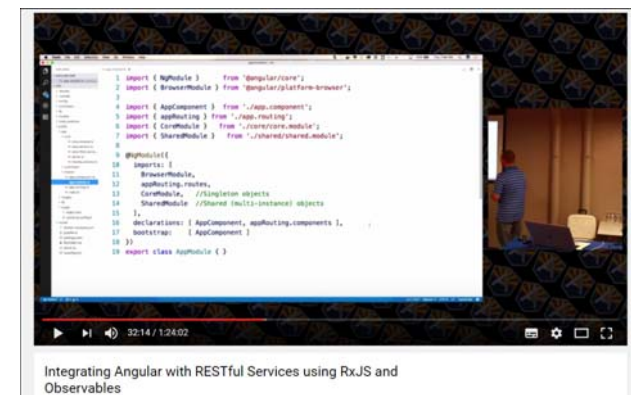
@NgModule({
  imports: [RouterModule.forRoot(routes, config)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

**Project Structure (Right):**

- angulartraining.nl
  - src
    - app
      - admin
      - contact
      - core
      - dates
      - home
      - info
      - layout
      - shared
      - training
      - app.component.html
      - app.component.ts
      - app.module.ts
      - app-routing.module.ts
    - assets
    - environments
    - favicon.ico
    - index.html
    - main.ts
    - polyfills.ts
    - styles.css
    - test.ts
    - tsconfig.app.json
    - tsconfig.spec.json
    - typings.d.ts
    - .firebaserc
    - .gitignore

