Back End

Carrera Programador full-stack

HTTP Server - NEST

¿Qué es un servidor web?

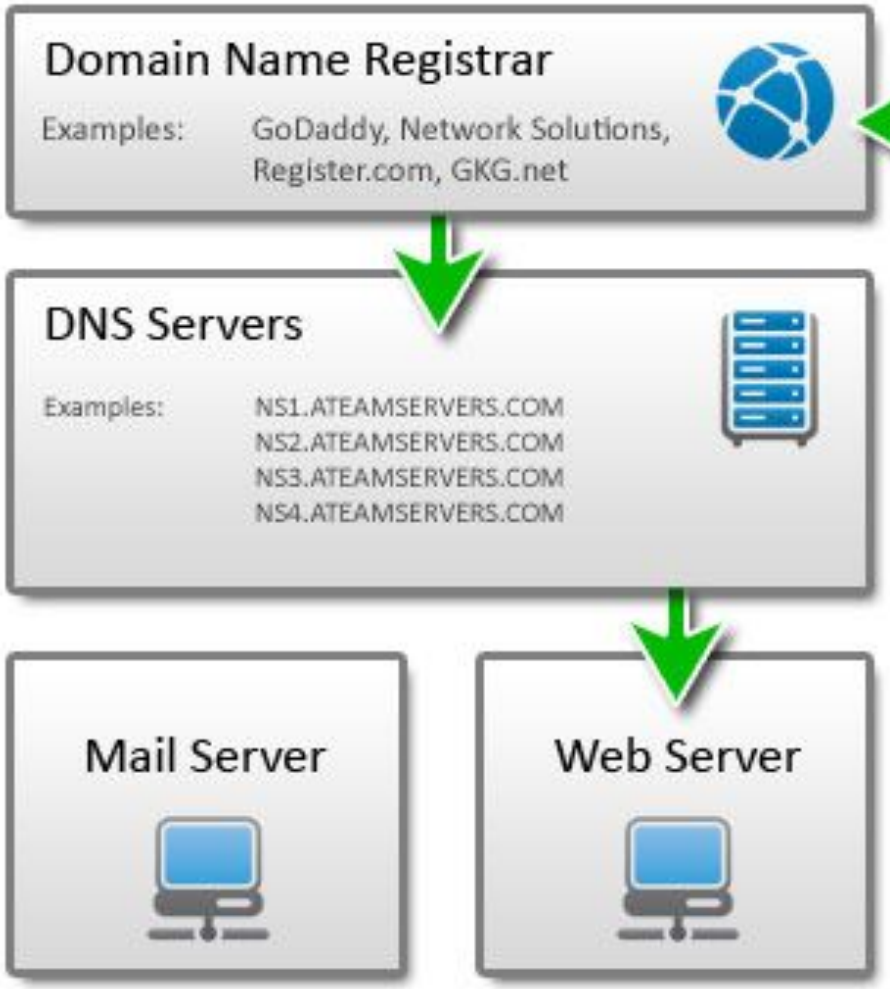
Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor realizando:

- conexiones bidireccionales y/o unidireccionales
- conexiones síncronas o asíncronas con el cliente
- generando o cediendo una respuesta

¿Qué hace un servidor web?

- La versión más simple de servidores web sirven archivos estáticos
- Simplemente envía el archivo que le pidieron en un response HTTP.
- Estos archivos son HTML, CSS, JS, imágenes, audio, etc.

DNS



Es un conjunto de protocolos y servicios que permite a los usuarios utilizar nombres en vez de tener que recordar direcciones IP numéricas.

HTTP

- HTTP: Hypertext Transfer Protocol, ‘protocolo de transferencia de hipertextos’
- Y su variante segura HTTPS
- Los usamos todo el tiempo para navegar por Internet
- Permite hacer pedidos de recursos a un servidor

HTTP y el navegador

El navegador es un cliente HTTP

Solicita archivos al servidor, lee y dibuja esos archivos

Elige qué otros archivos descargar (CSS, img, js, etc)

Hay códigos de respuesta:

- 200: OK
- 404: Not found
- 500: Internal server error

Lista completa de códigos de respuesta HTTP

Link a MDN (*remember, MDN is your friend*)

https://developer.mozilla.org/es/docs/Web/HTTP/Status

Nest: un framework backend

Nest o NestJS es un framework para construir aplicaciones Node.JS eficientes, y escalables

Combina OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming)

Documentación oficial del framework:

https://docs.nestjs.com/

Crear el proyecto

Vamos a hacer un web server de un sitio estático en Nest:

Instalar Nest CLI Tool:

```
npm i -g @nestjs/cli
```

Crear proyecto NEST:

```
nest new tracks-manager
```

Entrar a la carpeta y commitear el código inicial:

```
cd tracks-manager
git status
git add .
git status
git commit -m "Initial Project commit"
... (sigue) conectar un remoto en git y pushear ...
```

Conectar un remoto en git y pushear

Crear un proyecto en Github

Después:

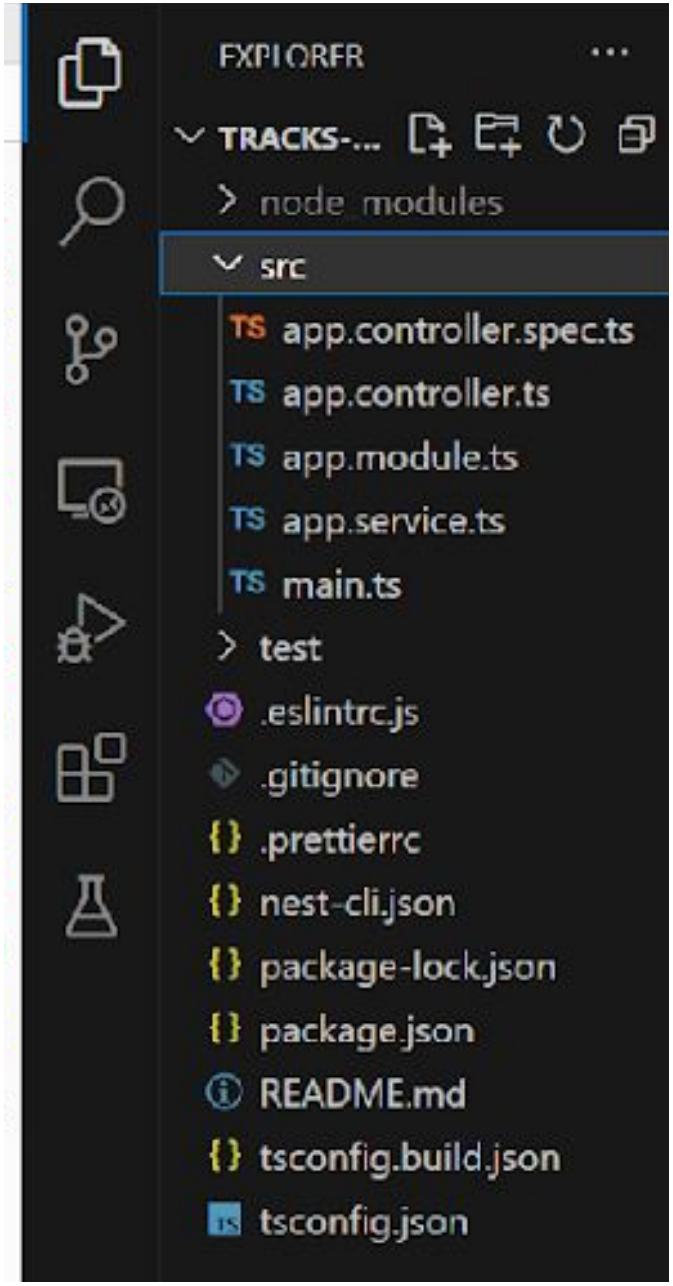
Opción A:

//se supone que el repositorio está vacío (sin commits)

```
git remote add origin URL_DE_GITHUB
git push -u origin main
```

Opción B:

- Hacer un clone
- Mover los archivos ahi (**incluir .gitignore**)



SCAFFOLDING

Este es el **andamiaje** que nos provee Nest JS a partir del cual comenzaremos a desarrollar nuestro backend.

Este andamiaje **refleja la arquitectura del *framework*** y sobre ese modelo de arquitectura se irán creando los artefactos de software que nuestro proyecto requiera.

Iniciar el proyecto

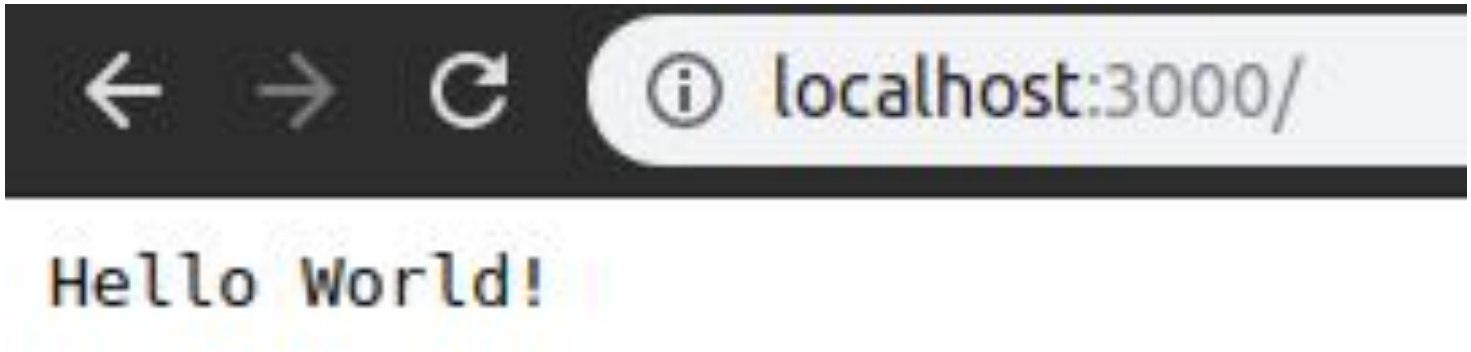
Iniciar el servidor:

```
npm run start:dev
```

Verlo en el navegador

http://localhost:3000

Ver el Hello World



API (para el futuro)

En el archivo app.controller.ts vemos una línea:

```
@Controller()
```

Este archivo hace un *“return”* que es lo que vemos en el navegador

Esto lo vamos a usar más adelante

Vamos a cambiarla a

```
@Controller('api')
```


localhost:3000/api
Hello World!

¿Qué está pasando?

- localhost: nombre del servidor o URL, en este caso localhost es un alias/apodo para nuestra PC.
- 3000: puerto, un “oído” en la PC asignado a un programa en particular.
- /api: endpoint, ruta que indica al servidor NEST que ejecutar.

¿Qué está pasando?

El mensaje ***Hello World!*** que vemos en el navegador está generado por esta línea en el archivo app.service.ts:

```
return 'Hello World!';
```

Esa línea está dentro de la función getHello en ese archivo.

En el archivo app.controller.ts vemos la llamada a la funcion getHello del servicio:

```
return this.appService.getHello();
```

De esta manera se genera la información en el Back End: un controlador le pide a un servicio que genere información, éste la genera y retorna, y el controlador la retorna a su vez a quien haya realizado la Request.

¿Qué está pasando?

Pero toda la información que mostramos en el navegador debe ser generada por un servicio del Back End?

No, hay partes que se administran desde el Front End. Y a esas partes se las denomina estáticas, en contraposición a las generadas desde el Back End, que son siempre dinámicas.

Esto no significa que las “estáticas” no provean interactividad, al contrario, a los ojos del usuario, esta es la única parte interactiva de toda la página.

¿Cómo logramos entonces que el servidor nos muestre esta parte “estática” de nuestra página?

Servir archivos estáticos

En consola ejecutar:

```
npm i --save @nestjs/serve-static
```

Agregamos un paquete npm que nos permite mostrar estos archivos

En app.module.ts agregar:

-en cabecera

```
import { ServeStaticModule } from '@nestjs/serve-static';
import { join } from 'path';
```

Lo importamos, junto con otro que necesitamos para indicar dónde estarán nuestros archivos

-seccion imports

```
[
    ServeStaticModule.forRoot({ rootPath: join(__dirname, '..', 'client') }),
],
```

Aqui configuramos la ruta y forma de acceso a la carpeta 'client' que es donde estarán nuestros archivos estáticos

https://docs.nestjs.com/recipes/serve-static

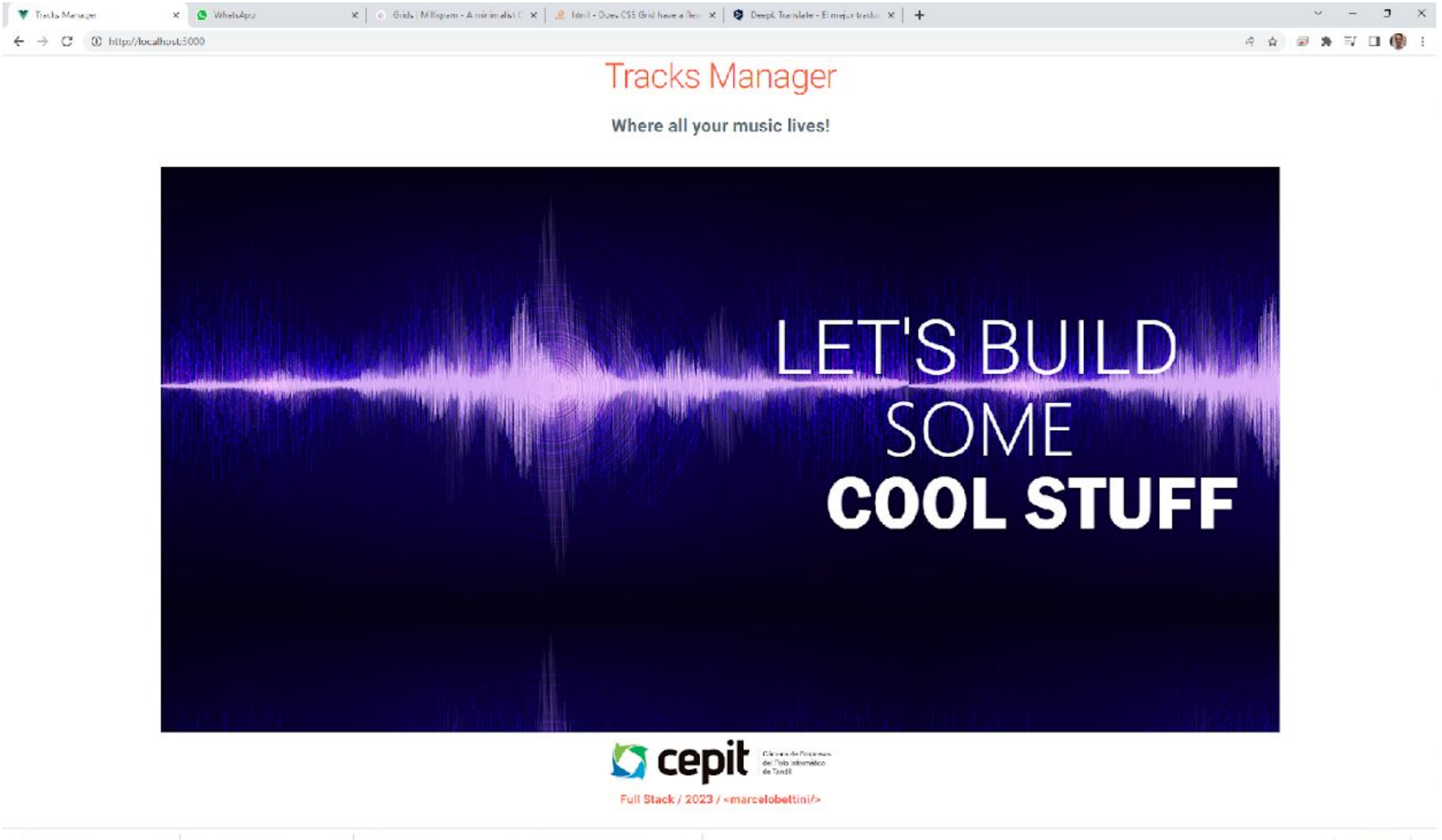
Crear el sitio estático

Crear los archivos estáticos en la carpeta “client”, en el mismo nivel que la carpeta “src”
//no adentro, ojo 🐵

Crear:

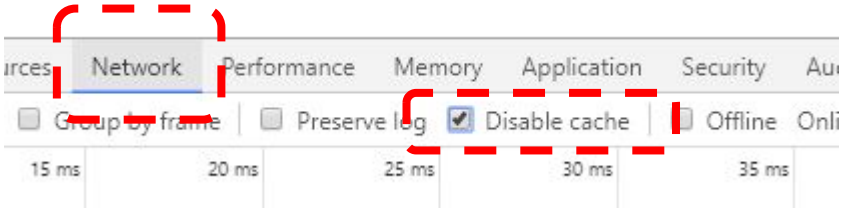
- un index.html
- una hoja de estilos en client/css/styles.css

Ruta raíz con archivo estático



Cache y Refresco

- A partir de ahora nuestra página web va a estar en un servidor.
- El navegador puede usar caché para acelerar la navegación, mostrándonos a veces una versión vieja de la página (o de algunos archivos CSS, JS, …)
- Usar siempre Ctrl+F5 para actualizar los archivos para forzar a borrar el cache o con el inspector deshabilitar el cache



Back End

Carrera Programador full-stack

Práctica:
Crear un sitio web

Crear un sitio web con dos o tres documentos HTML. Deben estar conectados por un sistema de navegación (HTML nativo).