

Curso de Consumo de APIs REST con Angular



Descarga de archivos con Http 21/23

RECURSOS

APUNTES

Una necesidad muy común desde un front-end es la **descarga de archivos provenientes de un servidor**. Veamos cómo puedes manipular este tipo de información y lograr que el usuario pueda descargar un archivo.

Manipulando datos binarios

Los datos se transladan en forma binaria a través del canal de comunicación HTTP. Vamos a recibir dichos datos binarios y a manipularlos para crear el archivo.

1. Servicio para el manejo de archivos

Comienza creando un servicio para realizar peticiones y manipular archivos.

```
// services/files.service.ts
import { Injectable } from '@angular/core';

@Injectable({
   providedIn: 'root'
})
export class FilesService {
   constructor() { }
```

```
getFile(urlFile: string): Promise<any> {
    return fetch(urlFile, {
        method: 'GET',
     });
  }
}
```

En el método getFile() hace un fetch a una URL para obtener el **Buffer** del archivo.

Un **Buffer** es un espacio en la memoria que almacena datos binarios. En NodeJS, es posible acceder a estos espacios de memoria con la clase Buffer. Los búferes almacenan una secuencia de enteros, de forma similar a una matriz en Javascript. Del lado del front-end, es posible recibir un búfer de datos y manipularlo para convertirlo en un PDF, un XLS, entre otros.

2. Convertir datos binario en un archivo

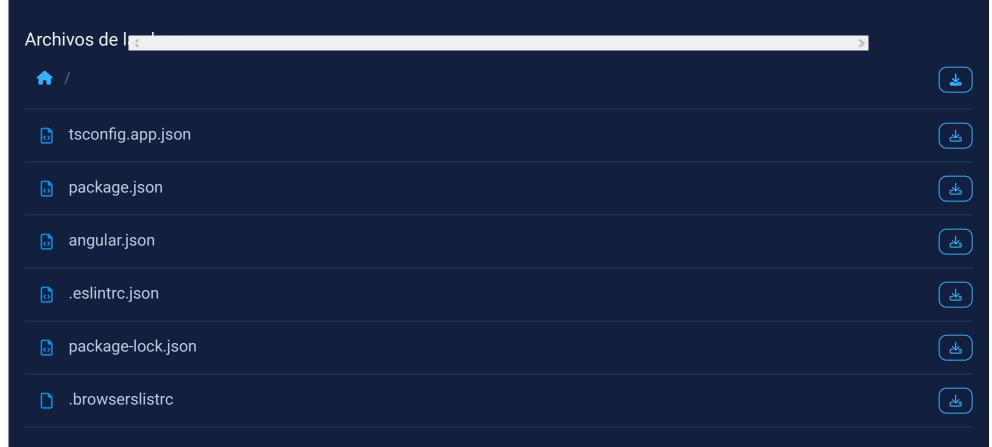
**I realizar la solicitud y recibir el Buffer, lo convertimos en Blob para posteriormente creamos una URL temporal con URL.createObjectURL() pasándole dicho Blob y abriendo una nueva pestaña utilizando window.open() en el navegador.

```
openFile(): void {
  this.filesService.getFile('../../assets/dummy.pdf')
    .then(response => response.blob())
    .then(pdf => {
      window.open(URL.createObjectURL(pdf), '_blank');
    })
    .catch(err => {
      console.log(err);
    }
}
```

});
}

La manipulación de archivos es esencial, cada navegador interpreta los tipos de datos de una manera distinta. Es común encontrarse con comportamientos distintos en cada navegador como por ejemplo, *Chrome* suele descargar directamente el archivo, mientras que *Firefox* lo abre en una nueva pestaña. Dependiendo tu necesidad, asegúrate de probar tu aplicación en varios navegadores diferentes.

Contribución creada por: Kevin Fiorentino.



obs-promises.js	(7)
tsconfig.spec.json	<u></u>
proxy.config.json	(7)
tsconfig.json	<u></u>
☐ README.md	<u></u>
.editorconfig	<u></u>
🗟 karma.conf.js	<u></u>
■ src	