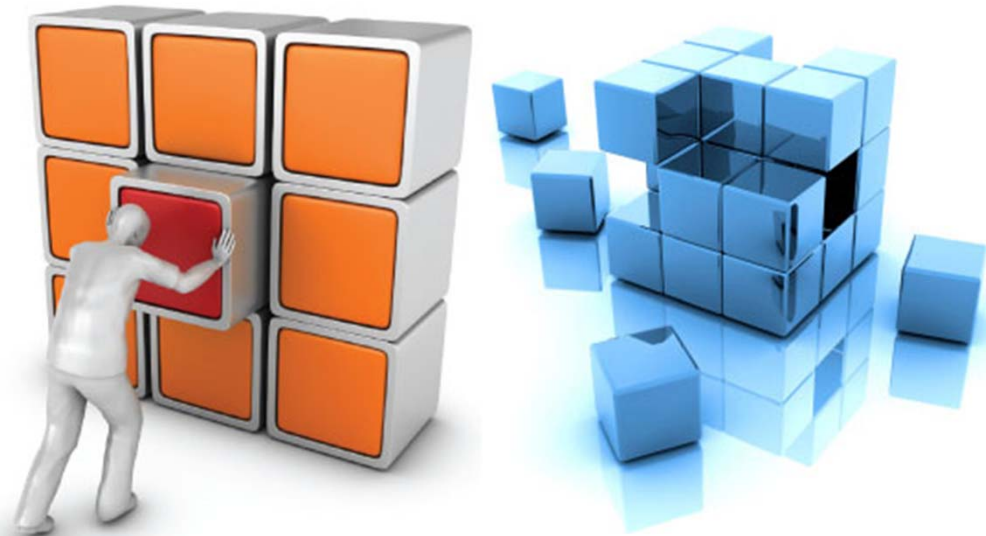
# Angular Modules

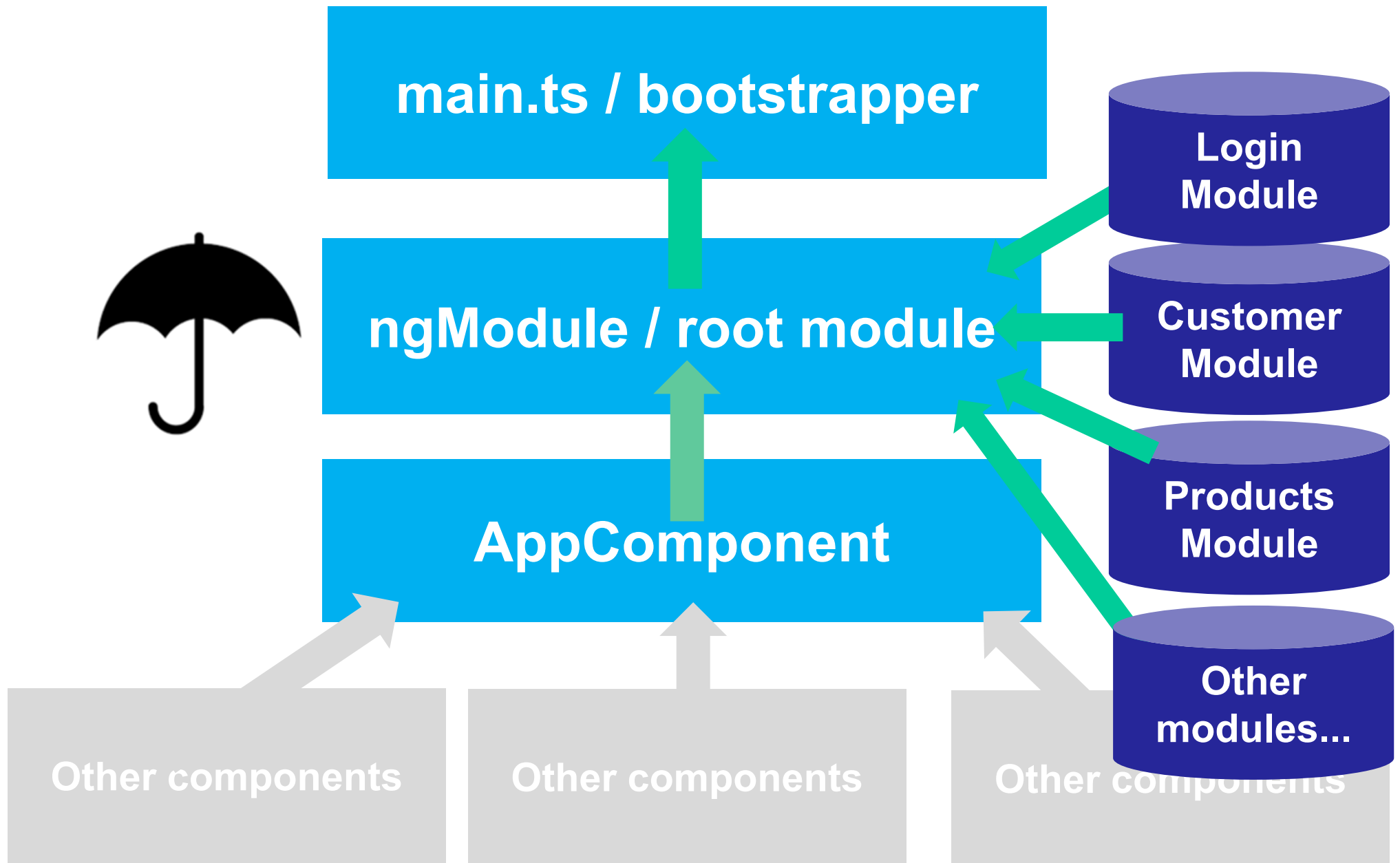
- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : **code organization**
- Use one AppModule - the root of your app
- Use one CoreModule - containing all *singletons* in your app
- Use one SharedModule - containing all shared resources, possible multiple instances
- Use additional modules *per feature*
- <https://www.youtube.com/watch?v=YxK4UW4UfCk>



# Application – multiple Modules – why?

- *Reuse* of Components, Pipes, Routes and Services etc. over different apps
- *Wrap* each set of logical related components, services, etc. in its own module.





# Steps

## 1. Create a new module

- Optional: test first with `--dry-run`
- `ng generate module customers --dry-run`

## 2. Create component(s) inside that module

- Again: test first with `--dry-run`
- `ng generate component customers --module customers --dry-run`

## 3. Apply UI, logic, etc. to your component

## 4. Export your component inside `customers.module.ts`

- `exports : [CustomersComponent],`
- Otherwise it can't be used in other components!

## 5. Provide new module to `app.module.ts`

- `imports: [CustomersModule]`



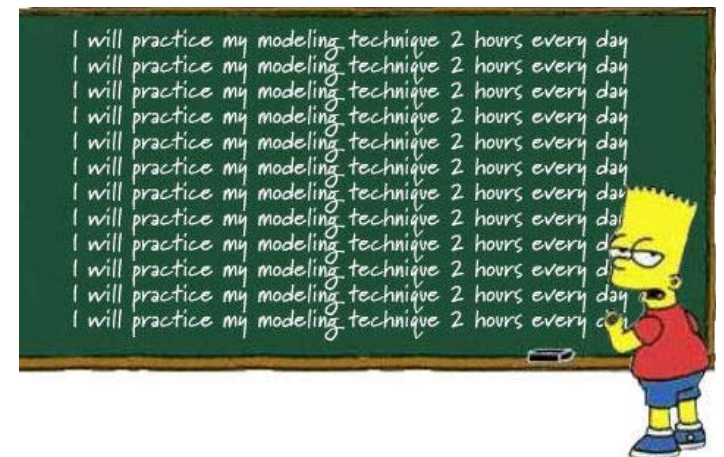
# Optional : SharedModule

- Reuse components in multiple modules? Use a SharedModule
  - `ng g m shared` – shorthand notation
- Create components inside SharedModule
- Import SharedModule in other modules
- It doesn't have to be in AppModule if you don't use it directly!
- It *does* not add size to module bundles



# Workshop

- Open ../100-multiple modules.
- Create a new module
- Create a new component inside this new module and give it some UI.
- Include the module in the Main Module and show it besides other modules
- Include the Search Component in your own module
- *OR:*
- Add Multiple Modules from scratch to your own application, using the steps described in this module.



# How to structure feature modules



242



## Why and how to structure Features in Modules in Angular

This might sound pretty basic, but I encounter these challenges over and over in customer projects and it's still an ongoing discussion internally.

A central project goal in a recent Angular project was to design features and UI components for reusability. To achieve this, we need to make sure our code is well isolated and has a simple and clear dependency model.

### Prologue: Feature vs. Technical Project Structure

When building small apps and looking at common code samples in the internet a lot of devs (including myself) tend to come up with a project structure like this:

```
MYAPP
├── src
│   ├── app
│   │   ├── components
│   │   │   ├── home
│   │   │   │   ├── home.component.html
│   │   │   │   ├── home.component.ts
│   │   │   └── user
│   │   │       ├── user.component.html
```

<https://medium.com/@philippbauknecht/why-and-how-to-structure-features-in-modules-in-angular-d5602c6436be>