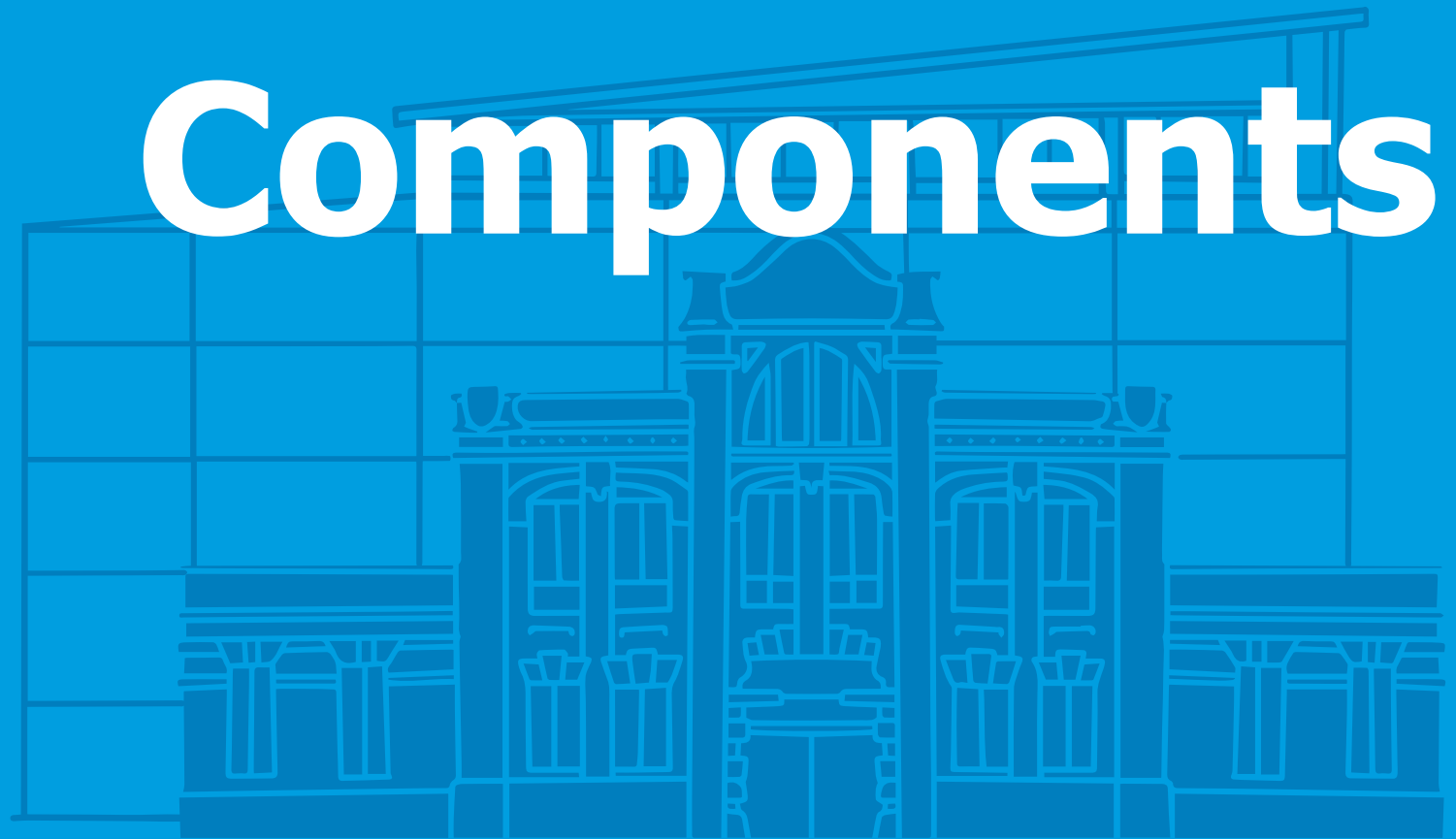
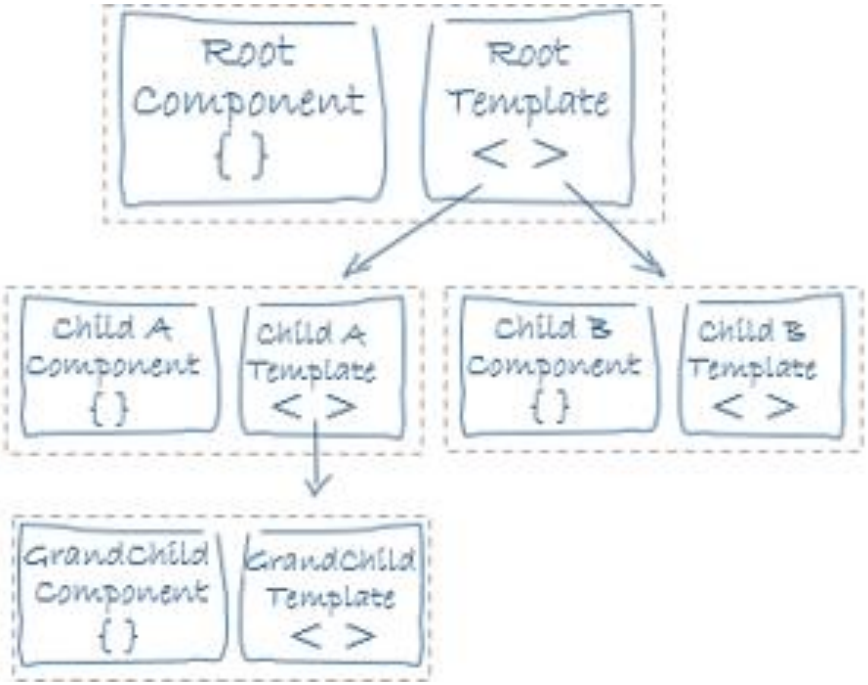
netmind
a bts company

