<https://www.itsolutionstuff.com/post/angular-13-crud-application-example-tutorialexample.html>

# Intorudcció

A aquest capítol es veurà com crear una apliació **CRUD** amb **Ionic**/**Angular** pas a pas. La versió d’**Angular** alhora de fer aquest capítol és la 13, i la d’**Ionic** la 5.4.

A l’aplicació es crearà im módul post amb CRUD per mostrar un llistat , una vista, inserció de post, actualització i eliminació. Per a tot això s’utilitzarà un **service** (servei). Això facilitarà la utilització de la **API** i d’un fitxer **JSON** amb valors predefinits.

Per a realitzar l’a`licació necessitarem crear les següents pàgines:

* Home Page (Pàgina d’inicia).
* Create Page (Creació d’items).
* Edit Page (Pàgina d’edició).
* Detail Page (Pàgina de detalls).

# Crear una nova aplicació

El primer pas serà la creació d’un nou projecte; en a aquest cas un projecta amb IOnic/Angular. Per això obrirem un terminal i escriurem el següent comandament:

$ng new my-crud-app --routing

Si l’aplicació es vol vrear amb Ionic escriurem

$ionic start my-crud-app blank –type=angular

# Important Bootstrap

El següent pas serà importarem el paquet de Boootstrap mitjançatn el següent comandamtent

$npm install bootstrap –save

D’aquesta forma podrem utilitzar les opcions de Bootstrap alhora de crear la vista de l’aplicació[[1]](#footnote-1).

Un cop hem instal·lat Bootstrap, l’hem d’importa. Per això obrirem el fitxer **styles.css** i afegirem la següent.

@import "~bootstrap/dist/css/bootstrap.css";

# Creació d’un Mòdul

Ja tenim creada la base de la nostre aplicació i on començarem a desenvolupar l’aplicació CRUD. Si tot ha anat bé, podem observar l’aspecte actual de l’apliació a un navegador d’internet, per fer això escriurem el següent comandament a un terminal.

$ng serve

Un cop introduït el comandament, si obrim un navegador i anem a l’adreça localhost:4200 veurem la pàgina per defecte que s’ha creat.

Per crear el mòdul post obrirem un nou terminal i insertarem el comandament per a la creació de mòduls que en aquest cas és\_

$ng g module post –routing

So el comandament no ha fallat haurem creat un nou directori anomenat **post** amb dos fitzers al seu interior: **post.module.ts** i **post-routing.module.ts**.

# Crear un component pel nou mòdul

Amb el nou mòdul creat, anem a afegir-hi un nou component. Per això tornarem al terminal i escriurem el següents comandaments:

$ng g component post/index

$ng g component post/view

$ng g component post/create

$ng g component post/edit

Aquests comandaments ens hauran creat un nou directori per a cada component dins de la carpeta **post**.

# Creant les rutes

En aquesta ocasió anem a afegir les rutes per a poder enllaçar amb els nous fitxers que acabem de crear. Per això s’ha d’actualitzar el fitxer **post-routing.module.ts**. Obrim aquest arxiu i el modifiquem de la següent manera:

* Importem els components.

//Components

import { IndexComponent } from './index/index.component';

import { ViewComponent } from './view/view.component';

import { CreateComponent } from './create/create.component';

import { EditComponent } from './edit/edit.component';

* Importem els **paths** (camins) a l’array **Routes**.

const routes: Routes = [

  { path: 'post', redirectTo: 'post/index', pathMatch: 'full' },

  { path: 'post/index', component: IndexComponent },

  { path: 'post/:postId/view', component: ViewComponent },

  { path: 'post/create', component: CreateComponent },

  { path: 'post/:postId/edit', component: EditComponent },

];

# Crear una interfase

El proper pas és la creació d’una interfase utilitzant el terminal. Així doncs anem al terminal i escribim el següent comandament.

$ng g interface post/post

Amb això haurem creat un nou arxiu, obrim el nou fitxer **post.ts**. i modifiquem el seu contingut per.

export interface Post {

    if: number,

    title: string,

    body: string,

}

# Crear un servei

Ara hem de crear un servei on escriurem els mètodes de trucada al web service: **getAll( )**, **create( )**,**find( )**, **update( )** i **delete( )**.

Estem utilitzant l’API <https://jsonplaceholder.typicode.com>, temporalment, que es molt senzilla d’utilitzar. Però l’objectiu seria crear el servei web amb un code **PHP** propi. Això ho podem consultar a les següents webs:

[Create Web Service using PHP Laravel](https://www.itsolutionstuff.com/post/laravel-6-rest-api-with-passport-tutorialexample.html)

[Create Web Service using PHP Codeigniter](https://www.itsolutionstuff.com/post/codeigniter-3-restful-api-tutorialexample.html)

Un cop dita això, anem a crear el servei post i afegir el codi del mètodes.

1. Ionic ja incorpora Bootstrap per defecte. [↑](#footnote-ref-1)