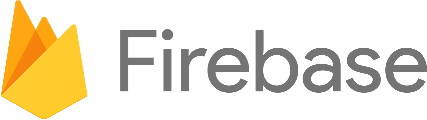
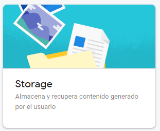
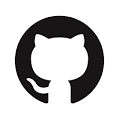
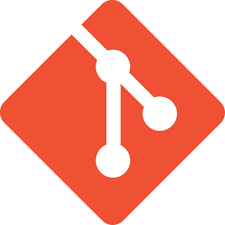
[](https://angular.io/)

<https://angular.io/>

Una webapp a través del Framework ANGULAR amb:

1. [Google Firebase](https://firebase.google.com/):
   1. Realtime Database
   2. Storage
2. [Github](https://github.com/) pel [repositori](https://github.com/MarcEsteve/porfolio-angular-2023)
3. Github PAGES per penjar la [webapp a un subdomini](https://marcesteve.github.io/porfolio-angular-2023/#/about)
4. Una plantilla de HTML5 para el proyecto de Angular, se llama URKU y está en [PIXEDEN](https://www.pixeden.com/html5-website-templates/urku-html5-portfolio-website)

C:\Users\Marc\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\2E6590D8.tmpINSTAL·LACIONS (inicials):

* [Visual Studio Code](https://code.visualstudio.com/download)
* Navegador (Chrome, [Brave](https://brave.com/download/), Firefox...)
* [Git](https://git-scm.com/downloads) (pel control de versions)
* [Node.js](https://nodejs.org/)
* Terminal (Símbol del sistema o Consola ja instal·lat al SO)
  + Instal·lar TypeScript "*npm install -g typescript*"
  + Instal·lar Angular "*npm install -g @angular/cli*"

+ **Extensiones** de Visual Studio Code:

* [Angular Snippets](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2)
* [TypeScript Importer](https://marketplace.visualstudio.com/items?itemName=pmneo.tsimporter)

(nou material cominng soon)

Creant rutes per a la web app d’Angular

Navegació de la pàgina web mostra de moment els components que els hi hem deixat actius, en aquest cas l’últim creat de ***item***:

<app-header></app-header>

<section class="ae-container-fluid rk-main">

  <!-- <app-portafolio></app-portafolio> -->

  <!-- <app-about></app-about> -->

  <app-item></app-item>

</section>

<app-footer></app-footer>

Creem a la carpeta “app” el nou arxiu de configuració del “mòdul”:

*app-routing.module.ts*

Mòdul encarregat de l’especificació de les rutes, cada pàgina que està al menú anirà en aquest mòdul. Es fa a un mòdul per no carregar el “app.module.ts” general. Al estàndard d’Angular es recomana fer un mòdul a part per a que sigui més fàcil de mantindre.

*PAS 1*

A Angular gairebé tot son **classes**, llavors afegim:

“class AppRouting {}” però com volem que sigui utilitzat fora hem d’afegir “export”.

export class AppRoutingModule { //Mòdul per a la navegació

}

*PAS 2*

Per a que això sigui un mòdul necessitem un **decorador**. Podem revisar el mòdul principal ja creat per defecte “app.module.ts” per revisar tot el necessari.

Escrivim @NgModule per importar el decorador.

@NgModule ({

    //Aquí dins va la configuración que donarem a AppRoutingModule

})

Ens hauria d’importar a la part superior:

import { NgModule } from "@angular/core";

*PAS 3*

Abans de començar amb el mòdul que tenim llest i servit, haig d’especificar les rutes de la navegació. Vull donar informació de com volem que funcionin aquestes rutes, crearem una constant app\_routes de tipus **Routes**, llavors importarem aquest tipus.

import { Routes } from "@angular/router"; //Es crea amb autocompletar

const app\_routes: Routes = [

    { path: '', component: PortafolioComponent}

];

Conforme anem afegint components els haurem d’importar (també automàticament amb els auto completar)

import { PortafolioComponent } from './pages/portafolio/portafolio.component';

La resta de path, per totes les direccions del menú de navegació d’aquestes rutes.

const app\_routes: Routes = [

    { path: '', component: PortafolioComponent}, //Ruta principal

    { path: 'about', component: AboutComponent},

    { path: 'item', component: ItemComponent},

    { path: '\*\*', pathMatch: 'full', redirectTo: ''} //Totes les altres rutes que no existeixin a les anteriors

];

\*Importar també RouterModule (afegint una coma i RouterModule a la importació de Routes

*PAS 4*

Revisem el app.modules.ts per veure que obtindrem de @NgModule, i veiem el ***declarations*** que será pels *pipes* i els *components* principalment. La part dels mòduls va a ***imports***.

Llavors quedarà així:

@NgModule ({

    imports: [

        RouterModule.forRoot( app\_routes)

    ]

})

I per tant, aquest mòdul finalment l’haurem d’enllaçar amb el app.module.ts, precisament a imports, després del BrowserModule separat de coma i automàticament es farà un import d’aquest AppRouting Module.

L’arxiu final de **app-routing.module.ts**

import { NgModule } from "@angular/core";

import { Routes, RouterModule } from "@angular/router";

import { AboutComponent } from "./pages/about/about.component";

import { PortafolioComponent } from './pages/portafolio/portafolio.component';

import { ItemComponent } from './pages/item/item.component';

const app\_routes: Routes = [

    { path: '', component: PortafolioComponent}, //Ruta principal

    { path: 'about', component: AboutComponent},

    { path: 'item', component: ItemComponent},

    { path: '\*\*', pathMatch: 'full', redirectTo: ''} //Totes les altres rutes que no existeixin

];

@NgModule ({

    imports: [

        RouterModule.forRoot( app\_routes)

    ]

})

export class AppRoutingModule {

}

Tot i així no hem acabat, comprovem al navegador l’app corrent i tot i posar be les rutes segueix sense mostrar el contingut. Això passa perquè enlloc li hem dit que renderitzi el contingut, ni renderitzi aquests components. Recordem que teníem només l’item mostar, la resta comentat. Afegim llavors <router-outlet></ router-outlet> comentant tots els components (portfoli, about i item) i revisem que ens mostra l’error.

<app-header></app-header>

<section class="ae-container-fluid rk-main">

  <!-- <app-portafolio></app-portafolio> -->

  <!-- <app-about></app-about> -->

  <!-- <app-item></app-item> -->

  <router-outlet></router-outlet>

</section>

<app-footer></app-footer>

Llavors afegim el nostre RouterModule al **exports** de **app-routing.module.ts** i amb això podem utilitzar for a d’aquesta configuració aquest mòdul.

@NgModule ({

    imports: [

        RouterModule.forRoot( app\_routes)

    ],

**exports: [**

**RouterModule**

**]**

})

I ja hauria de marxar l’error i funcionar els path correctes, és a dir, la principal que son el llistat de ítems Masonry, el about i el ítem, tot i que fa una actualització de la pàgina totalment i no es la forma que volem que carreguin. Faltarien els menús correctament del <header>

**ENLLESTIM MENÚ DE NAVEGACIÓ**

Al nostre header.component.html hem d’actualitzar només els enllaços que volem utilitzar i funcionin correctament a la seva navegació amb les rutes creades.

El primer que canviem per exemple son els **href** per ***routerLink***.

<h1 class="rk-logo"><a routerLink="">App Angular<sup>tm</sup></a></h1>

Deixem buit l’enllaç del titular 1 perquè així anirà al nostre principal de portafolis. El mateix a HOME, podem canviar a Inici per exemple i anirà al mateix enllaç:

<li class="active rk-menu\_\_item">

    <a routerLink="" class="rk-menu\_\_link">Inici</a>

</li>

Ja no fan “full refresh” de tota la pàgina i per cert, hauríem d’afegir la class “**active**” per a quan estigui seleccionat cada ruta del menú de navegació. Però ho farem al <li> no al <a> i a més a més per a que funcioni posarem en comptes del link buit “” a tot l’inici posarem “inici” i al app-routing.modules.ts també a la constant app\_routes.

<div class="ae-container-fluid rk-topbar">

      <h1 class="rk-logo"><a routerLink="inici">App Angular<sup>tm</sup></a></h1>

      <nav class="rk-navigation">

        <ul class="rk-menu">

          <li routerLinkActive="active" class="rk-menu\_\_item">

            <a routerLink="inici" class="rk-menu\_\_link">Inici</a>

          </li>

          <li routerLinkActive="active"  class="rk-menu\_\_item">

            <a routerLink="about" class="rk-menu\_\_link">About</a>

          </li>

        </ul>

        <form class="rk-search">

          <input type="text" placeholder="Search" id="urku-search" class="rk-search-field">

          <label for="urku-search">

            <svg>

              <use xlink:href="assets/img/symbols.svg#icon-search"></use>

            </svg>

          </label>

        </form>

      </nav>

    </div>

Farem el mateix al footer.component.html (veure Notes del Box de “Recursos Online” per trobar SVG de qualsevol logo social per exemple LinkedIn o Github: <https://iconmonstr.com/>)

Exemple de FOOTER

<footer class="ae-container-fluid rk-footer ">

    <div class="ae-grid ae-grid--collapse">

      <div class="ae-grid\_\_item item-lg-4 au-xs-ta-center au-lg-ta-left">

        <ul class="rk-menu rk-footer-menu">

          <li class="rk-menu\_\_item"><a routerLink="inici" class="rk-menu\_\_link">Inici</a>

          </li>

          <li class="rk-menu\_\_item"><a routerLink="about" class="rk-menu\_\_link">About</a>

          </li>

        </ul>

        <p class="rk-footer\_\_text rk-footer\_\_copy "> <span class="ae-u-bold">© </span><span class="ae-u-bolder">{{year}} App Angular </span>Creative Commons</p>

      </div>

      <div class="ae-grid\_\_item item-lg-4 au-xs-ta-center">

          <a href="https://twitter.com/MarcEsteveG" target="\_blank" class="rk-social-btn ">

            <svg>

              <use xlink:href="assets/img/symbols.svg#icon-twitter"></use>

            </svg>

        </a>

        <a href="https://www.linkedin.com/in/marcestevegarcia/" target="\_blank" class="">

          <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"><path d="M12 0c-6.627 0-12 5.373-12 12s5.373 12 12 12 12-5.373 12-12-5.373-12-12-12zm-2 16h-2v-6h2v6zm-1-6.891c-.607 0-1.1-.496-1.1-1.109 0-.612.492-1.109 1.1-1.109s1.1.497 1.1 1.109c0 .613-.493 1.109-1.1 1.109zm8 6.891h-1.998v-2.861c0-1.881-2.002-1.722-2.002 0v2.861h-2v-6h2v1.093c.872-1.616 4-1.736 4 1.548v3.359z"/></svg>

        </a>

        </div>

      <div class="ae-grid\_\_item item-lg-4 au-xs-ta-center au-lg-ta-right">

        <p class="rk-footer\_\_text rk-footer\_\_contact "> <span class="ae-u-bold">Email: </span><span class="ae-u-bolder"> <a href="#0" class="rk-dark-color ">marc@pr0j3ct.com </a></span></p>

      </div>

    </div>

</footer>

Finalment ens quedarà la pàgina de ITEM per fer ús del component creat. Llavors hem d’anar a **portfolio.component.html** però de moment fem un canvi només a un *item*, no a tots perquè els recorrerem tots sense anar un a un.

<a routerLink="/item" class="rk-item ae-masonry\_\_item">

    <img src="assets/img/project-1.jpg" alt="">

    <div class="item-meta">

       <h2>Essential Stationery</h2>

       <p>Branding</p>

    </div>

</a>

Per tal de que a Github Pages no ens donin error les direccions per exemple “home” o “about” com si fossin en comptes de pàgines concretes dels nostres components, als servidors podrien malinterpretar que son directoris o carpetes /home/ o /about/, farem ús del hash # per tal de no tenir aquests errors, tot i que tenint permisos del típic arxiu .htaccess dels nostres servidors web podríem canviar-ho que les nostres rutes pesessin per la principal index.html. Llavors al app-routing.module.ts als imports afegirem un true a useHash:

@NgModule ({

    imports: [

        RouterModule.forRoot( app\_routes, {useHash: true})

    ],

    exports: [

        RouterModule

    ]

})

**Serveis, constants i HTTP**

1. Crear un JSON amb informació
2. Servei – Informació de la pàgina
3. Crear interfase per controlar la informació de la pàgina
4. Utilitzar el servei per reemplaçar informació a la pàgina web
5. Agregar animació a la nostra web app angular

**Crear un JSON amb informació**

Per a consultar la informació més rellevant d’una pàgina en un arxiu molt senzill i estructurat per un fàcil accés fem servir un .JSON.

JSON en assets/data/data-pagina.json

{

    "titulo": "Web App Angular",

    "email": "marc@pr0j3ct.com",

    "nombre\_corto": "Marc",

    "pagina\_autor": "https://pr0j3ct.com",

    "twitter": "https://twitter.com/marcesteveg",

    "linkedin": "https://www.linkedin.com/in/marcestevegarcia/",

    "equipo\_trabajo": []

}

Revisar a “JSON parser online”

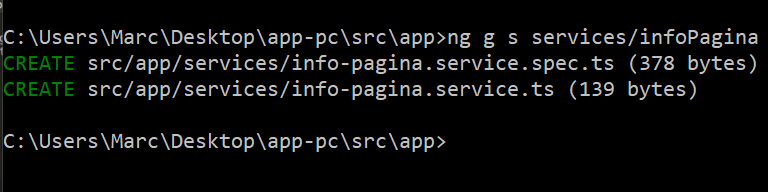
**Servei – Informació de la pàgina**

Per a poder llegir el JSON anterior a totes les pàgines, hi ha un lloc especial a Angular anomenat SERVEI, que es pot utilitzar per compartir informació global al llarg de l’aplicació o per gestionar certa lògica a algun lloc específic d’alguna pàgina.

Per tant, crearem un servei que llegeixi el JSON i poder compartir per tota l’app i totes les pàgines i ho llegirà un cop a cada carrega de la web.

*PAS 1*

Creem el service: **ng g s services/infoPagina**



*PAS 2*

Revisem al ***app.module.ts*** i a aquest arxiu veiem que no s’ha actualitzat cap servei ni cap import ni res, i normalment trobarem els serveis a “providers”. Això vol dir que ja no es necessari, podem **injectar** un servei des del servei (***info-pagina.service.ts***) :

@Injectable({

  providedIn: 'root'

})

*PAS 3*

Anem a provar a ***info-pagina.service.ts*** com podem injectar dependències des d’un servei, per tant, afegim un constructor al class InfoPaginaService i de moment només un console.log, però tot i que sigui injectable a través del @Injectable necessitem realitzar la injecció, revisem codi que encara no mostra el console.log

export class InfoPaginaService {

  constructor() {

     console.log("Servicio de infoPagina listo")

   }

}

*PAS 4*

Realitzem la injecció de dependència a ***app.component.ts*** i revisem que només amb això si que carrega cada full refresh un sol cop. Si revisem la consola hauria de funcionar i de fet es el lloc ideal per a poder rebre per exemple un JSON.

export class AppComponent {

  constructor (public infoPaginaService:InfoPaginaService){

     //Inyección de dependencia del servicio

  }

}

*PAS 5*

Lectura del JSON necessitem per mètode get() podrem fer peticions a servidors REST, llavors hem de realitzar la importació de **HttpClientModule** a ***app.module.ts*** :

import { HttpClientModule } from '@angular/common/http';

I al imports

imports: [

    BrowserModule,

    AppRoutingModule,

    HttpClientModule ],

*PAS 6*

Per poder llegir el JSON llegirem el path 'assets/data/data-pagina.json'

Aqui crearem les peticions get per http amb el subscribe que serà el “Observable”

SERVEI : info-pagina.service.ts

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { InfoPagina } from '../interfaces/info-pagina.interface';

@Injectable({

  providedIn: 'root'

})

export class InfoPaginaService {

  info: InfoPagina = {}; // Si no tenim la interfase InfoPagina seria de tipus any

  cargada = false; // No cal dir boolean

  constructor( private http: HttpClient ) {

    // console.log('Servicio de infoPagina listo');

    // Leer el archivo JSON

    this.http.get('assets/data/data-pagina.json')

        .subscribe( (resp: InfoPagina) => {

          this.cargada = true;

          this.info = resp; // provar resp. I veurem les propietats JSON

          console.log(resp);

        });

  }

}

**Crear interfase per controlar la informació de la pàgina**

INTERFASE : info-pagina.interface.ts (a la carpeta /app/interfases/) per donar **tipus**

Aprofitem l’extensió **JSON to TS** per convertir JSON a aquest interfase amb Ctrl +Shift + Alt + V

Amb l’interrogant fem opcionals totes les variables i tipus (botó central ratolí i Ctrl + ->)

export interface InfoPagina {

  titulo?: string;

  email?: string;

  nombre\_corto?: string;

  pagina\_autor?: string;

  facebook?: string;

  twitter?: string;

  instagram?: string;

  tublr?: string;

  equipo\_trabajo?: any[];

}

**Utilitzar el servei per reemplaçar informació a la pàgina web**

Afegim per exemple a **header.component.ts** el servei i que puguem entendre el JSON a través del servei InfoPaginaService, ara tindrem la injecció de dependència finalitzada.

export class HeaderComponent {

  constructor( public \_servicio:InfoPaginaService) {}

}

Recordem la importació correcta del servei :

import { InfoPaginaService } from 'src/app/services/info-pagina.service';

Llavors al **header.component.html**  podrem utilitzar per exemple al h1 agafar el títol:

<h1 class="rk-logo">

<a routerLink="inici">{{\_servicio.info.titulo}}</a>

</h1>

Revisar la càrrega correcte de les dades del JSON, fer modificacions i veure que canvia correctament, però només al primer “*full refresh”*.

Fem el mateix a footer amb les dades necessàries com correu per exemple o títol de la pàgina, etc **footer.component.ts**

export class FooterComponent {

  constructor( public \_servicio:InfoPaginaService) {}

}

Exemple de **l’enllaç** a la icona de Twitter a **footer.component.html**

<a [href]="\_servicio.info.twitter" target="\_blank" class="rk-social-btn ">

          <svg>

            <use xlink:href="assets/img/symbols.svg#icon-twitter"></use>

          </svg>

        </a>

I exemple del **correu** també al footer

<p class="rk-footer\_\_text rk-footer\_\_contact ">

   <span class="ae-u-bold">Email: </span>

   <span class="ae-u-bolder">

    <a href="#0" class="rk-dark-color">{{\_servicio.info.email}}</a>

   </span>

 </p>

**Agregar animació a la nostra web app angular**

Afegim al class=”animate\_\_animated animate\_\_fadeIn” on vulguem l’animació, prèviament al nostre **index.html** posarem l’enllaç del css:

<link

    rel="stylesheet"

    href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"

 />

<https://animate.style/> En el nostre cas un fadeIn

**portfolio.component.html**

<a routerLink="/item" class="animate\_\_animated animate\_\_fadeIn rk-item ae-masonry\_\_item">

          <img src="assets/img/project-1.jpg" alt="">

          <div class="item-meta">

            <h2>Essential Stationery</h2>

            <p>Branding</p>

          </div>

      </a>

**about.component.html**

<div class="ae-grid au-xs-ta-center au-mb-4">

    <div class="animate\_\_animated animate\_\_fadeIn ae-grid\_\_item item-lg-4 ae-kappa au-mb-3">

      <img src="assets/img/team-1.jpg" alt="" class="au-mb-3">

      <h5 class="ae-u-bolder au-mt-2">Robert T. Williams</h5>

      <p class="ae-u-bolder au-mb-3">Graphic Designer</p>

      <p class="au-lg-ta-left au-mb-3 au-pl-4 au-pr-4">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor.</p>

      <a href="#0" class="ae-u-bolder au-underline">@robertwilliams</a>

    </div>

    <div class="animate\_\_animated animate\_\_fadeIn ae-grid\_\_item item-lg-4 ae-kappa au-mb-3">

      <img src="assets/img/team-2.jpg" alt="" class="au-mb-3">

      <h5 class="ae-u-bolder au-mt-2">Angelina B. Widow</h5>

      <p class="ae-u-bolder au-mb-3">Founder / CEO</p>

      <p class="au-lg-ta-left au-mb-3 au-pl-4 au-pr-4">Ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit.</p>

      <a href="#0" class="ae-u-bolder au-underline">@angelinawidow</a>

    </div>

    <div class="animate\_\_animated animate\_\_fadeIn ae-grid\_\_item item-lg-4 ae-kappa au-mb-3">

      <img src="assets/img/team-3.jpg" alt="" class="au-mb-3">

      <h5 class="ae-u-bolder au-mt-2">Patrick Anderson</h5>

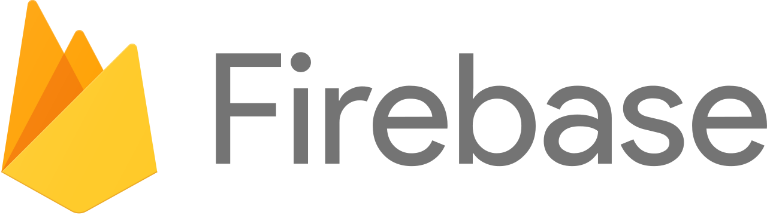
      <p class="ae-u-bolder au-mb-3">Art Director</p>

      <p class="au-lg-ta-left au-mb-3 au-pl-4 au-pr-4">Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

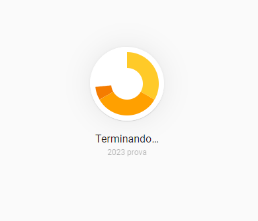
      <a href="#0" class="ae-u-bolder au-underline">@patrickanderson</a>

    </div>

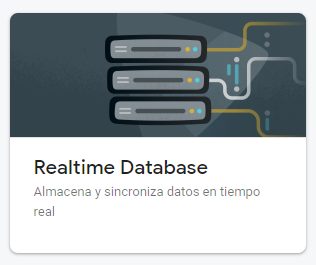
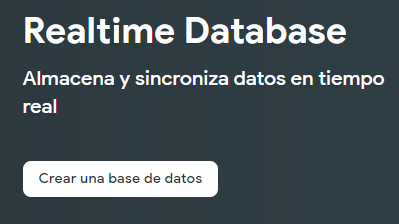
</div>

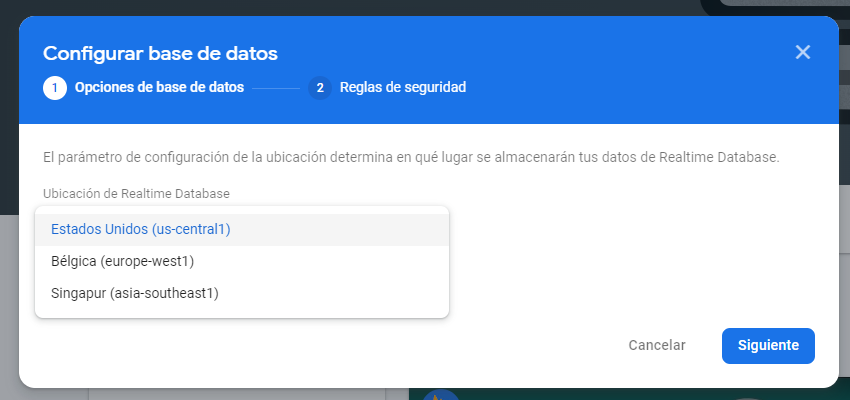
Creant projecte Firebase (Realtime Database + Storage)

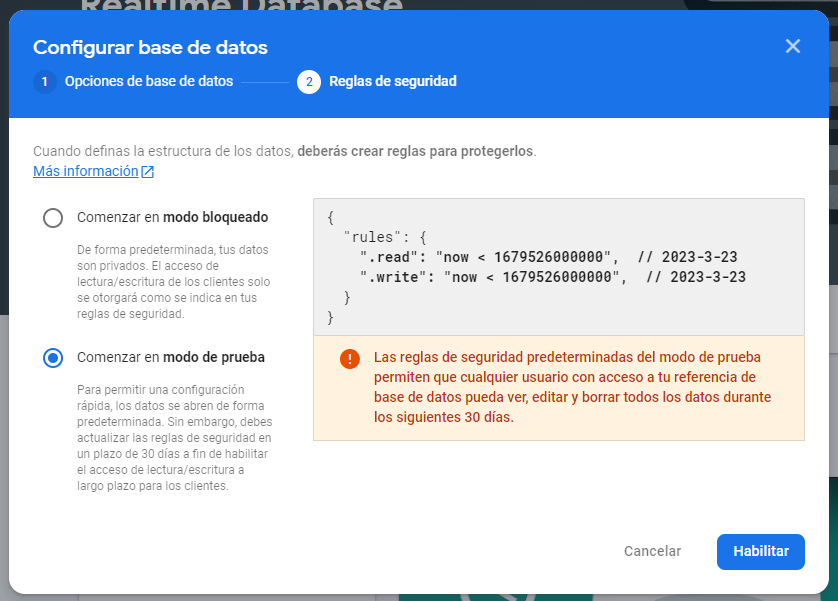
Iniciem sessió a Google, si tenim gmail més ràpid, sinó cal registrar-se i entrem a firebase.google.com (O cerca al navegador “firebase Google”

Entrem a la nostra “CONSOLA” per obrir un nou projecte, posem nom i seguim:

Per tal de poder afegir JSON al nostre contingut necessitarem obrir **Realtime Database + Storage**.







Regles:

{

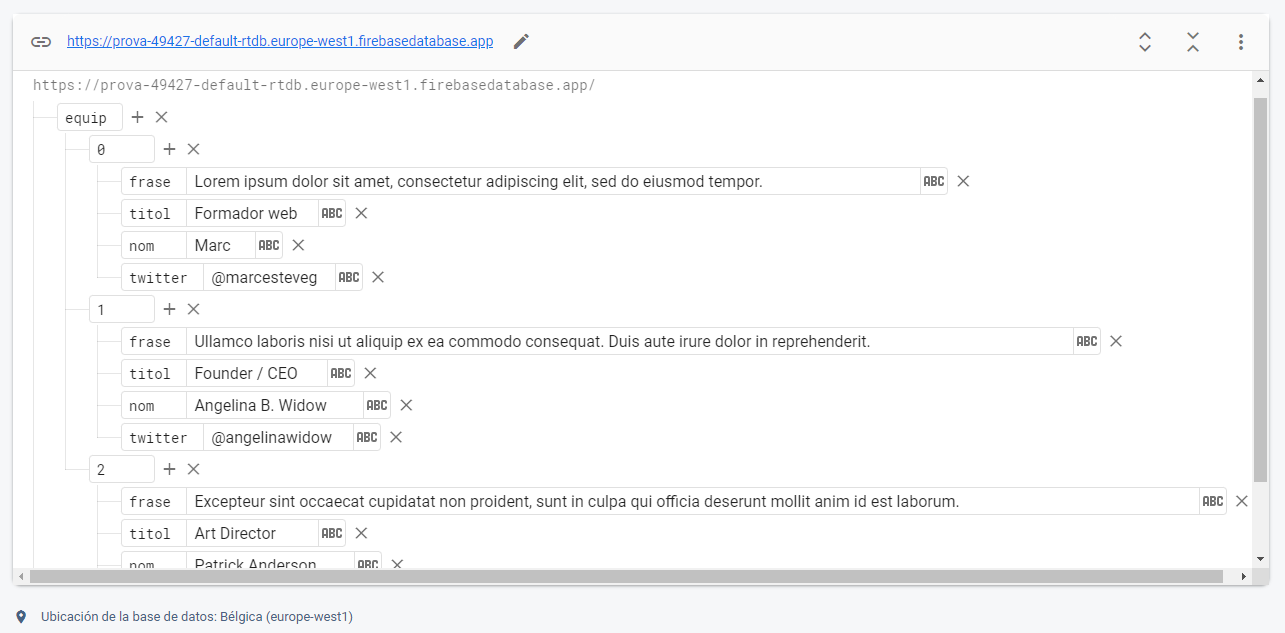
  "rules": {

    ".read": "now < 1679526000000",  // 2023-3-23

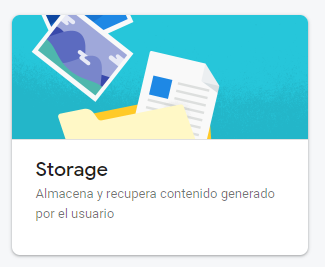
    ".write": false //Només nosaltres escrivim

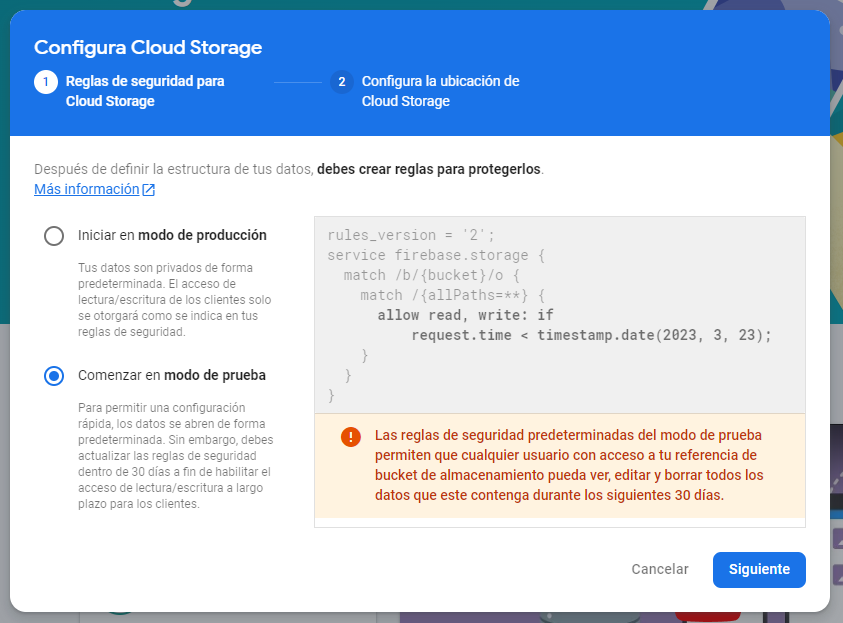
  }

}



Donem a “Agregar” i click a **equip** I ens donarà un enllaç que afegirem la terminació de l’extensió .json:

<https://prova-49427-default-rtdb.europe-west1.firebasedatabase.app/equip.json>

Ara anem a STORAGE per agefir les imatges

Reglas “rules”:

rules\_version = '2';

service firebase.storage {

  match /b/{bucket}/o {

    match /{allPaths=\*\*} {

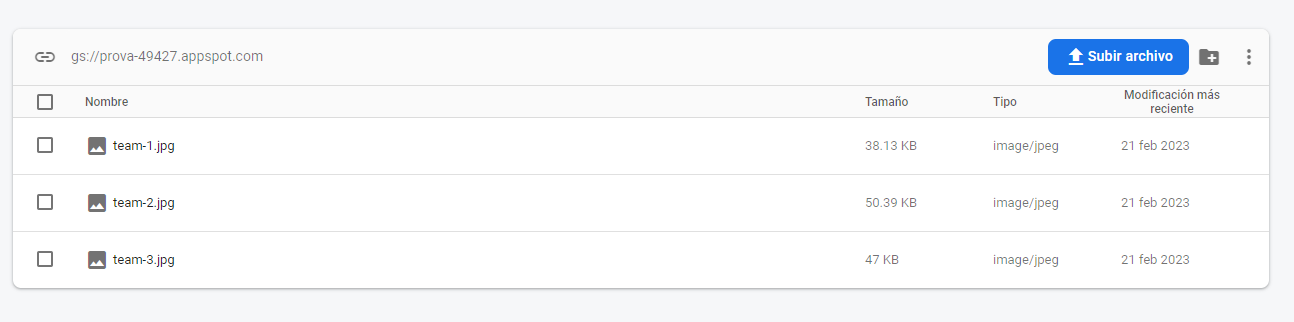
      allow read: if true;

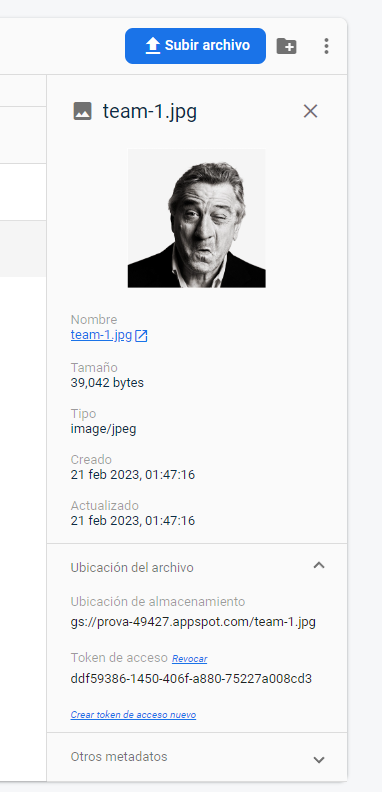
    }

  }

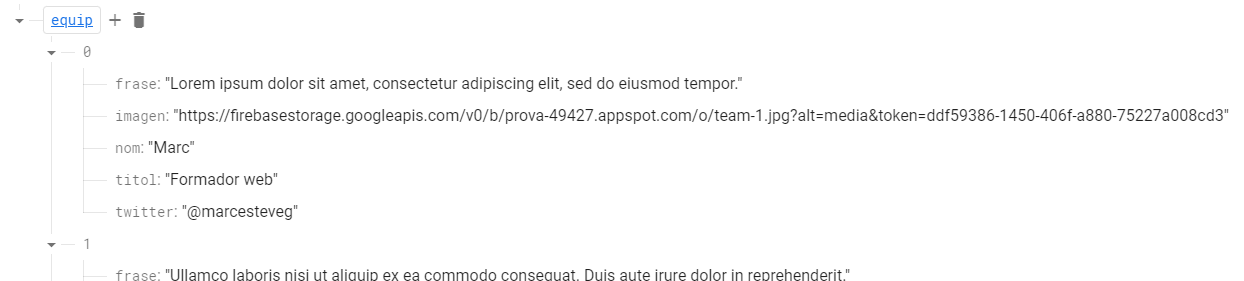
}

**eurowest3** per exemple





Ejemplo de imagen: <https://firebasestorage.googleapis.com/v0/b/prova-49427.appspot.com/o/team-1.jpg?alt=media&token=ddf59386-1450-406f-a880-75227a008cd3>



A RealTime Database podem fer un backup donant a “export” i tenint una copia de seguretat

Revisar en JSON Formatter extensió Chrome o Parser online

Carregar la informació de Firebase

Crear un mètode a **info-pagina.service.ts** que es diu cargarEquipo()

Llavors a l’arxiu info-pagina.service.ts canviarem:

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { InfoPagina } from '../interfaces/info-pagina.interface';

@Injectable({

  providedIn: 'root'

})

export class InfoPaginaService {

  info: InfoPagina = {};

  cargada =false;

  equipo:any = []; //Donde guardamos el equipo

  constructor(private http:HttpClient) {

    this.cargarInfo();

    this.cargarEquipo();

   }

   private cargarInfo() {

      this.http.get('assets/data/data-pagina.json')

      .subscribe( (resp: InfoPagina) => {

        this.cargada = true;

        this.info = resp;

      });

   }

   private cargarEquipo() { //Donde revisamos la info de Firebase

    this.http.get('https://prova-49427-default-rtdb.europe-west1.firebasedatabase.app/equip.json')

    .subscribe( resp  => {

      this.equipo = resp;

      console.log(resp);

    });

   }

}

Mostrant la informació de Firebase al HTML

Dins l’arxiu de **about.component.ts** afegim:

export class AboutComponent {

  constructor (public infoService:InfoPaginaService){}

}

Automàticament importa:

import { InfoPaginaService } from '../../services/info-pagina.service';

Ara no caldrà fer de un en un els elements, sino recorre amb \*ngFor totes les dades de l’equip que tenim a Firebase a **about.component.html**

\*ngFor="let persona of infoService.equipo"

I si repasem totes les dades per repetir les dades de cada membre de l’equip des de la info de Firebase podria quedar tal que així el nostre ABOUT:

<div class="ae-grid au-xs-ta-center au-mb-4">

    <div \*ngFor="let persona of infoService.equipo" class="animate\_\_animated animate\_\_fadeIn ae-grid\_\_item item-lg-4 ae-kappa au-mb-3">

      <img [src]="persona.imagen" alt="" class="au-mb-3">

      <h5 class="ae-u-bolder au-mt-2">{{ persona.nom }}</h5>

      <p class="ae-u-bolder au-mb-3">{{ persona.titol }}</p>

      <p class="au-lg-ta-left au-mb-3 au-pl-4 au-pr-4">{{ persona.frase }}</p>

      <a href="https://twitter/{{persona.twitter}}" class="ae-u-bolder au-underline">{{ persona.twitter }}</a>

    </div>

</div>

Creant Webapp Angular a productiu i testeig

Primer haurem de preparar la nostra app a “pre-producció”.

Aturem la app!!!

Comanda **ng build** per crear la webapp a productiu

També a la carpeta /dist/portfolio/ npm i angular-http-server

I finalment “angular-http-server” per fer córrer la app en productiu a <http://localhost:8080>

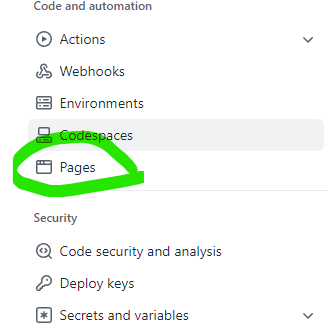
Desplegant la nostra WebApp a Github Pages

Ara anirem a github.com per crear un repositori i preparem els arxius a una carpeta anomenada /docs/ on abans era /dist/portfolio/

A index.html treiem o comentem

<base href="/">

Afegirem els nostres arxius amb COMMIT/PUSH al nostre repositori creat per aquesta app

Busquem a Settings > Pages i al **Branch** afegim la carpeta /docs on tenim la app a productiu i si ho hem fet bé en generarà un enllaç en el que tindrem correns la nostra app a un subdomini de Github similar a aquesta: <https://github.com/MarcEsteve/portfoli-angular> (repositori preparat)

[https://marcesteve.github.io/portfoli-angular/#/home](https://marcesteve.github.io/portfoli-angular/%23/home)