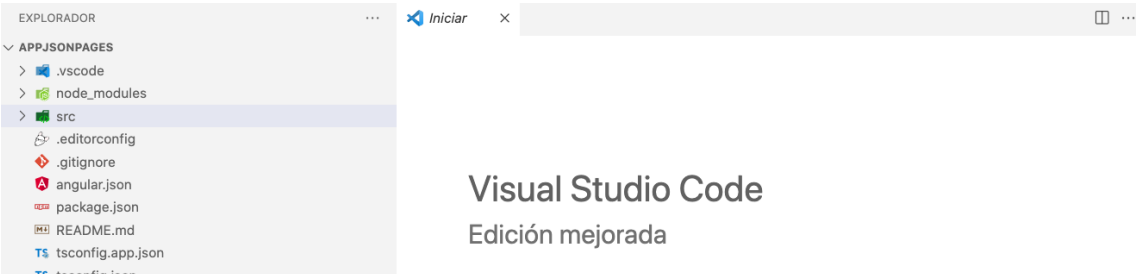
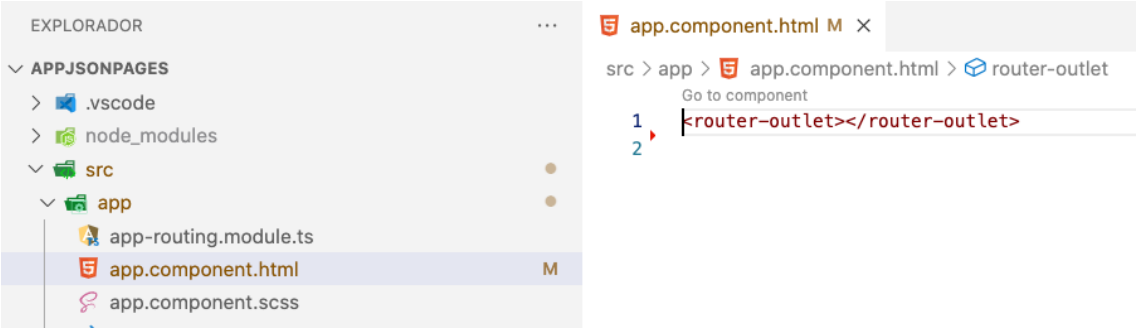
1. Proyecto con lectura estática de datos:

```
jsersan@iMac-de-Jose angular % ng new appJSONpages
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS   [ https://sass-lang.com/documentation/syntax#scss                ]
CREATE appJSONpages/README.md (1066 bytes)
CREATE appJSONpages/.editorconfig (274 bytes)
CREATE appJSONpages/.gitignore (548 bytes)
CREATE appJSONpages/angular.json (2905 bytes)
CREATE appJSONpages/package.json (1044 bytes)
CREATE appJSONpages/tsconfig.json (901 bytes)
CREATE appJSONpages/tsconfig.app.json (263 bytes)
```

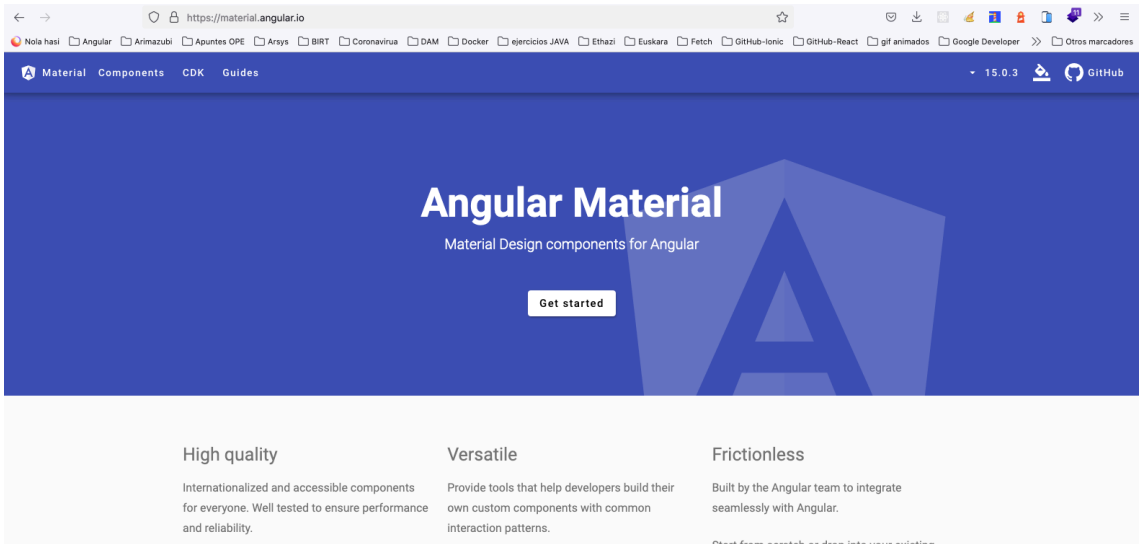
Lo abrimos en vs code:



Dejamos app.component.html así:



Instalamos Material:



Lo añadimos a nuestro proyecto. En la documentación de material:

Install Angular Material

Use the Angular CLI's installation schematic to set up your Angular Material project by running the following command:

```
ng add @angular/material
```

En la consola de vs code:

```
jsersan@iMac-de-Jose appJSONpages % ng add @angular/material
i Using package manager: npm
✔ Found compatible package version: @angular/material@15.0.3.
✔ Package information loaded.

The package @angular/material@15.0.3 will be installed and executed.
Would you like to proceed? (Y/n)
```

Procedemos y seleccionamos la primera opción y le decimos que Sí a la configuración de estilos tipográficos:

```
jsersan@iMac-de-Jose appJSONpages % ng add @angular/material
Skipping installation: Package already installed
? Choose a prebuilt theme name, or "custom" for a custom theme: Indigo/Pink        [ Preview:
https://material.angular.io?theme=indigo-pink ]
? Set up global Angular Material typography styles? Yes
? Include the Angular animations module? (Use arrow keys)
❯ Include and enable animations
  Include, but disable animations
  Do not include
```

Cuando termine:

```
? Set up global Angular Material typography styles? Yes
? Include the Angular animations module? Include and enable animations
UPDATE package.json (1110 bytes)
✔ Packages installed successfully.
UPDATE src/app/app.module.ts (502 bytes)
UPDATE angular.json (3039 bytes)
UPDATE src/index.html (580 bytes)
UPDATE src/styles.scss (181 bytes)
jsersan@iMac-de-Jose appJSONpages %
```

En index.html:

```html
<head>
  <meta charset="utf-8">
  <title>AppJSONpages</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>
<body class="mat-typography">
  <app-root></app-root>
</body>
```

Y en app.module.ts:

```
src > app > app.module.ts > ...
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

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

Creamos el componente databaseTabla:

```
jsersan@iMac-de-Jose appJSONpages % ng g c components/databaseTabla --skip-tests --inline-style
CREATE src/app/components/database-tabla/database-tabla.component.html (29 bytes)
CREATE src/app/components/database-tabla/database-tabla.component.ts (201 bytes)
UPDATE src/app/app.module.ts (625 bytes)
jsersan@iMac-de-Jose appJSONpages %
```

Aquí lo tenemos:

```
src > app > components > database-tabla > database-tabla.component.html > p
Go to component
<p>database-tabla works!</p>
```

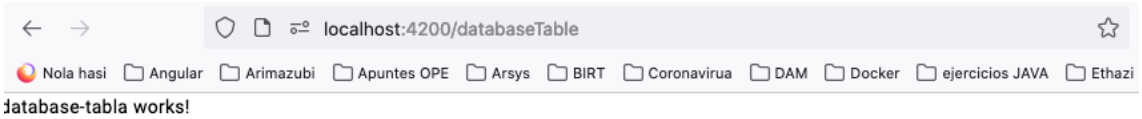
Vamos a definir las rutas:

```
src > app > app-routing.module.ts > AppRoutingModule
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { DatabaseTablaComponent } from './components/database-tabla/database-tabla.component';

const routes: Routes = [{
  path:'databaseTable',
  component: DatabaseTablaComponent
}];
```

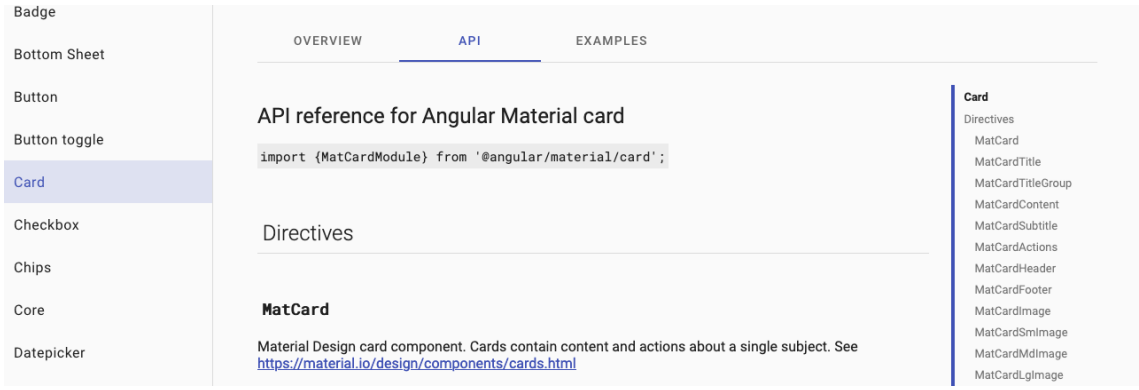
Levantamos el proyecto:

```
jsersan@iMac-de-Jose appJSONpages % npm run start

> app-jsonpages@0.0.0 start
> ng serve

✔ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   2.33 MB |
styles.css, styles.js | styles        | 328.37 kB |
polyfills.js          | polyfills     | 314.28 kB |
main.js               | main          |   8.22 kB |
runtime.js            | runtime       |   6.52 kB |

                      | Initial Total |   2.97 MB

Build at: 2022-12-26T17:14:50.189Z - Hash: d79b0f73994eaf68 - Time: 19394ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


✔ Compiled successfully.
```

Resultado:



Funciona! Como era de esperar. Nos vamos a este componente y copiamos desde material.angular.io el código para una tarjeta (materialCard):



En app.module.ts:

```
8
9   //Material
10  import {MatCardModule} from '@angular/material/card';
11
12  @NgModule({
13    declarations: [
14      AppComponent,
15      DatabaseTablaComponent
16    ],
17    imports: [
18      BrowserModule,
19      AppRoutingModule,
20      BrowserAnimationsModule,
21      MatCardModule
22    ],
```

Vamos a utilizarlo en el componente databaseTable:

Basic cards

HTML TS

```
<mat-card>
  <mat-card-content>Simple card</mat-card-content>
</mat-card>
```

Simple card

Así:

app-routing.module.ts M | app.module.ts M | database-tabla.component.html U

src > app > components > database-tabla > database-tabla.component.html > ...

Go to component

```
1 <mat-card>
2     <mat-card-content>database-tabla works"</mat-card-content>
3 </mat-card>
```

Vista previa:

localhost:4200/databaseTable

Nola hasi | Angular | Arimazubi | Apuntes OPE | Arsys | BIRT | Coronavirua | DAM | Otros marcadores

database-tabla works"

Nos copiamos de Bootstrap el css:

```
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="icon" type="image/x-icon" href="favicon.ico">
9   <link rel="preconnect" href="https://fonts.gstatic.com">
10  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap" rel="stylesheet
11  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
12  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="styleshee
13 </head>
```

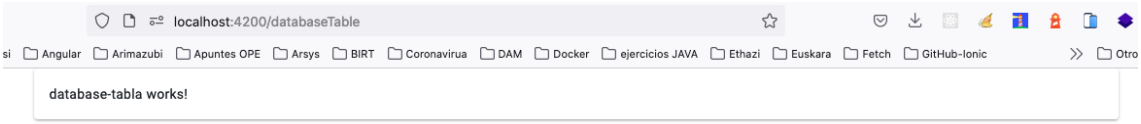
Ahora en nuestro componente utilizamos la clase de Bootstrap:

src > app > components > database-tabla > database-tabla.component.html > div.container > div.row

Go to component

```
1 <div class="container">
2     <div class="row">
3         <div class="col-12">
4             <mat-card>
5                 <mat-card-content>database-tabla works!</mat-card-content>
6             </mat-card>
7         </div>
8     </div>
9 </div>
```

El resultado:

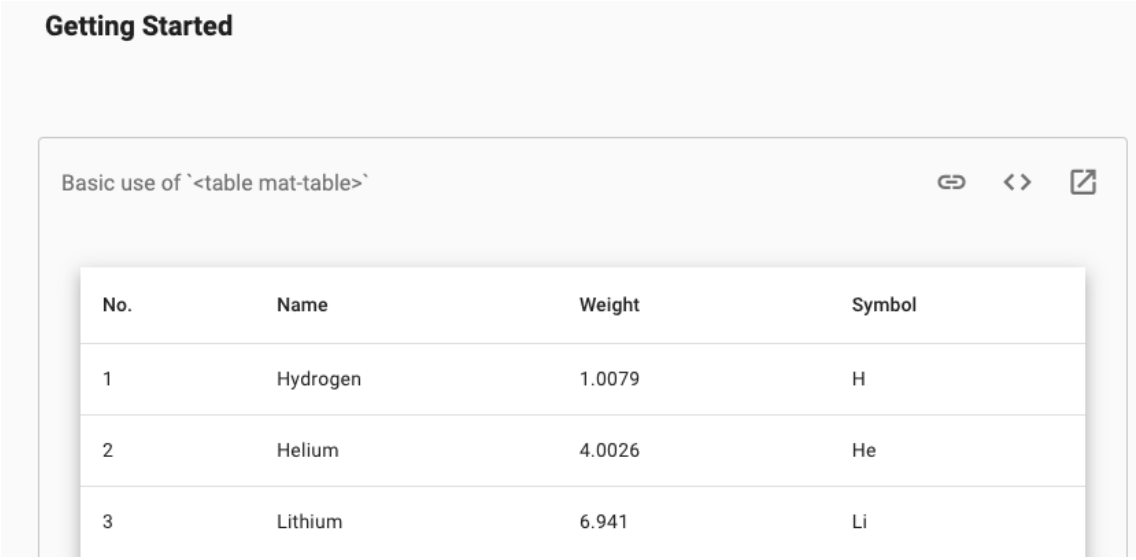


Ahora agregamos una tabla desde Material:

En app.module.ts:

```
9   //Material
10  import {MatCardModule} from '@angular/material/card';
11  import {MatTableModule} from '@angular/material/table';
12
13  @NgModule({
14    declarations: [
15      AppComponent,
16      DatabaseTablaComponent
17    ],
18    imports: [
19      BrowserModule,
20      AppRoutingModule,
21      BrowserAnimationsModule,
22      MatCardModule,
23      MatTableModule
24    ],
```

Vamos a copiar un ejemplo:



Le damos a <> para acceder al código:

HTML TS CSS

```
<table mat-table [dataSource]="dataSource" class="mat-elevation-z8">

  <!--- Note that these columns can be defined in any order.
        The actual rendered columns are set as a property on the row definition" -->

  <!-- Position Column -->
  <ng-container matColumnDef="position">
    <th mat-header-cell *matHeaderCellDef> No. </th>
    <td mat-cell *matCellDef="let element"> {{element.position}} </td>
  </ng-container>
```

Lo copiamos en:

database-tabla.component.html > div.container > div.row > div.col-12 > mat-card > mat-card

Go to component

```
1  <div class="container">
2      <div class="row">
3          <div class="col-12">
4              <mat-card>
5                  <mat-card-content>
6                      <table mat-table [dataSource]="dataSource" class="mat-elevation-z8">
```

Nos da este error porque falta el código de typescript:

```
export interface PeriodicElement {
  name: string;
  position: number;
  weight: number;
  symbol: string;
}

const ELEMENT_DATA: PeriodicElement[] = [
  {position: 1, name: 'Hydrogen', weight: 1.0079, symbol: 'H'},
  {position: 2, name: 'Helium', weight: 4.0026, symbol: 'He'},
  {position: 3, name: 'Lithium', weight: 6.941, symbol: 'Li'},
  {position: 4, name: 'Beryllium', weight: 9.0122, symbol: 'Be'},
  {position: 5, name: 'Boron', weight: 10.811, symbol: 'B'},
  {position: 6, name: 'Carbon', weight: 12.0107, symbol: 'C'},
  {position: 7, name: 'Nitrogen', weight: 14.0067, symbol: 'N'},
  {position: 8, name: 'Oxygen', weight: 15.9994, symbol: 'O'},
```

Lo pegamos en el componente:

src > app > components > database-tabla > database-tabla.component.ts > ...

```
1   import { Component } from '@angular/core';
2
3   export interface PeriodicElement {
4     name: string;
5     position: number;
6     weight: number;
7     symbol: string;
8   }
9
10  const ELEMENT_DATA: PeriodicElement[] = [
11    {position: 1, name: 'Hydrogen', weight: 1.0079, symbol: 'H'},
```

Nos va a hacer falta también estas dos variables:

```
export class TableBasicExample {
  displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
  dataSource = ELEMENT_DATA;
}
```

Lo pegamos en el componente:

```
30  export class DatabaseTablaComponent {
31
32    displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
33    dataSource = ELEMENT_DATA;
34
35    constructor(){}
36
37  }
38
```

Fijamos el ancho de la tabla:

```
src > styles.scss > table
1  /* You can add global styles to this file, and also import other style files */
2
3  html, body { height: 100%; }
4  body { margin: 0; font-family: Roboto, "Helvetica Neue", sans-serif; }
5  table {width: 100%; }
```

Guardamos y tenemos:

localhost:4200/databaseTable

No.	Name	Weight	Symbol
1	Hydrogen	1.0079	H
2	Helium	4.0026	He
3	Lithium	6.941	Li
4	Beryllium	9.0122	Be
5	Boron	10.811	B
6	Carbon	12.0107	C
7	Nitrogen	14.0067	N
8	Oxygen	15.9994	O
9	Fluorine	18.9984	F
10	Neon	20.1797	Ne

A la primera!

Vamos ahora a los ejemplos de material y encontramos:

Table with pagination

No.	Name	Weight	Symbol
1	Hydrogen	1.0079	H
2	Helium	4.0026	He
3	Lithium	6.941	Li
4	Beryllium	9.0122	Be
5	Boron	10.811	B

Items per page: 5 1 – 5 of 20

Si vemos el código, aparece al final:

```
<mat-paginator [pageSizeOptions]="[5, 10, 20]"
               showFirstLastButtons
               aria-label="Select page of periodic elements">
</mat-paginator>
```

Lo copiamos y lo pegamos en:

```
34
35                  <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
36                  <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
37              </table>
38
39              <mat-paginator [pageSizeOptions]="[5, 10, 20]"
40                             showFirstLastButtons
41                             aria-label="Select page of periodic elements">
42              </mat-paginator>
```

Para quitar el error debemos importar:

OVERVIEW API EXAMPLES

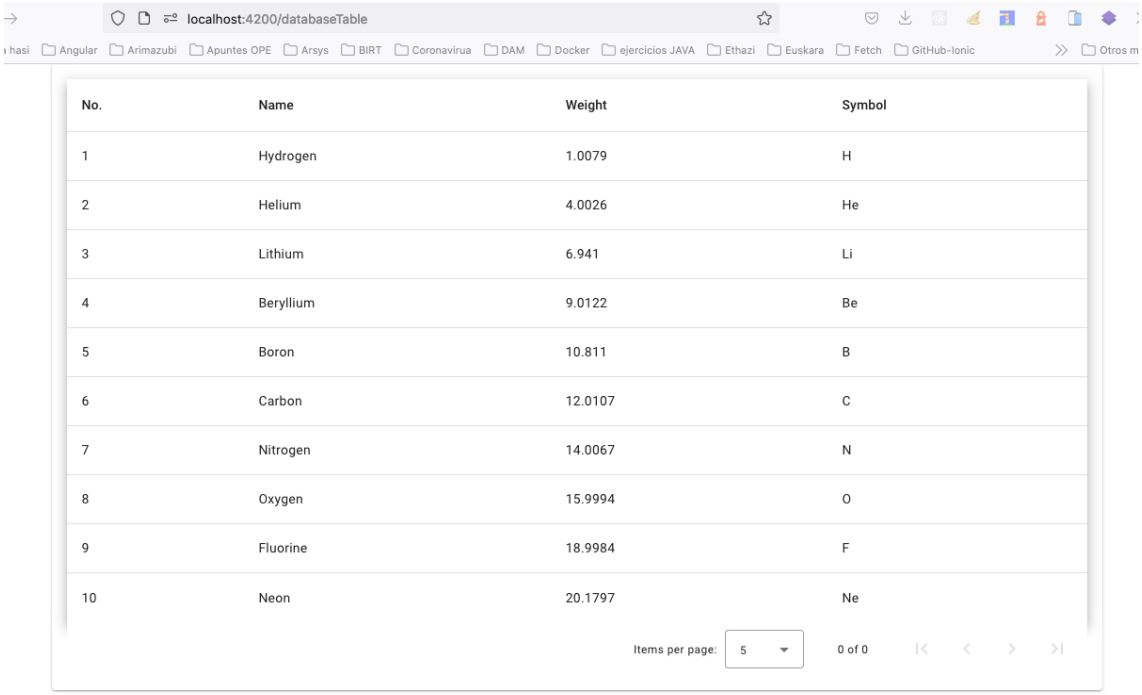
API reference for Angular Material paginator

```
import {MatPaginatorModule} from '@angular/material/paginator';
```

Así en app.module.ts:

```
9   //Material
10  import {MatCardModule} from '@angular/material/card';
11  import {MatTableModule} from '@angular/material/table';
12  import {MatPaginatorModule} from '@angular/material/paginator';
13
14  @NgModule({
15    declarations: [
16      AppComponent,
17      DatabaseTablaComponent
18    ],
19    imports: [
20      BrowserModule,
21      AppRoutingModule,
22      BrowserAnimationsModule,
23      MatCardModule,
24      MatTableModule,
25      MatPaginatorModule
26    ],
```

Si recargamos:



Está la interfaz pero todavía no funcionan los botones. Para ello en Table with pagination debemos declarar:

```
export class TablePaginationExample implements AfterViewInit {
  displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
  dataSource = new MatTableDataSource<PeriodicElement>(ELEMENT_DATA);

  @ViewChild(MatPaginator) paginator: MatPaginator;

  ngAfterViewInit() {
    this.dataSource.paginator = this.paginator;
  }
}
```

Copiamos este código y lo pegamos en el componente:

```
32  export class DatabaseTablaComponent {
33
34    displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
35    dataSource = ELEMENT_DATA;
36
37    @ViewChild(MatPaginator) paginator: MatPaginator;
38
39    ngAfterViewInit() {
40      this.dataSource.paginator = this.paginator;
41    }
```

Para quitar los errores importamos las librerías que faltan:

```
37  @ViewChild(MatPaginator) paginator: MatPaginator;
38
39  Corrección rápida...
40    Actualizar importación desde "@angular/core"
41    Agregar todas las importaciones que faltan
42    Agregar la declaración de función "ViewChild" que falta
43    Agregar todas las declara  Enter para aplicar, ⌘Enter para previsualizar
44
```

Después de agregar las librerías necesarias:

```
26  @Component({
27    selector: 'app-database-tabla',
28    templateUrl: './database-tabla.component.html',
29    styles: [
30    ]
31  })
32  export class DatabaseTablaComponent {
33
34    displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
35    dataSource = ELEMENT_DATA;
36
37    @ViewChild(MatPaginator) paginator!: MatPaginator;
38
39    ngAfterViewInit() {
40      this.dataSource.paginator = this.paginator;
41    }
42
43    constructor(){}
44
45  }
```

Para quitar este error, marcamos este componente como opcional:

```
34  displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
35  dataSource = ELEME  any
36
37  @ViewChild(MatPagi  La propiedad 'paginator' no existe en el tipo
38                      'PeriodicElement[]'. ts(2339)
39  ngAfterViewInit()   Ver el problema (⌥F8)   No hay correcciones rápidas disponibles
40    this.dataSource.paginator! = this.paginator;
41  }
```

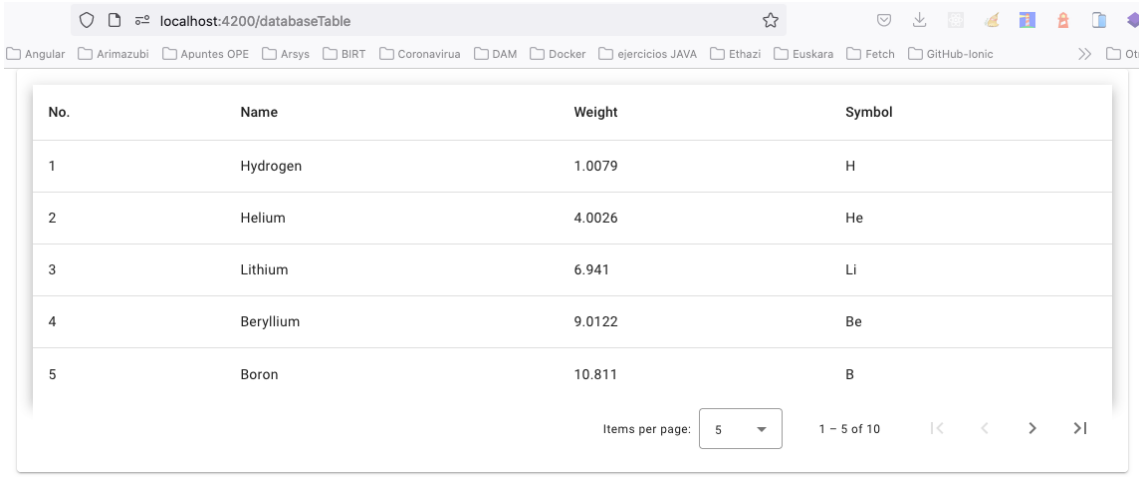
Para eso, siguiendo la documentación debemos definir el dataSource:

```
export class TablePaginationExample implements AfterViewInit {
  displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
  dataSource = new MatTableDataSource<PeriodicElement>(ELEMENT_DATA);
```

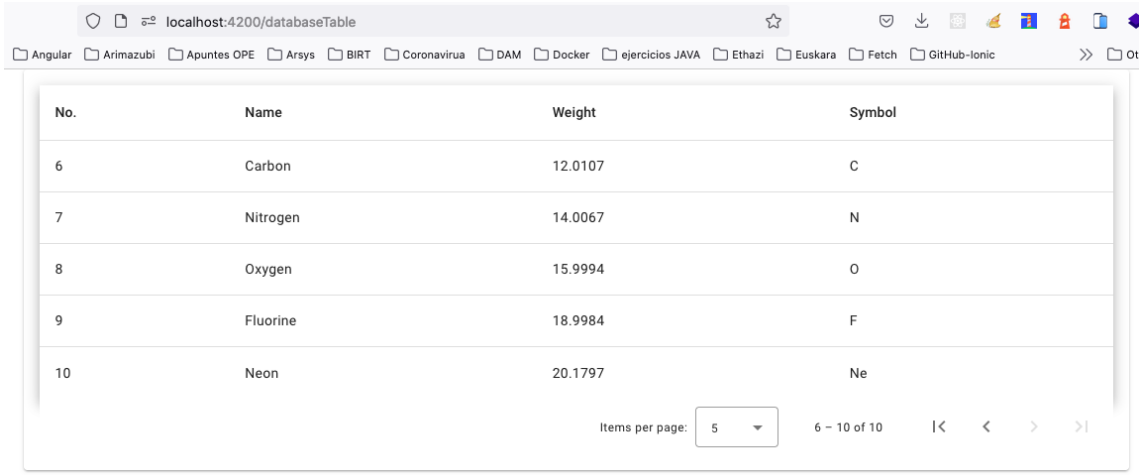
Copiamos y pegamos en el componente y tras quitar el error que nos sale:

```
32  export class DatabaseTablaComponent {
33
34    displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
35    dataSource = new MatTableDataSource<PeriodicElement>(ELEMENT_DATA);
36
37    @ViewChild(MatPaginator) paginator!: MatPaginator;
38
39    ngAfterViewInit() {
40      this.dataSource.paginator! = this.paginator;
41    }
```

Miramos ahora:



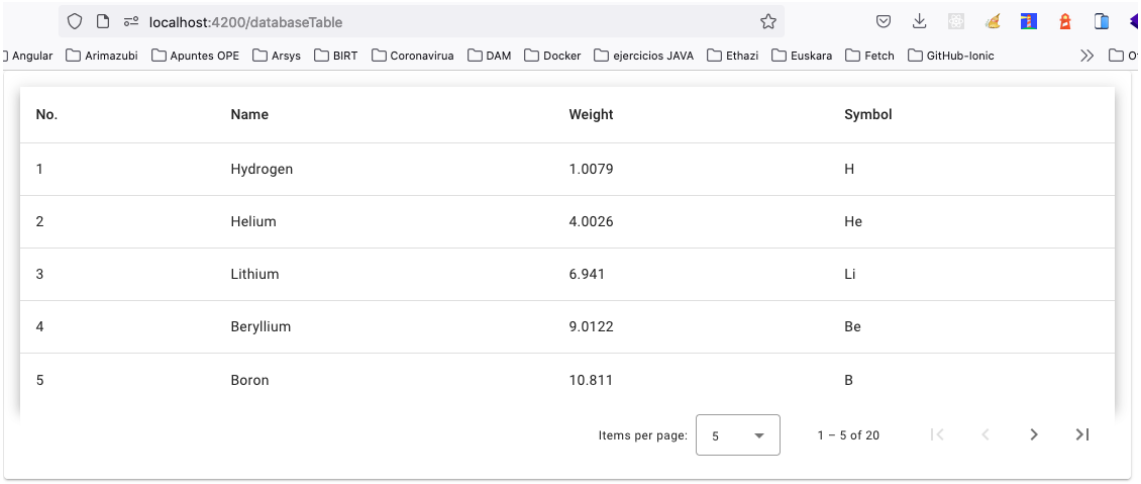
Los siguientes cinco:



Si ampliamos la data a 20 elementos:

```
12  const ELEMENT_DATA: PeriodicElement[] = [
13    {position: 1, name: 'Hydrogen', weight: 1.0079, symbol: 'H'},
14    {position: 2, name: 'Helium', weight: 4.0026, symbol: 'He'},
15    {position: 3, name: 'Lithium', weight: 6.941, symbol: 'Li'},
16    {position: 4, name: 'Beryllium', weight: 9.0122, symbol: 'Be'},
17    {position: 5, name: 'Boron', weight: 10.811, symbol: 'B'},
18    {position: 6, name: 'Carbon', weight: 12.0107, symbol: 'C'},
19    {position: 7, name: 'Nitrogen', weight: 14.0067, symbol: 'N'},
20    {position: 8, name: 'Oxygen', weight: 15.9994, symbol: 'O'},
21    {position: 9, name: 'Fluorine', weight: 18.9984, symbol: 'F'},
22    {position: 10, name: 'Neon', weight: 20.1797, symbol: 'Ne'},
23    {position: 11, name: 'Sodium', weight: 22.9897, symbol: 'Na'},
24    {position: 12, name: 'Magnesium', weight: 24.305, symbol: 'Mg'},
25    {position: 13, name: 'Aluminum', weight: 26.9815, symbol: 'Al'},
26    {position: 14, name: 'Silicon', weight: 28.0855, symbol: 'Si'},
27    {position: 15, name: 'Phosphorus', weight: 30.9738, symbol: 'P'},
28    {position: 16, name: 'Sulfur', weight: 32.065, symbol: 'S'},
29    {position: 17, name: 'Chlorine', weight: 35.453, symbol: 'Cl'},
30    {position: 18, name: 'Argon', weight: 39.948, symbol: 'Ar'},
31    {position: 19, name: 'Potassium', weight: 39.0983, symbol: 'K'},
32    {position: 20, name: 'Calcium', weight: 40.078, symbol: 'Ca'},
33  ];
```

Ahora:



2. Proyecto con lectura de datos desde una API:

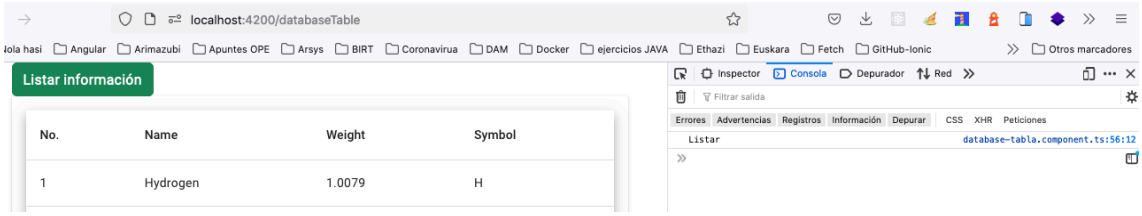
Vamos a conectar la app anterior a un servicio para importar los datos. En el template html añadimos una nueva línea.

```
Go to component
1  <div class="container">
2    <div class="row">
3      <div class="col-12">
4        <button class="btn btn-success" (click)="listar()">Listar Información
5        </button>
6      </div>
7    </div>
8  </div>
```

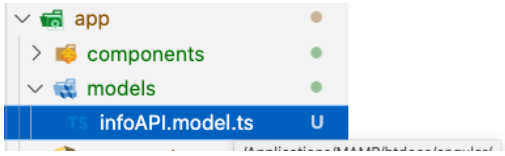
En el componente creamos este método:

```
53    constructor(){}
54
55    listar(){
56      console.log("Listar");
57    }
58
59  }
```

Probamos dándole al botón:



Ahora en nuestro proyecto vamos a crear una carpeta para nuestro modelo de datos:



El contenido es:

```
src > app > models > TS infoAPI.model.ts > infoAPI > Cors
1  export interface infoAPI{
2      API: string,
3      Description: string,
4      Link: string,
5      Category: string,
6      Cors: string
7  }
```

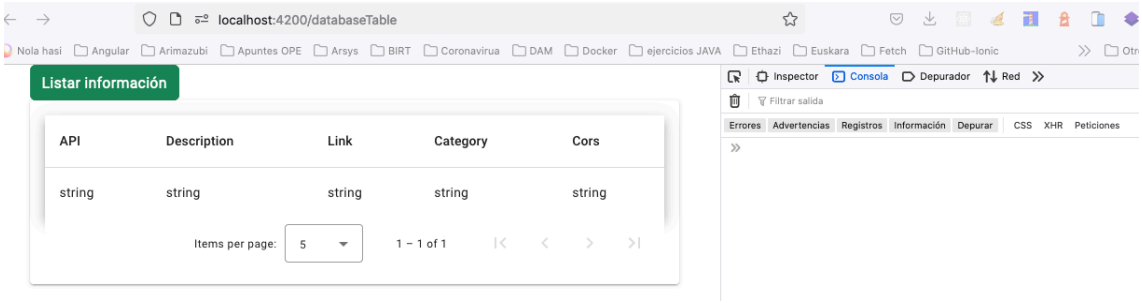
Así en el componente database-tabla.componente.ts:

```
4   import { infoAPI } from 'src/app/models/infoAPI.model';
5
6   const DATA: infoAPI[]= [
7     {   API: 'string',
8         Description: 'string',
9         Link: 'string',
10        Category: 'string',
11        Cors: 'string'
12    },
13  ];
14
15  @Component({
16    selector: 'app-database-tabla',
17    templateUrl: './database-tabla.component.html',
18    styles: [
19    ]
20  })
21  export class DatabaseTablaComponent {
22
23    displayedColumns: string[] = ['API', 'Description', 'Link', 'Category','Cors'];
```

En el template:

```
<table mat-table [dataSource]="dataSource" class="mat-elevation-z8">

    <!--- Note that these columns can be defined in any order.
          The actual rendered columns are set as a property on the row definition" -->

    <!-- API Column -->
    <ng-container matColumnDef="API">
      <th mat-header-cell *matHeaderCellDef> API </th>
      <td mat-cell *matCellDef="let element"> {{element.API}} </td>
    </ng-container>

    <!-- Description Column -->
    <ng-container matColumnDef="Description">
      <th mat-header-cell *matHeaderCellDef> Description </th>
      <td mat-cell *matCellDef="let element"> {{element.Description}} </td>
    </ng-container>

    <!--Link Column -->
    <ng-container matColumnDef="Link">
      <th mat-header-cell *matHeaderCellDef> Link </th>
      <td mat-cell *matCellDef="let element"> {{element.Link}} </td>
    </ng-container>

    <!-- Category Column -->
    <ng-container matColumnDef="Category">
      <th mat-header-cell *matHeaderCellDef>Category</th>
      <td mat-cell *matCellDef="let element"> {{element.Category}} </td>
    </ng-container>

    <!-- Cors Column -->
    <ng-container matColumnDef="Cors">
        <th mat-header-cell *matHeaderCellDef>Cors</th>
        <td mat-cell *matCellDef="let element"> {{element.Cors}} </td>
    </ng-container>

    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
</table>

<mat-paginator [pageSizeOptions]="[5, 10, 20]"
        showFirstLastButtons></mat-paginator>
```

Sólo se verá uno que se puso manualmente a propósito:



Esta información la ha cargado de:

```
6  const DATA: infoAPI[]= [
7    {   API: 'string',
8        Description: 'string',
9        Link: 'string',
10       Category: 'string',
11       Cors: 'string'
12   },
13 ];
```

Si queremos que el botón de verdad cargue información, eliminamos este contenido y cogemos los datos de la API. El DataSource va a estar vacío la primera vez. Lo dejamos así:

```
5
6  @Component({
7    selector: 'app-database-tabla',
8    templateUrl: './database-tabla.component.html',
9    styles: [
10   ]
11 })
12 export class DatabaseTablaComponent {
13
14   displayedColumns: string[] = ['API', 'Description', 'Link', 'Category','Cors'];
15   dataSource = new MatTableDataSource<infoAPI>([]);
16
17   @ViewChild(MatPaginator) paginator!: MatPaginator;
```

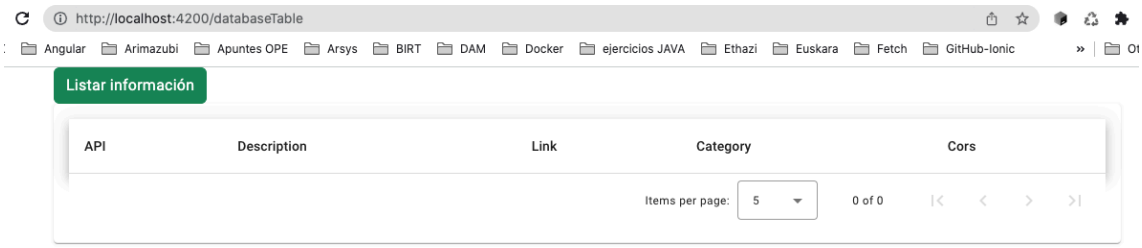
En la función listar:

```
29   listar() {
30     this.dataSource.data = [
31       {
32         API: 'string',
33         Description: 'string',
34         Link: 'string',
35         Category: 'string',
36         Cors: 'string',
37       },
38     ];
39   }
```

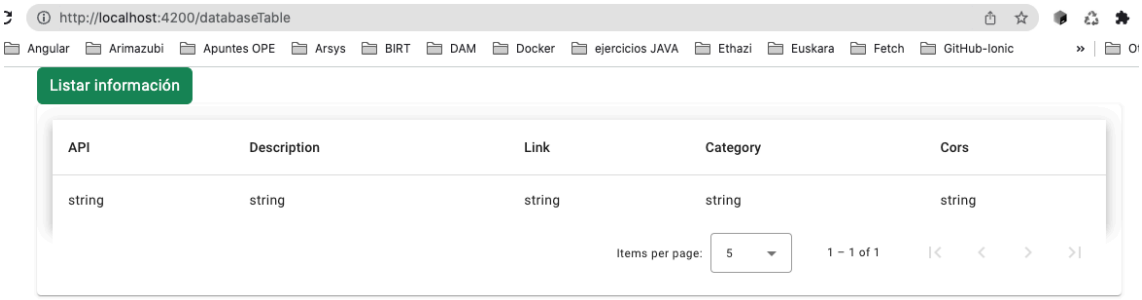
El componente queda:

```
13  export class DatabaseTablaComponent {
14    displayedColumns: string[] = [
15      'API',
16      'Description',
17      'Link',
18      'Category',
19      'Cors',
20    ];
21    dataSource = new MatTableDataSource<infoAPI>([]);
22
23    @ViewChild(MatPaginator) paginator!: MatPaginator;
24
25    ngAfterViewInit() {
26      this.dataSource.paginator! = this.paginator;
27    }
28
29    constructor() {}
30
31    listar() {
32      this.dataSource.data = [
33        {
34          API: 'string',
35          Description: 'string',
36          Link: 'string',
37          Category: 'string',
38          Cors: 'string',
39        },
40      ];
41    }
42  }
```

Probamos y vemos si carga bien la información:



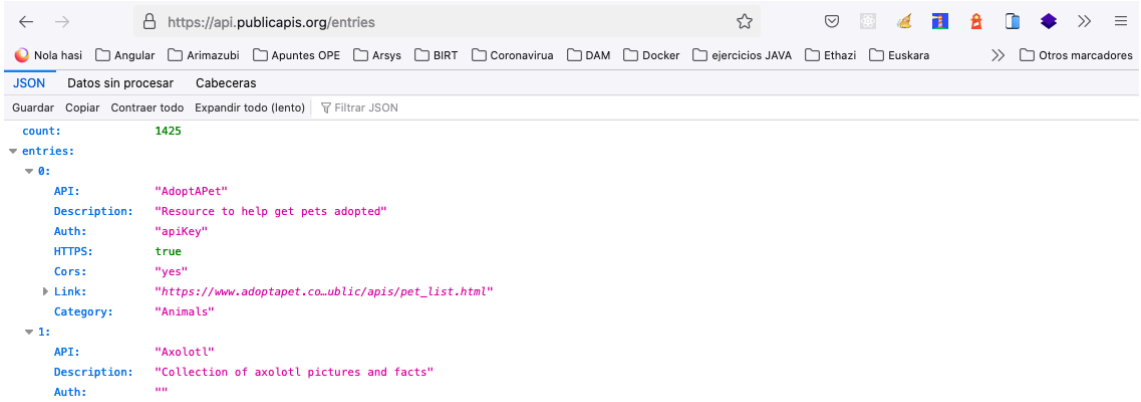
Le damos por primera vez al botón:



Lo óptimo es hacer esto mediante un servicio. Para ello nos creamos una carpeta services/dataTableService:

```
jsersan@iMac-de-Jose appJSONpages % ng g s services/dataTableService/data
CREATE src/app/services/dataTableService/data.service.spec.ts (347 bytes)
CREATE src/app/services/dataTableService/data.service.ts (133 bytes)
jsersan@iMac-de-Jose appJSONpages %
```

Vamos a utilizar esta url que las direcciones de varias APIs de carácter público.



Copiamos la url y la pegamos en el servicio:

```
7   export class DataService {
8     url:string = "https://api.publicapis.org/entries";
```

Para pedir este servicio debemos utilizar un cliente http y lo inyectamos en el constructor:

```
2   import { HttpClient} from '@angular/common/http';
3
4   @Injectable({
5     providedIn: 'root'
6   })
7   export class DataService {
8     url:string = "https://api.publicapis.org/entries";
9
10    constructor(private _http:HttpClient) { }
```

Ahora nos creamos un método para obtener las entradas:

```
3   import { Observable } from 'rxjs';
4   import { infoAPI } from 'src/app/models/infoAPI.model';
5
6   @Injectable({
7     providedIn: 'root'
8   })
9   export class DataService {
10    url:string = "https://api.publicapis.org/entries";
11
12    constructor(private _http:HttpClient) { }
13
14    getEntradas():Observable<infoAPI>{
15
16    }
17  }
```

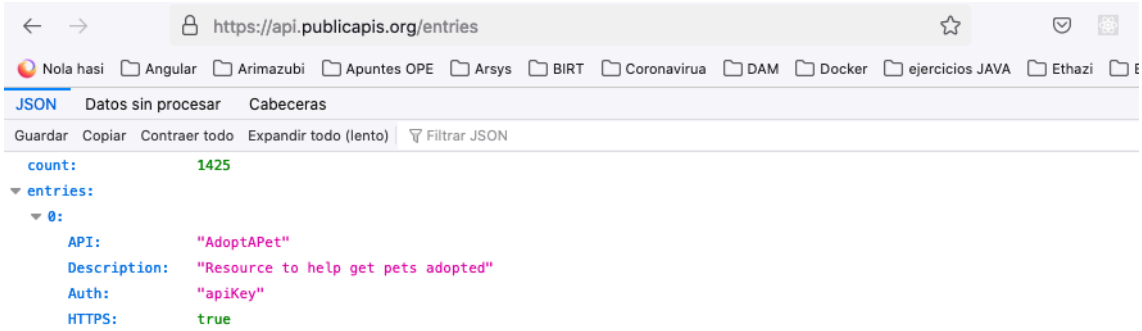
Da error porque esta función debe retornar algo:

```
14    getEntradas():Observable<infoAPI>{
15      return this._http.get(this.url)
16    }
```

Nos da error porque hay que decirle de qué tipo es lo retornado:

```
14    getEntradas():Observable<infoAPI>{
15      return this._http.get<infoAPI>(this.url)
16    }
```

Si nos fijamos, la respuesta tiene esta forma:



Modificamos la interface y creamos inforAPIResponse:

```
src > app > models > TS infoAPI.model.ts > ...
1   export interface infoAPIResponse{
2       count: number,
3       entradas: infoAPI[]
4   }
5
6   export interface infoAPI{
7       API: string,
8       Description: string,
9       Link: string,
10      Category: string,
11      Cors: string
12  }
```

Así el servicio queda:

```
4   import { infoAPIResponse } from 'src/app/models/infoAPI.model';
5
6   @Injectable({
7     providedIn: 'root'
8   })
9   export class DataService {
10    url:string = "https://api.publicapis.org/entries";
11
12    constructor(private _http:HttpClient) { }
13
14    getEntradas():Observable<infoAPIResponse>{
15      return this._http.get<infoAPIResponse>(this.url)
16    }
17  }
```

En nuestro componente ahora inyectamos el servicio en el constructor y en listar llamamos a la función getEntradas del servicio:

```
30    constructor(private _entradaServicio:DataService) {}
31
32    listar() {
33      this._entradaServicio.getEntradas();
```

Como es un observable (promesa), debemos realizar un subscribe para obtener los datos de forma asíncrona:

```
30    constructor(private _entradaServicio:DataService) {}
31
32    listar() {
33      this._entradaServicio.getEntradas().subscribe(
34        response =>{
35          console.log(response);
36          if(response.count>0){
37            this.dataSource.data = response.entradas;
38          }
39        })
40
41
42    /*  this.dataSource.data = [
43        {
44          API: 'string',
45          Description: 'string',
46          Link: 'string',
```

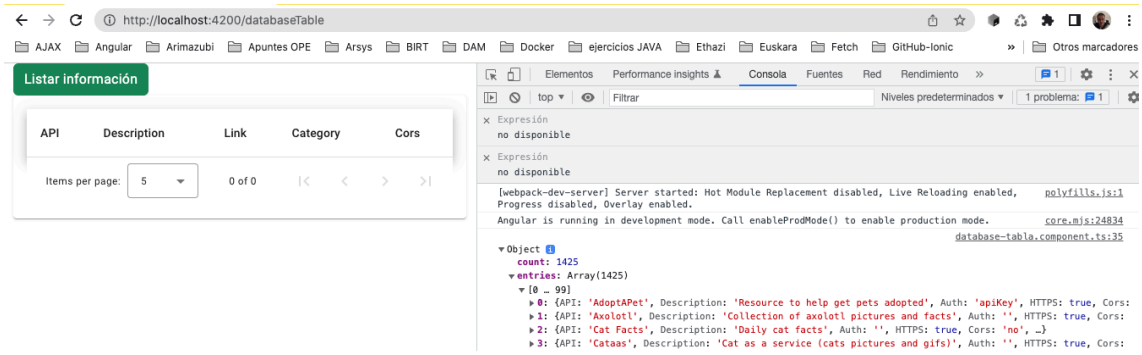
En la consola tenemos este error:

```
ERROR Error: Uncaught (in promise): NullInjectorError: R3InjectorError(AppModule)[DataService ->      main.ts:6
HttpClient -> HttpClient -> HttpClient]:
  NullInjectorError: No provider for HttpClient!
NullInjectorError: R3InjectorError(AppModule)[DataService -> HttpClient -> HttpClient -> HttpClient]:
  NullInjectorError: No provider for HttpClient!
    at NullInjector.get (core.mjs:8096:27)
    at R3Injector.get (core.mjs:8546:33)
    at R3Injector.get (core.mjs:8546:33)
    at R3Injector.get (core.mjs:8546:33)
```

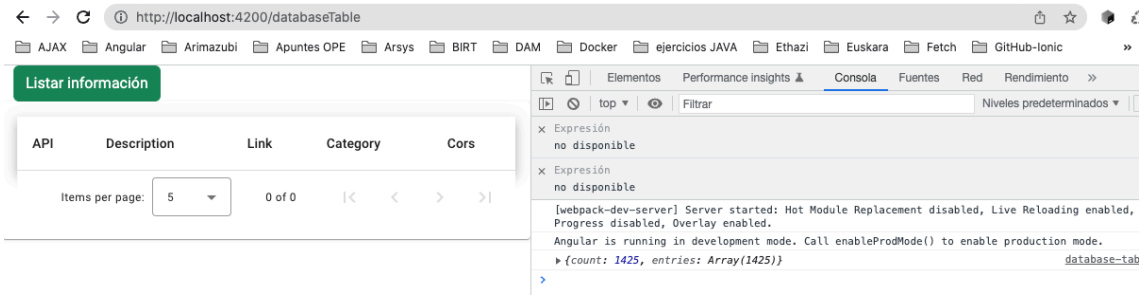
Esto es porque en app.module.ts no hemos declarado httpclient.module:

```
13  import { HttpClientModule } from '@angular/common/http';
14
15  @NgModule({
16    declarations: [
17      AppComponent,
18      DatabaseTablaComponent
19    ],
20    imports: [
21      BrowserModule,
22      AppRoutingModule,
23      BrowserAnimationsModule,
24      MatCardModule,
25      MatTableModule,
26      MatPaginatorModule,
27      HttpClientModule
28    ],
```

Guardamos y le damos al botón:



Sin embargo, no tenemos rellena la tabla pese al obtener los datos correctamente desde la API:

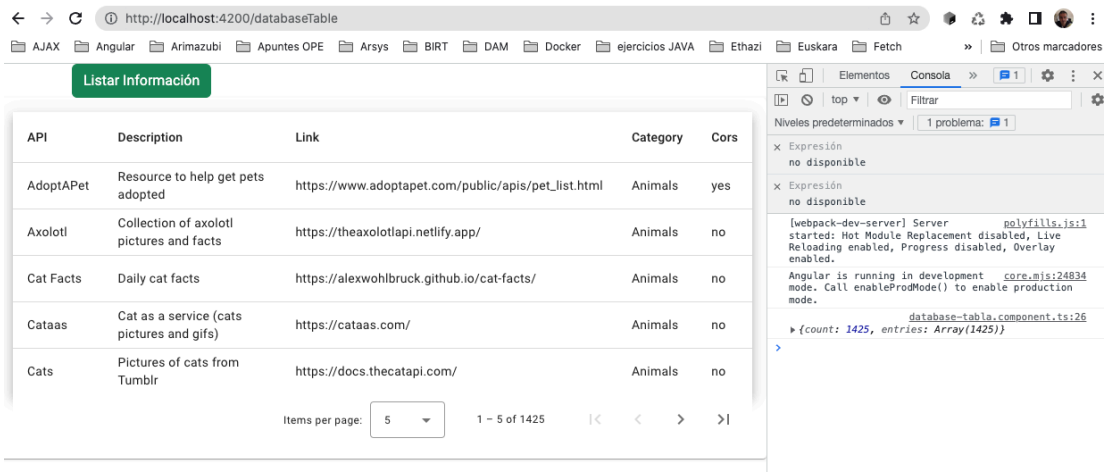


Para ello, cambiamos en el modelo las entradas como nos la devuelve la API:

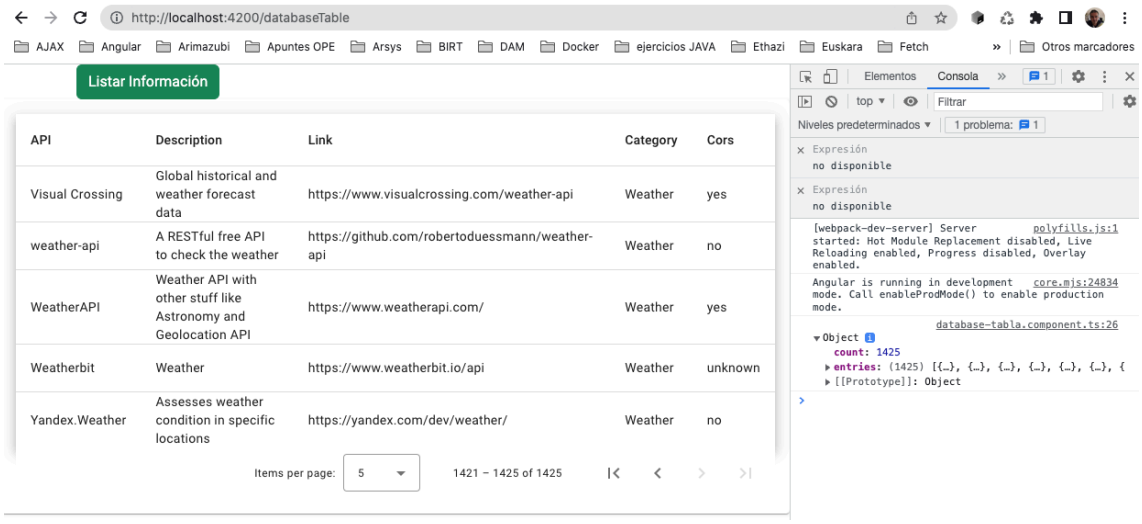
src > app > models > TS infoAPI.model.ts > infoAPIResponse > entries

```
1   export interface infoAPIResponse{
2       count: number,
3       entries: infoAPI[]
4   }
5
6   export interface infoAPI{
7       API: string,
8       Description: string,
9       Link: string,
10      Category: string,
11      Cors: string
12  }
```

Ahora:



La última página:



Podemos cambiar el número de ítems por página: