

Lab - PWA y Angular

Puedes imaginar lo letal que sería la combinación de los dos dominantes, Progressive Web Apps (PWA) y Angular. Las PWA están dominando la industria al ofrecer una experiencia de navegación ininterrumpida de los sitios web y las aplicaciones móviles. La facilidad con la que se ejecuta en segundo plano y funciona incluso en ausencia de conectividad a Internet es algo que ha dejado sorprendidos a los empresarios. Además, la PWA funciona perfectamente anclada y con capacidad de respuesta en el escritorio, el móvil y la tableta, independientemente del navegador que se elija.

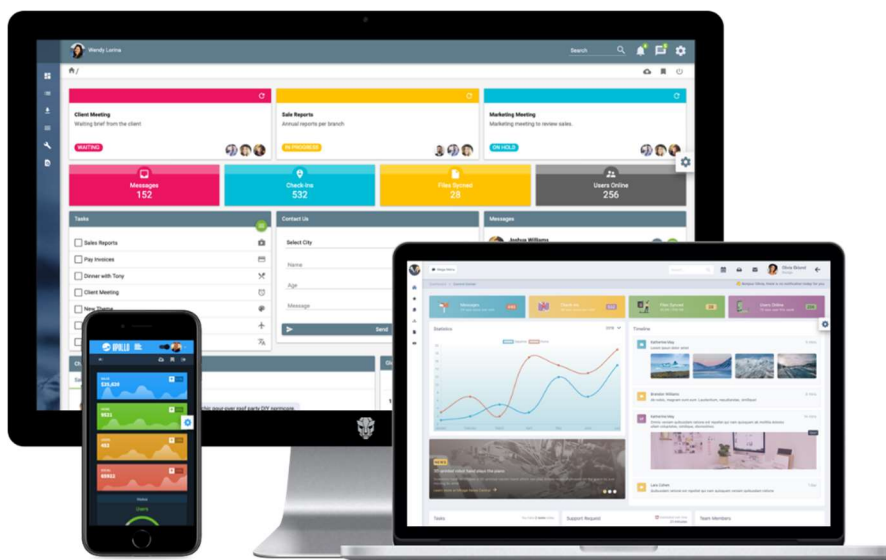
Además, la PWA no requiere una instalación en el móvil; ofrece una experiencia de aplicación nativa utilizando las URL. Puedes compartir fácilmente la aplicación utilizando la URL. En otras palabras, la velocidad y la comodidad de la PWA confirmarán el crecimiento de tu negocio.

Angular, por otro lado, es un framework PWA robusto que mejora el compromiso de la aplicación. La librería `@angular/service Worker` introducido en la versión 5 de Angular abrió las puertas del desarrollo de PWA en Angular. Sus características mejoradas facilitaron a los desarrolladores la construcción de una PWA en Angular convirtiendo el código HTML y TypeScript de Angular en JavaScript.

I. Desarrollo de la aplicación

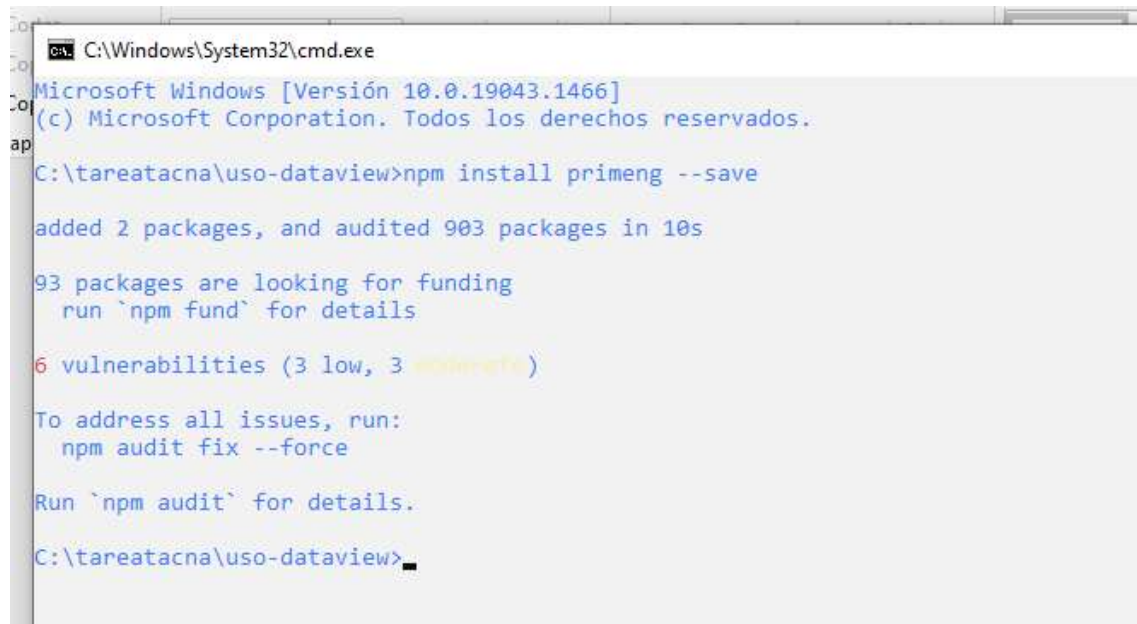
Después de crear la aplicación de este laboratorio, se va a instalar una librería de componentes visuales llamada PrimeNG (<https://www.primefaces.org/primeng/showcase/#/>).

PrimeNG es una colección de componentes de interfaz de usuario para Angular. Todos los widgets son de código abierto y de uso gratuito bajo licencia MIT. PrimeNG está desarrollado por PrimeTek Informatics, un proveedor con años de experiencia en el desarrollo de soluciones de interfaz de usuario de código abierto.



Para instalar PrimeNg en el proyecto, se debe ejecutar el siguiente comando:

```
npm install primeng -save
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\tareatacna\uso-dataview>npm install primeng --save

added 2 packages, and audited 903 packages in 10s

93 packages are looking for funding
  run `npm fund` for details

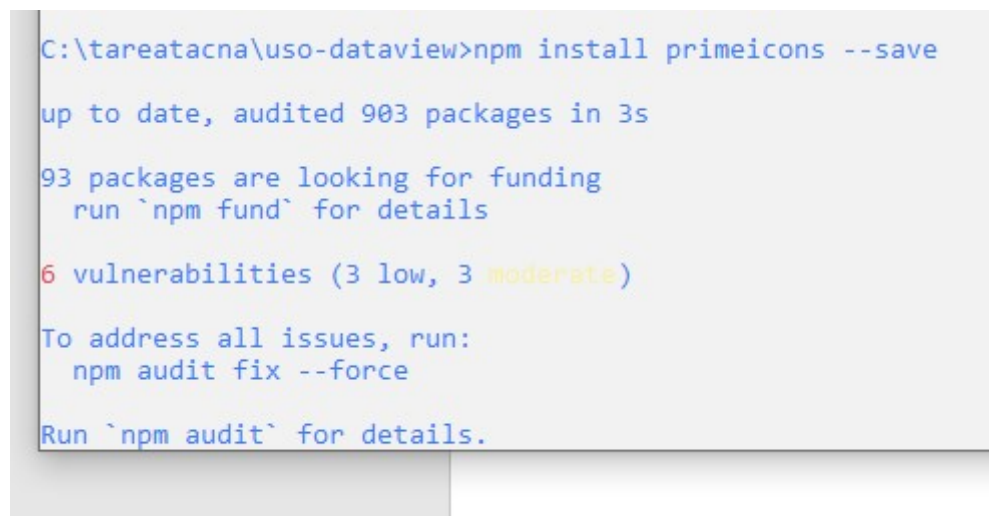
6 vulnerabilities (3 low, 3 moderate)

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.

C:\tareatacna\uso-dataview>
```

```
npm install primeicons -save
```



```
C:\tareatacna\uso-dataview>npm install primeicons --save

up to date, audited 903 packages in 3s

93 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (3 low, 3 moderate)

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
```

```
npm install font-awesome --save
```

```
C:\Windows\System32\cmd.exe

C:\tareatacna\uso-dataview>npm install font-awesome --save

added 1 package, and audited 904 packages in 4s

93 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (3 low, 3 moderate)

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.

C:\tareatacna\uso-dataview>
```

npm install @angular/cdk --save

```
C:\tareatacna\uso-dataview>npm install @angular/cdk --save

added 2 packages, and audited 906 packages in 4s

93 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (3 low, 3 moderate)

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.

C:\tareatacna\uso-dataview>
```

Si ahora vamos a package.json, veremos las siguientes dependencias de primeng

```

"private": true,
"dependencies": {
  "@angular/animations": "~13.1.0",
  "@angular/cdk": "^13.2.1",
  "@angular/common": "~13.1.0",
  "@angular/compiler": "~13.1.0",
  "@angular/core": "~13.1.0",
  "@angular/forms": "~13.1.0",
  "@angular/platform-browser": "~13.1.0",
  "@angular/platform-browser-dynamic": "~13.1.0",
  "@angular/router": "~13.1.0",
  "font-awesome": "^4.7.0",
  "primeicons": "^5.0.0",
  "primeng": "^13.1.0",
  "rxjs": "~7.4.0",
  "tslib": "^2.3.0",
  "zone.js": "~0.11.4"
},

```

Abrir angular.json y añade lo siguiente en la sección de estilos

```

"./node_modules/primeicons/primeicons.css",
"./node_modules/primeng/resources/themes/lara-light-blue/theme.css",
"./node_modules/primeng/resources/primeng.min.css",

```

```

],
"styles": [
  "./node_modules/primeicons/primeicons.css",
  "./node_modules/primeng/resources/themes/lara-light-blue/theme.css",
  "./node_modules/primeng/resources/primeng.min.css",
  "src/styles.css"
],
"scripts": []
},

```

Por ahora, se modificará el componente app de esta manera:

app.component.html

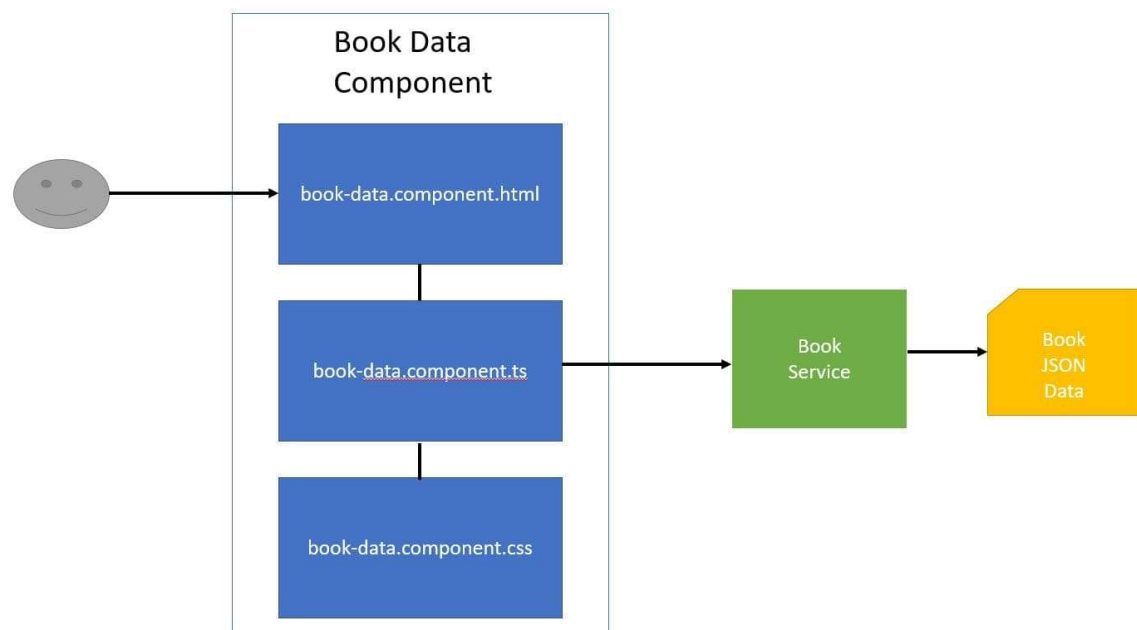
```
<router-outlet></router-outlet>
```

El componente book-data es registrado en app.module.ts

app.module.ts

```
angular.json M  app.component.html M  TS app.module.ts M X
src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { BookDataComponent } from './book-data/book-data/book-data.component';
7
8  @NgModule({
9    declarations: [
10     AppComponent,
11     BookDataComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
```

Esta será la organización del proyecto en desarrollo:

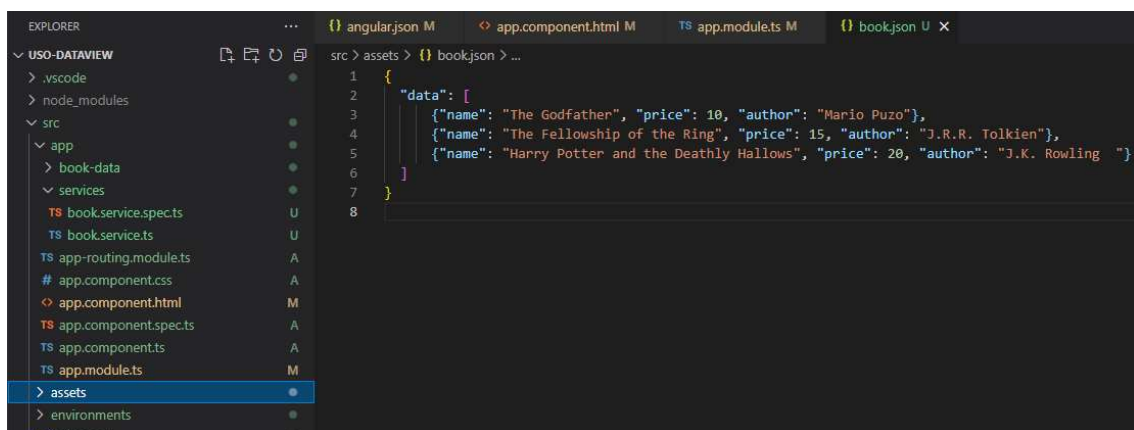


por ello, se construye una carpeta services, donde estará alojado la clase service y el archivo de datos.

Vamos a crear un servicio llamado BookService que obtendrá los datos de los libros mediante una llamada HTTP. Actualmente no vamos a hacer una llamada http a ningún servicio REST expuesto, sino que lo obtendremos de un archivo JSON llamado books.json que crearemos en la carpeta assets.

El book.json será el siguiente

```
{
  "data": [
    {"name": "The Godfather", "price": 10, "author": "Mario Puzo"},
    {"name": "The Fellowship of the Ring", "price": 15, "author": "J.R.R. Tolkien"},
    {"name": "Harry Potter and the Deathly Hallows", "price": 20, "author": "J.K. Rowling "}
  ]
}
```



luego, se creará el servicio Book, cuyo código es el siguiente:

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

export interface Book {
  name: string;
  price: number;
  author:string;
}
```



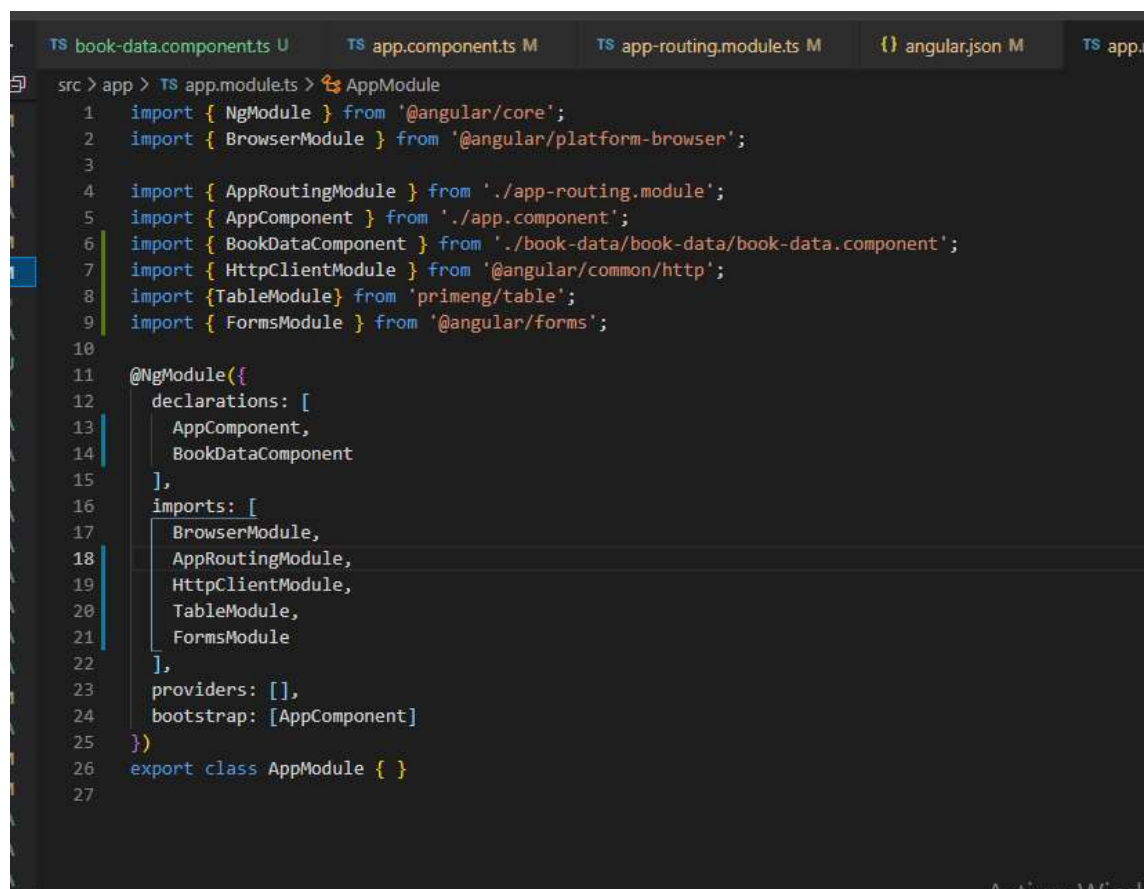
```

@Injectable({
  providedIn: 'root'
})
export class BookService {
  constructor(private http: HttpClient) {}

  getBooks() {
    return this.http.get<any>('assets/books.json')
      .toPromise()
      .then(res => <Book[]>res.data)
      .then(data => { return data; });
  }
}

```

para que HttpClient pueda ser usado, debe registrarse en app.module, además debe agregarse el registro de TableModule que es el componente visual PrimeNg que será usado en este ejemplo y FormsModule. De esa manera, app.module quedara asi:



```

src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { BookDataComponent } from './book-data/book-data/book-data.component';
7  import { HttpClientModule } from '@angular/common/http';
8  import { TableModule } from 'primeng/table';
9  import { FormsModule } from '@angular/forms';
10
11  @NgModule({
12    declarations: [
13      AppComponent,
14      BookDataComponent
15    ],
16    imports: [
17      BrowserModule,
18      AppRoutingModule,
19      HttpClientModule,
20      TableModule,
21      FormsModule
22    ],
23    providers: [],
24    bootstrap: [AppComponent]
25  })
26  export class AppModule { }
27

```

A continuación se configura `app.routing.module.ts` para hacer routing con el componente creado.

app.routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BookDataComponent } from '../book-data/book-data/book-data.component';

const routes: Routes = [
  { path: '', component: BookDataComponent }
];

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

Modifique el componente `book-data` para obtener los datos de respaldo de la tabla de datos **primeng** llamando al servicio creado anteriormente:

book-data.component.ts

```
import { Component, OnInit } from '@angular/core';
import { BookService, Book } from
'src/app/services/book.service';

@Component({
  selector: 'app-book-data',
  templateUrl: '../book-data.component.html',
  styleUrls: ['../book-data.component.css']
})
export class BookDataComponent implements OnInit {
```



```

books!: Book[];

constructor(private bookService: BookService) { }

ngOnInit() {
    this.bookService.getBooks().
        then(books => this.books = books);
}

}

```

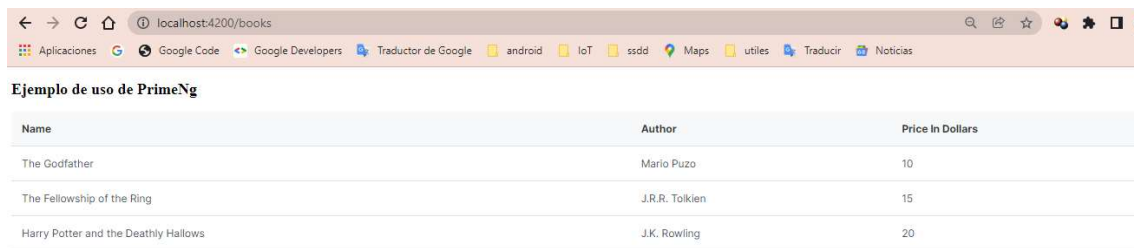
book-data.compoent.html

```

<p-table [value]="books">
    <ng-template pTemplate="header">
        <tr>
            <th>Name</th>
            <th>Author</th>
            <th>Price In Dollars</th>
        </tr>
    </ng-template>
    <ng-template pTemplate="body" let-book>
        <tr>
            <td>{{book.name}}</td>
            <td>{{book.author}}</td>
            <td>{{book.price}}</td>
        </tr>
    </ng-template>
</p-table>

```

Al ejecutar la aplicación obtenemos este resultado:



The screenshot shows a web browser window with the address bar at localhost:4200/books. The page title is 'Ejemplo de uso de PrimeNg'. Below the title is a table with three columns: Name, Author, and Price in Dollars. The table contains three rows of book data.

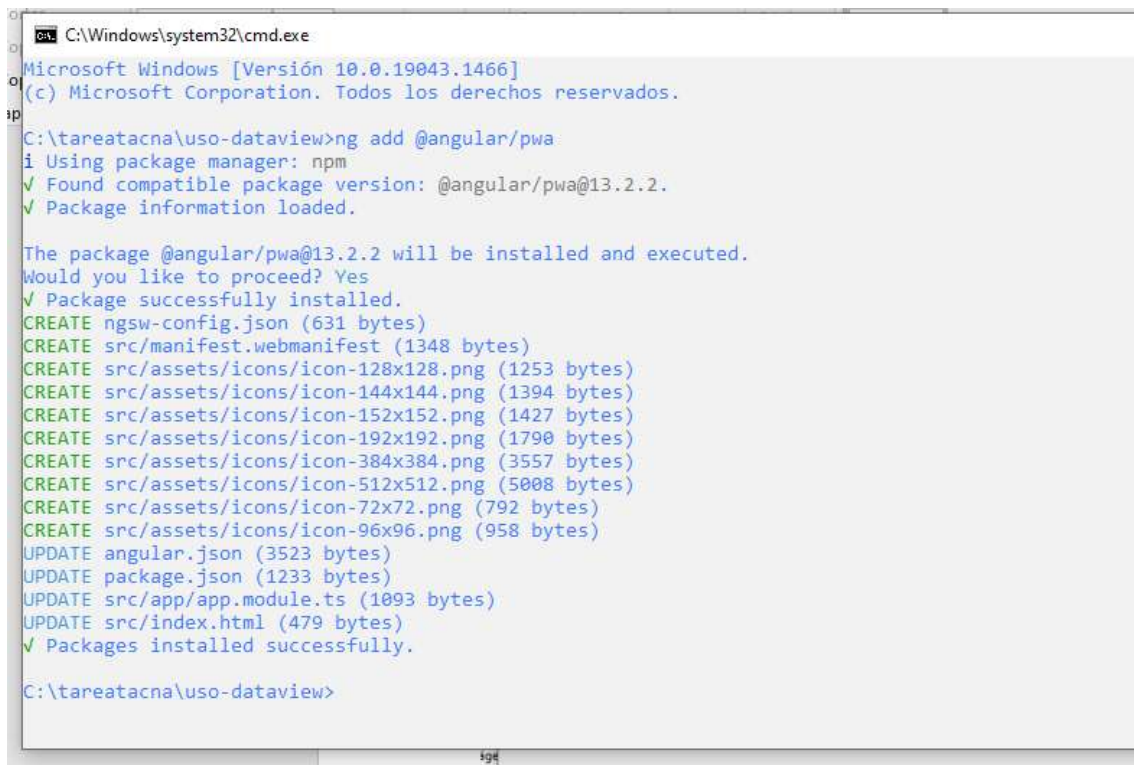
Name	Author	Price in Dollars
The Godfather	Mario Puzo	10
The Fellowship of the Ring	J.R.R. Tolkien	15
Harry Potter and the Deathly Hallows	J.K. Rowling	20

II. Conversión a Progressive web Application (PWA)

Esta característica introducida en la versión 6 del framework nos permite instalar paquetes y además ejecutar scripts. Por ejemplo, para manipular el código y automáticamente la integración.

Google usa Schematics para transformar nuestra app en PWA con este sencillo comando.

`ng add @angular/pwa`



The screenshot shows a Windows command prompt window with the title 'C:\Windows\system32\cmd.exe'. The user has entered the command 'ng add @angular/pwa'. The output shows the package manager (npm) finding a compatible version of @angular/pwa (13.2.2) and installing it. The installation process creates several files and updates existing ones.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\tareatacna\uso-dataview>ng add @angular/pwa
i Using package manager: npm
√ Found compatible package version: @angular/pwa@13.2.2.
√ Package information loaded.

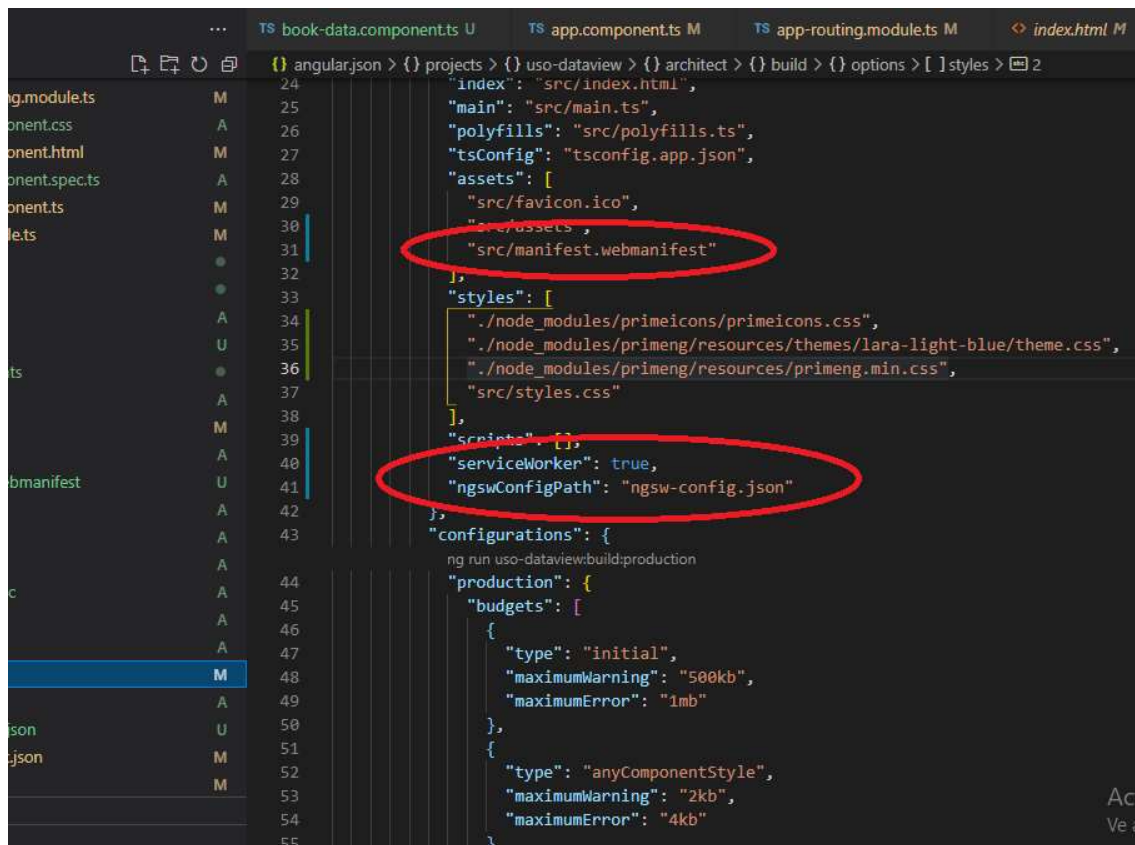
The package @angular/pwa@13.2.2 will be installed and executed.
Would you like to proceed? Yes
√ Package successfully installed.
CREATE ngsw-config.json (631 bytes)
CREATE src/manifest.webmanifest (1348 bytes)
CREATE src/assets/icons/icon-128x128.png (1253 bytes)
CREATE src/assets/icons/icon-144x144.png (1394 bytes)
CREATE src/assets/icons/icon-152x152.png (1427 bytes)
CREATE src/assets/icons/icon-192x192.png (1790 bytes)
CREATE src/assets/icons/icon-384x384.png (3557 bytes)
CREATE src/assets/icons/icon-512x512.png (5008 bytes)
CREATE src/assets/icons/icon-72x72.png (792 bytes)
CREATE src/assets/icons/icon-96x96.png (958 bytes)
UPDATE angular.json (3523 bytes)
UPDATE package.json (1233 bytes)
UPDATE src/app/app.module.ts (1093 bytes)
UPDATE src/index.html (479 bytes)
√ Packages installed successfully.

C:\tareatacna\uso-dataview>
```

Después de la ejecución de esta librería, se han modificado 4 archivos:

a. angular.json

se marcan los añadidos..



```
angular.json > {} projects > {} uso-dataview > {} architect > {} build > {} options > [ ] styles > 2
  "index": "src/index.html",
  "main": "src/main.ts",
  "polyfills": "src/polyfills.ts",
  "tsConfig": "tsconfig.app.json",
  "assets": [
    "src/favicon.ico",
    "src/assets",
    "src/manifest.webmanifest"
  ],
  "styles": [
    "./node_modules/primeicons/primeicons.css",
    "./node_modules/primeng/resources/themes/lara-light-blue/theme.css",
    "./node_modules/primeng/resources/primeng.min.css",
    "src/styles.css"
  ],
  "scripts": [],
  "serviceWorker": true,
  "ngswConfigPath": "ngsw-config.json"
},
"configurations": {
  "ng run uso-dataview:build:production": {
    "production": {
      "budgets": [
        {
          "type": "initial",
          "maximumWarning": "500kb",
          "maximumError": "1mb"
        },
        {
          "type": "anyComponentStyle",
          "maximumWarning": "2kb",
          "maximumError": "4kb"
        }
      ]
    }
  }
}
```

b. ngsw-config.json

Este es un archivo de configuración nuevo, específicamente para operaciones PWA

```
... TS book-data.component.ts U TS app.component.ts M TS app-routing.module.ts M {} angular.json M {} ng
{} ngsw-config.json > ...
1 {
2   "$schema": "../node_modules/@angular/service-worker/config/schema.json",
3   "index": "/index.html",
4   "assetGroups": [
5     {
6       "name": "app",
7       "installMode": "prefetch",
8       "resources": {
9         "files": [
10          "/favicon.ico",
11          "/index.html",
12          "/manifest.webmanifest",
13          "/*.css",
14          "/*.js"
15        ]
16      }
17    },
18    {
19      "name": "assets",
20      "installMode": "lazy",
21      "updateMode": "prefetch",
22      "resources": {
23        "files": [
24          "/assets/**",
25          "/*.svg|cur|jpg|jpeg|png|apng|webp|avif|gif|otf|ttf|woff|woff2"
26        ]
27      }
28    }
29  ]
30 }
31 }
```

c. app.module.ts

Se ha agregado a environment y al modulo ServiceWorkerModule

```
src > app > TS app.module.ts > AppModule
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { BookDataComponent } from './book-data/book-data.component';
7 import { HttpClientModule } from '@angular/common/http';
8 import { TableModule } from 'primeng/table';
9 import { FormsModule } from '@angular/forms';
10 import { ServiceWorkerModule } from '@angular/service-worker';
11 import { environment } from '../environments/environment';
12
13 @NgModule({
14   declarations: [
15     AppComponent,
16     BookDataComponent
17   ],
18   imports: [
19     BrowserModule,
20     AppRoutingModule,
21     HttpClientModule,
22     TableModule,
23     FormsModule,
24     ServiceWorkerModule.register('ngsw-worker.js', {
25       enabled: environment.production,
26       // Register the ServiceWorker as soon as the app is stable
27       // or after 30 seconds (whichever comes first).
28       registrationStrategy: 'registerWhenStable:30000'
29     })
30   ],
31   providers: [],
32   bootstrap: [AppComponent]
33 })
```

d. index.html

```
<> index.html M X
src > <> index.html > ...
1  <!doctype html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>UsoDataview</title>
6    <base href="/">
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="manifest" href="manifest.webmanifest">
10   <meta name="theme-color" content="#1976d2">
11 </head>
12 <body>
13   <app-root></app-root>
14   <noscript>Please enable JavaScript to continue using this application.</noscript>
15 </body>
16 </html>
17
```

Con esto, ya tendríamos una PWA, aunque nos queda adaptarlo y añadir las mejoras que se planteen en el futuro.

Antes de nada vamos a compilarlo y probarlo para ver si funciona. Para ello, se debe editar **package.json** y en la zona de scripts agregar esto:

"prod": "ng build --prod"

```
version: 0.0.0,
  > Debug
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test",
    "prod": "ng build --prod"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~13.1.0",
```

A continuación, se ejecutara el siguiente comando:

npm run prod


```

C:\tareatacna\uso-dataview>npm run prod

> uso-dataview@0.0.0 prod
> ng build --prod

Option "--prod" is deprecated: No need to use this option as this builder defaults to configuration "production".
✓ Browser application bundle generation complete.
✓ Copying assets complete.
✓ Index html generation complete.
✓ Service worker generation complete.

Initial Chunk Files | Names | Raw Size | Estimated Transfer Size
main.c14353c439efec44.js | main | 514.52 kB | 113.57 kB
styles.4fb0ed0ee70be183.css | styles | 151.20 kB | 15.78 kB
polyfills.738b5e610941e0c6.js | polyfills | 36.23 kB | 11.48 kB
runtime.c041f4138aa8d615.js | runtime | 1.05 kB | 608 bytes

| Initial Total | 703.00 kB | 141.43 kB

Build at: - Hash: - Time: ms

Warning: bundle initial exceeded maximum budget. Budget 500.00 kB was not met by 203.00 kB with a total of 703.00 kB.

```

Nota importante:

Algunas veces, al ejecutarse este comando, podrá aparecer el siguiente error:

```

C:\tareatacna\uso-dataview>npm run prod

> uso-dataview@0.0.0 prod
> ng build --prod

Option "--prod" is deprecated: No need to use this option as this builder defaults to
✓ Browser application bundle generation complete.
An unhandled exception occurred: Transform failed with 1 error:
error: Invalid version: "15.2-15.3"
See "C:\Users\pc\AppData\Local\Temp\ng-fl1PeV\angular-errors.log" for further details.

```

y al revisar el error, aparecerá la razón:

```

angular-errors.log: 000 de 000
Archivo Edición Formato Ver Ayuda
[error] HookWebpackError: Transform failed with 1 error:
error: Invalid version: "15.2-15.3"
    at makeWebpackError (C:\tareatacna\uso-dataview\node_modules\webpack\lib\HookWebpackError.js:48:9)
    at C:\tareatacna\uso-dataview\node_modules\webpack\lib\Compilation.js:3055:12
    at eval (eval at create (C:\tareatacna\uso-dataview\node_modules\tapable\lib\HookCodeFactory.js:33:10), <anonymous>:98:1)
    at processTicksAndRejections (node:internal/process/task_queues:96:5)
-- inner error --
Error: Transform failed with 1 error:
error: Invalid version: "15.2-15.3"
    at failureErrorWithLog (C:\tareatacna\uso-dataview\node_modules\esbuild\lib\main.js:1557:15)
    at C:\tareatacna\uso-dataview\node_modules\esbuild\lib\main.js:1346:29
    at C:\tareatacna\uso-dataview\node_modules\esbuild\lib\main.js:637:9
    at handleIncomingPacket (C:\tareatacna\uso-dataview\node_modules\esbuild\lib\main.js:734:9)
    at Socket.readFromStdout (C:\tareatacna\uso-dataview\node_modules\esbuild\lib\main.js:604:7)
    at Socket.emit (node:events:390:28)
    at addChunk (node:internal/streams/readable:315:12)
    at readableAddChunk (node:internal/streams/readable:289:9)
    at Socket.Readable.push (node:internal/streams/readable:228:10)
    at Pipe.onStreamRead (node:internal/stream_base_commons:199:23)

```

una de las formas de resolverlo, es editar el archivo .browserslistsrc.ts y agregar las siguientes líneas:

not ios_saf 15.2-15.3

not safari 15.2-15.3

quedando de la siguiente manera:

```
package.json M .browserslistrc 9+, M X
.browserslistrc
1 # This file is used by the build system
2 # For additional information regarding
3 # https://github.com/browserslist/browserslist
4
5 # For the full list of supported browsers
6 # https://angular.io/guide/browser-support
7
8 # You can see what browsers were selected
9 # npx browserslist
10
11 last 1 Chrome version
12 last 1 Firefox version
13 last 2 Edge major versions
14 last 2 Safari major versions
15 last 2 iOS major versions
16 not ios_saf 15.2-15.3
17 not safari 15.2-15.3
18 Firefox ESR
19
```

con lo que se resuelve el problema.

Al terminar la ejecución del comando indicado, se creará un nuevo directorio en la raíz llamado “dist/uso-dataview”.

po > SISTEMA (C:) > tareatacna > uso-dataview > dist >			
Nombre	Fecha de modificación	Tipo	Tamaño
uso-dataview	7/02/2022 15:30	Carpeta de archivos	

Esto está especificado en el outputPath del archivo “angular.json”, si quisieramos cambiarlo lo hacemos desde ese apartado.


```

"root": "",
"sourceRoot": "src",
"prefix": "app",
"architect": {
  "build": {
    "builder": "@angular-devkit/build-angular:browser",
    "options": {
      "outputPath": "dist/uso-dataview",
      "index": "src/index.html",
      "main": "src/main.ts",
      "polyfills": "src/polyfills.ts",
      "tsConfig": "tsconfig.app.json",
      "assets": [
        "src/favicon.ico",
        "src/assets",
        "src/manifest.webmanifest"
      ],
      "styles": [
        "./node_modules/primeicons/primeicons.css",
        "./node_modules/primeng/resources/themes/lara-light-blue/theme.css",
        "./node_modules/primeng/resources/primeng.min.css",
        "src/styles.css"
      ],
      "scripts": [],

```

Instalamos las dependencias de manera global para probar aplicaciones Angular (añadir sudo antes si estamos trabajando con Linux o MacOSX):

`npm install -g angular-http-server`



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\tareatacna\uso-dataview>npm install -g angular-http-server

added 14 packages, and audited 15 packages in 5s

found 0 vulnerabilities

C:\tareatacna\uso-dataview>

```

Una vez instalado, accedemos al proyecto compilado y lo inicializamos:

`cd dist/uso-dataview`

Ejecutamos para inicializar:

`angular-http-server`

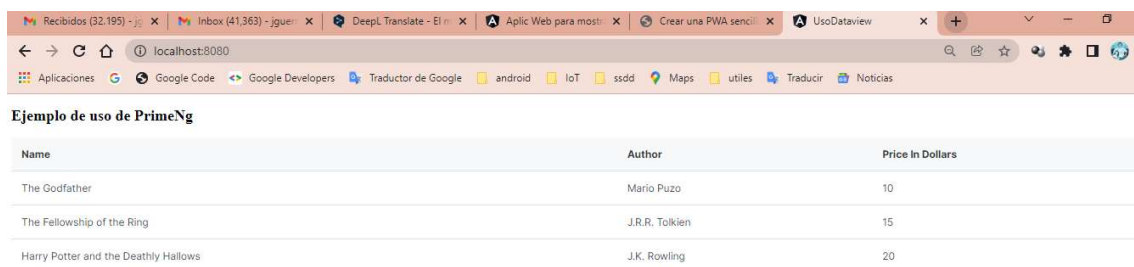
```
Microsoft Windows [Versión 10.0.19043.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\tareatacna\uso-dataview\dist\uso-dataview>angular-http-server
Serving from path: C:\tareatacna\uso-dataview\dist\uso-dataview
Using default file: C:\tareatacna\uso-dataview\dist\uso-dataview\index.html
Listening on 8080
```

Por defecto se inicializa en el puerto 8080 y su url será:

`http://localhost:8080`

Abrimos en el Google Chrome y tendrá esta apariencia:

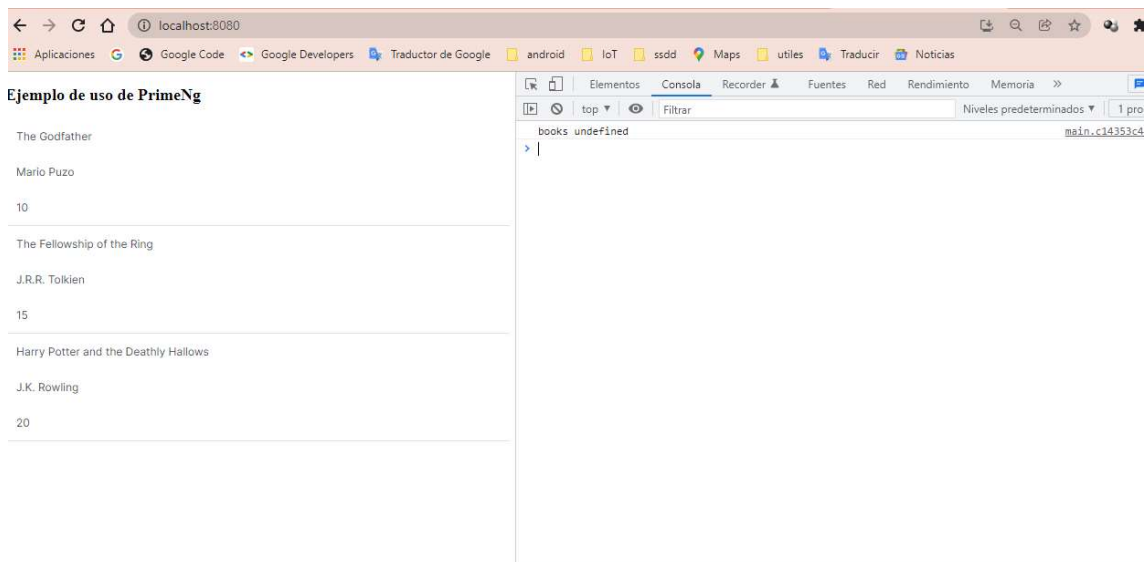


Como se observa, no hay diferencias respecto a una aplicación Angular normal y corriente, y así es. Únicamente tiene añadidas funcionalidades para que se pueda visualizar el contenido en los dispositivos móviles con mayor rendimiento y que la experiencia de usuario sea lo mas agradable posible.

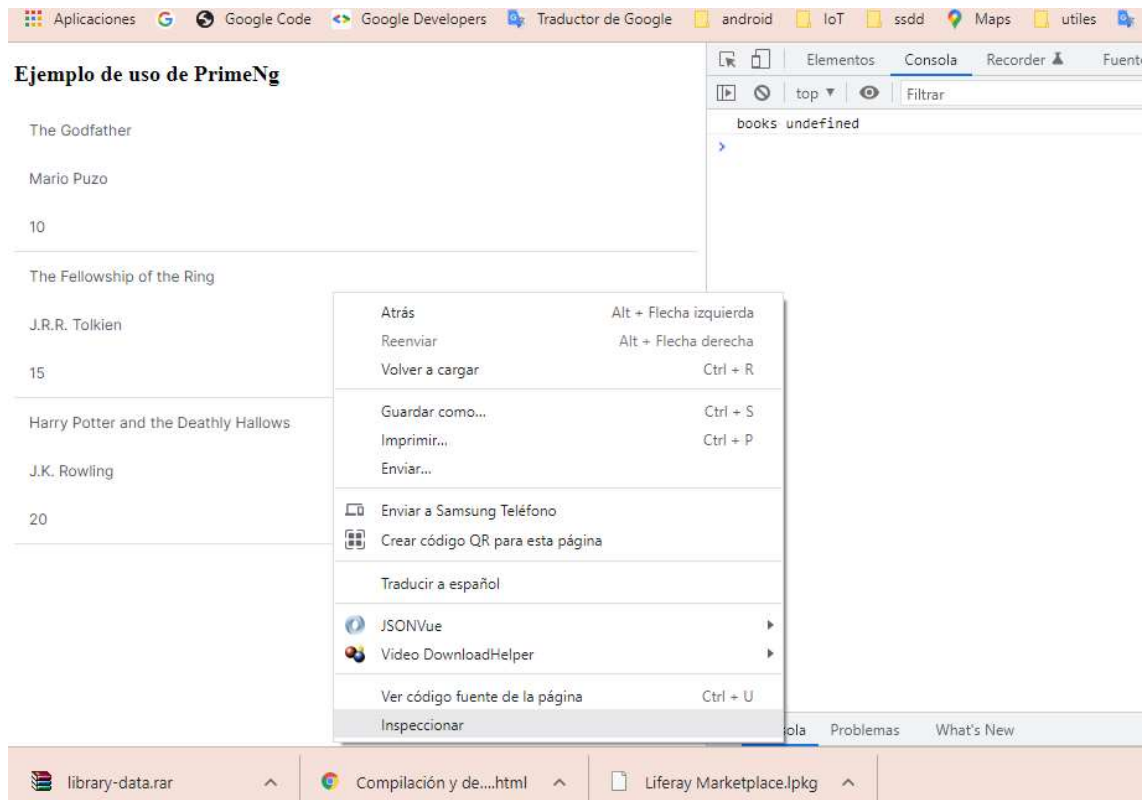
Herramientas probar si la aplicación es PWA

La mejor y mas sencilla forma de probar el cumplimiento de condiciones de PWA es la propia consola de desarrolladores.

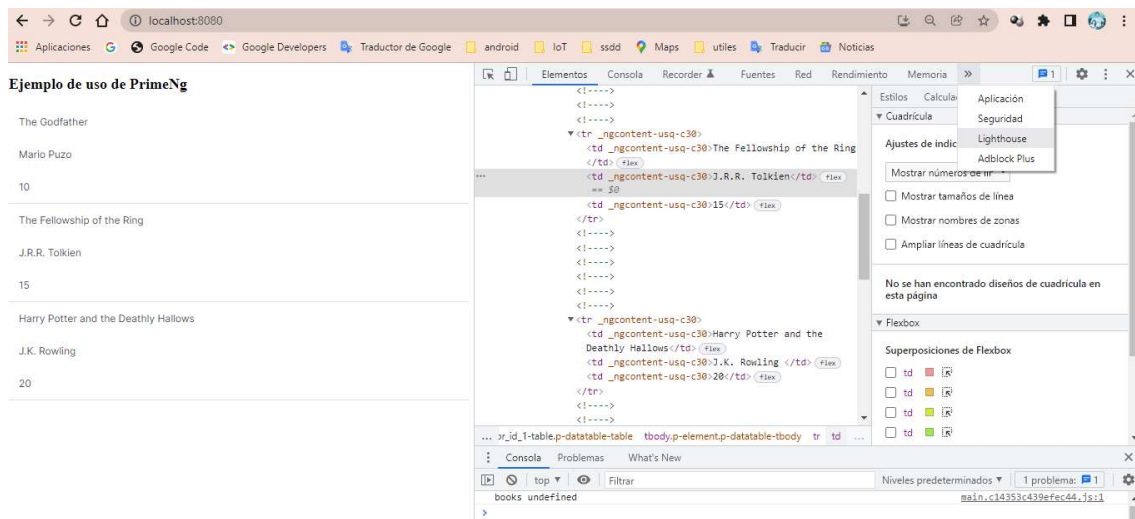
1. Debe ejecutarse el proyecto en modo dist (localhost:8080)
2. abrir la herramienta *consola de desarrolladores*



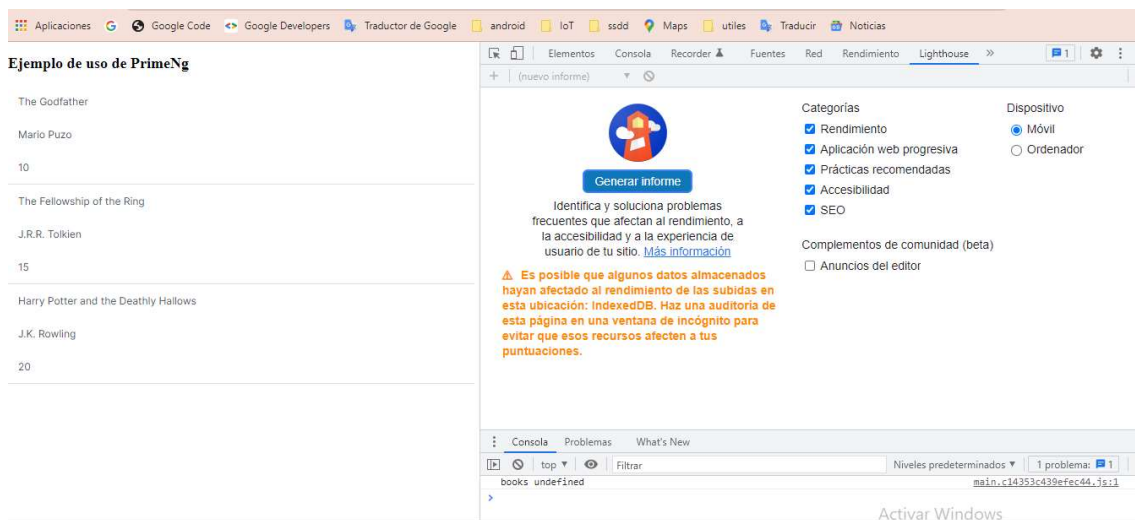
3. se hace en cualquier parte de la aplicación en ejecución, y luego botón derecho, y se escoge **inspeccionar..**



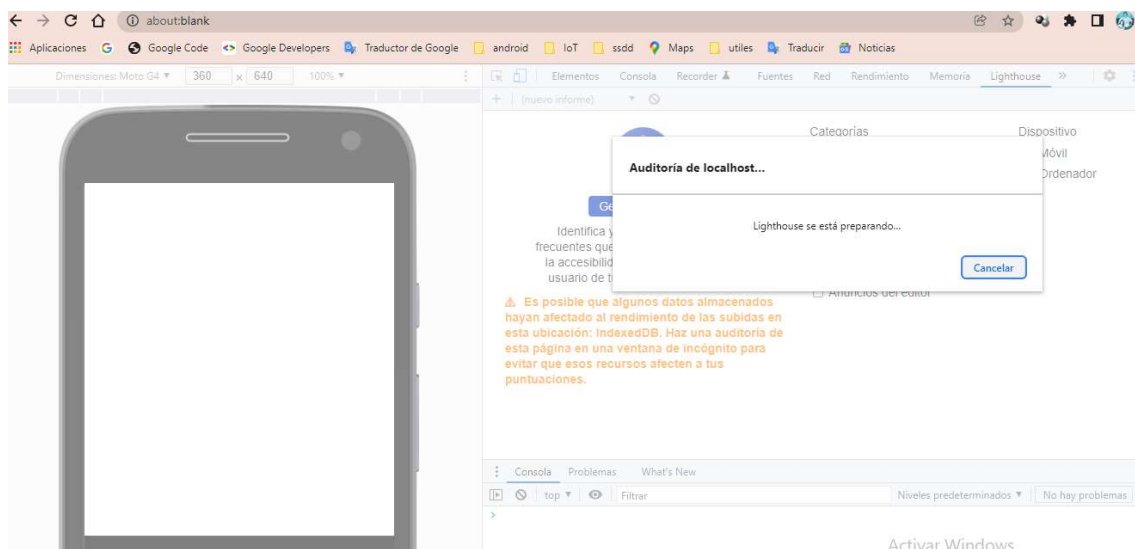
4. Se abrirá la inspección de código, que permite ver como las líneas de código, se están ejecutando. A continuación, se hace clic en el menú y se escoge **lighthouse**.

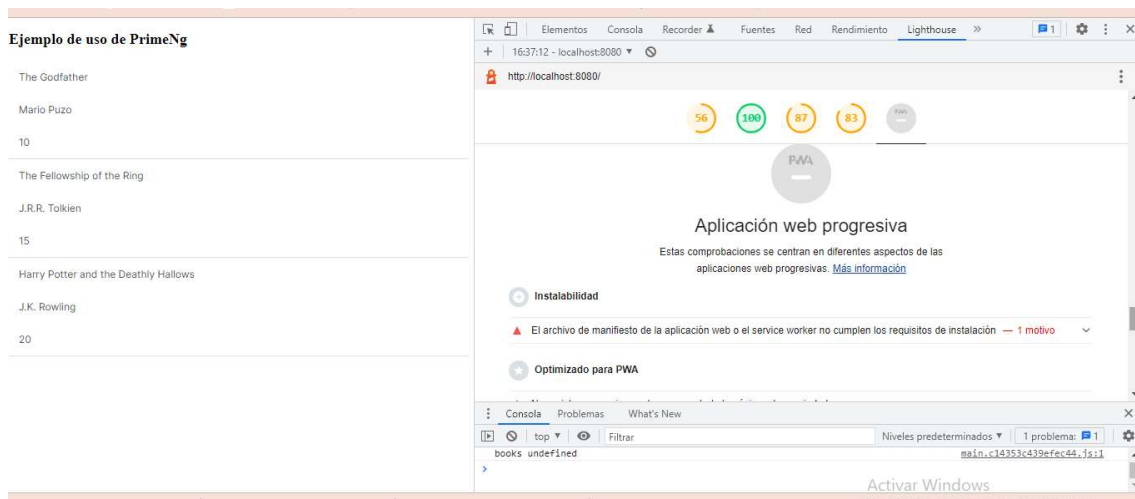


5. Esta opción revisará varias características de la operatividad del programa, entre ellas las características PWA



Al presionar el botón comienza la evaluación:





Al final, mostrara un informe detallado de la aplicación y consejos para optimizar su funcionamiento.



Otro de los aspectos que nos dicen a mejorar es el tener que añadir el “apple touch icon” que es el icono que aparece en el escritorio de un dispositivo iOS (iPhone y iPad) cuando pones el enlace a una aplicación o a un sitio web. Por defecto, para los sitios web iOS añade una pequeña captura del sitio.

Para optimizar esto, simplemente tenemos que ir a index.html y añadir dentro de las etiquetas “<head></head>”

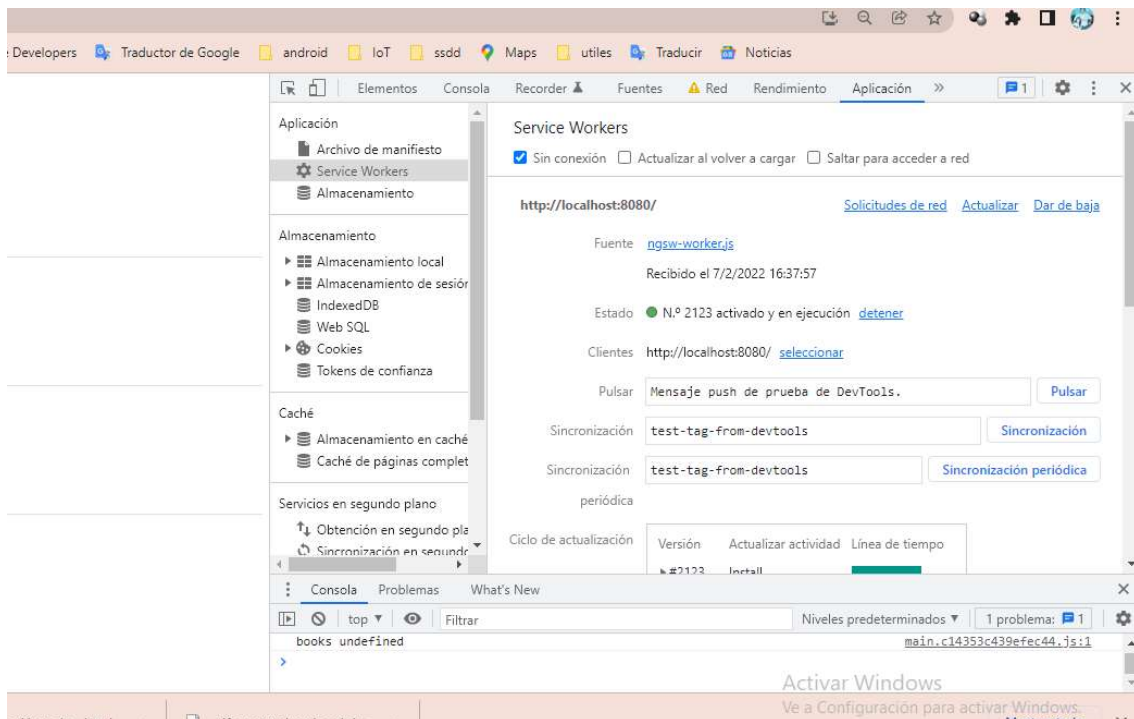
```
<link rel="apple-touch-icon" href="assets/icons/icon-512x512.png">
```

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>Uso de Dataview en Angular</title>
6   <base href="/">
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="manifest" href="manifest.webmanifest">
10  <meta name="theme-color" content="#1976d2">
11  <link rel="apple-touch-icon" href="assets/icons/icon-512x512.png">
12 </head>
13 <body>
14   <app-root></app-root>
15   <noscript>Please enable JavaScript to continue using this application.</noscript>
16 </body>
17 </html>
18
```

Ahora vamos a comprobar que el service worker esta funcionando también y vamos a probarlo dejando la web sin conexión a internet para que se cargue directamente de la caché.

Con esto comprobamos que si la instalan después podrán verla estando offline.

Para hacer esto nos vamos a la pestaña "Application" y en la sección "Service Workers" marcamos la casilla "Offline". Al hacer esto si navegamos a cualquier web de Internet nos dará un error, sin embargo nuestra web deberá seguir cargando.





Sin conexión a Internet

Prueba a:

- Comprobar los cables de red, el módem y el router
- Volver a conectarte a una red Wi-Fi
- [Ejecutar Diagnósticos de red de Windows](#)

ERR_INTERNET_DISCONNECTED



Ejemplo de uso de PrimeNg

Name	Author	Price in Dollars
The Godfather	Mario Puzo	10
The Fellowship of the Ring	J.R.R. Tolkien	15
Harry Potter and the Deathly Hallows	J.K. Rowling	20