Angular 14 - 05

Components



Angular components

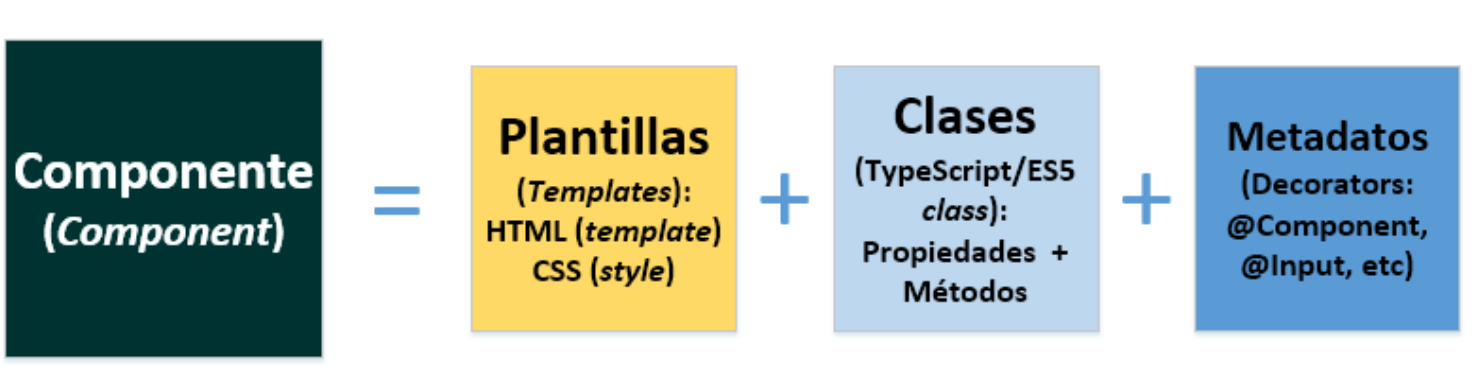


- Components are the **most basic UI building block** of an Angular app. An Angular app contains a tree of Angular components.
- Angular components are a **subset of directives**, always associated with a template.
- Unlike other directives, **only one component** can be instantiated for a given element in a **template**.
- A component **must belong to an NgModule** in order for it to be available to another component or application. To make it a member of an NgModule, list it in the declarations field of the NgModule metadata.

https://angular.io/api/core/Component

Structure of a component

- Let's recall the component scheme:



- The template can contain "*bindings* " to bind to data stored in the class (using *mustache syntax* **{{data}}**) as well as other directives defined by Angular.
- A class, which provides the data that the UI and the functional part must link (it has properties and methods).
- Metadata (***decorators***), which configure the behavior of both and their relationship to each other, and registers them in the "engine" of Angular.

Adding a component

🔗 **Using the CLI**

```
ng g component [name]

ng g component [relative_path]/[name]

Ng g component [name] –m [module-name]
```

This command will do several things:
1. It will create a folder with the name of the component (CamelCase)
2. It will create the ts component, its template, its css and its test
3. It will add the component to the main module

🔗 **For more options:**

```
ng g component --help
```

Declaring the new component in its module

The declarations are separated to the module that handles the main component, so that the main module looks like this:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
import { ProductsListComponent } from './product/products-list/products-list.component';

@NgModule({
  declarations: [
    AppComponent,
    ProductsListComponent,
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

@Component

🔗 Decorator that marks a class as an Angular component and provides configuration metadata that determines how the component should be processed, instantiated, and used at runtime.

Properties:

- changeDetection?
- viewProviders?
- moduleId?
- **templateUrl?**
- template?
- **styleUrls?**
- styles?
- animations?
- encapsulation?
- interpolation?
- entryComponents?
- preserveWhitespaces?
- **standalone?**
- imports?
- schemas?
- **selector?**
- **inputs?**
- **outputs?**
- **providers?**
- exportAs?
- queries?
- host?
- jit?

https://angular.io/api/core/Component

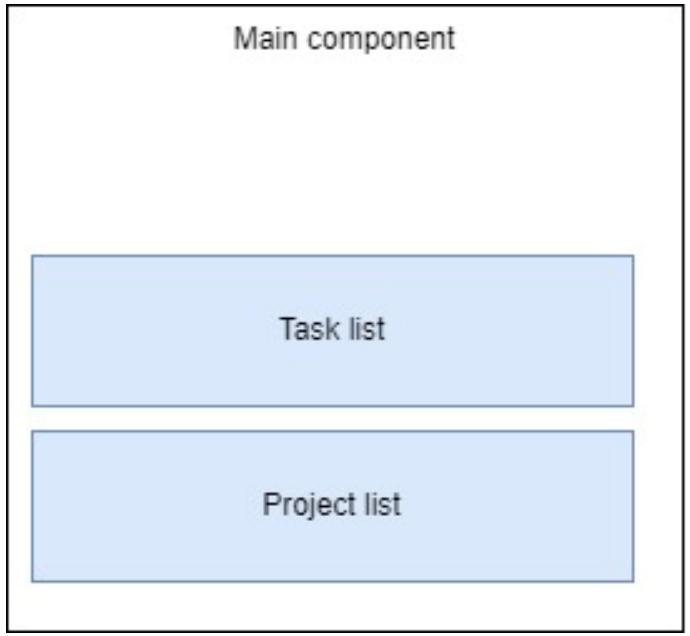
OBJECTIVE

Let's put it into practice: Tasks/Projects App

INSTRUCTIONS

Create two components in the Tasks and Projects app:

1. The first component should display the text "Task List"
2. The second component should display the text "List of Projects"



Standalone components

- A standalone component is a type of component which is not part of any Angular module.
- Starting with Angular 14, you can create the whole application without making any custom Angular modules.
- Angular 14 even allows you to bootstrap the whole application using a standalone component.

🔗 You can **create a standalone component**, pipe or directive by using the --standalone flag in the ng generate component command:

```
ng g c [component_name] --standalone

ng g p [pipe_name] --standalone

ng g d [directive_name] –standalone
```

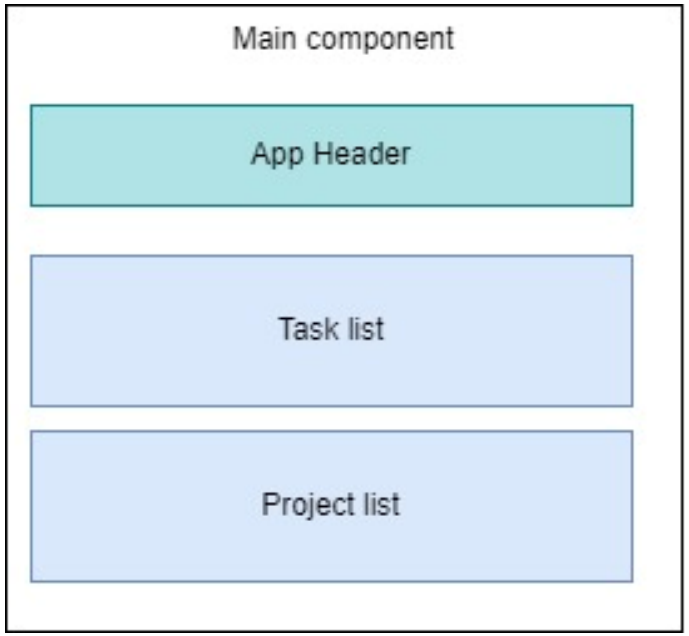
🔗 Dependencies can be added to a standalone component using the **imports array** of the @Component decorator.

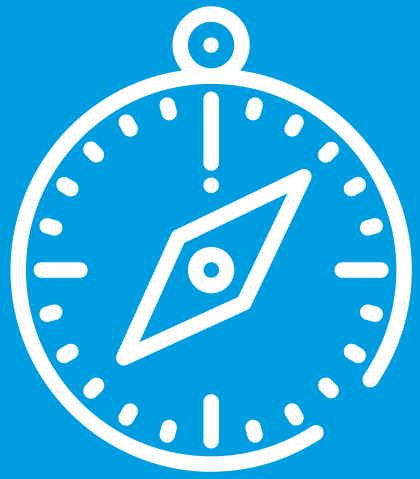
OBJECTIVE

Let's put it into practice: Tasks/Projects App

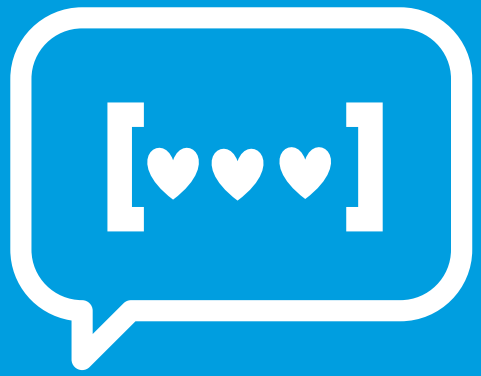
INSTRUCTIONS

Create a standalone component in the Tasks and Projects app, name it “app-header”





Next steps



We would like to know your opinion!

Please, let us know what you think about the content.
From Netmind we want to say thank you, we appreciate time and effort you have taking in answering all of that is important in order to improve our training plans so that you will always be satisfied with having chosen us
quality@netmind.es

netmind
a bts company

Thanks!

Follow us: