

Ejercicio - Despliegue de Aplicación Web Fullstack en VPS

Tarea realizada por Javier Uría

En primer lugar, instalo mysql de forma segura y creo una base de datos llamada test_virtual que usaré para el backend de la aplicación.

Luego, instalo maven "sudo apt-get install maven" ya que con ello me instalará también el jdk (el openjdk-11). A continuación instalo el openjdk-8 "sudo apt-get install openjdk-8-jdk". Una vez instalado, compruebo la versión de java y la de su máquina virtual.

```
master@daw-120:~$ java -version
openjdk version "11.0.13" 2021-10-19
OpenJDK Runtime Environment (build 11.0.13+8-Ubuntu-0ubuntu1.18.04)
OpenJDK 64-Bit Server VM (build 11.0.13+8-Ubuntu-0ubuntu1.18.04, mixed mode)
master@daw-120:~$
```

```
master@daw-120:~$ javac -version
javac 1.8.0_312
master@daw-120:~$
```

El siguiente paso es hacerse con la aplicación. En primer lugar, hago un git clone al repositorio de github del backend de la aplicación en mi equipo real

```

vespertino@DESKTOP-I6A768Q MINGW64 ~
$ pwd
/c/Users/vespertino

vespertino@DESKTOP-I6A768Q MINGW64 ~
$ cd Desktop

vespertino@DESKTOP-I6A768Q MINGW64 ~/Desktop
$ ls
'Balsamiq Wireframes.lnk'*
'Contenido Base Ejercicios (VI).rar'
'Eclipse IDE for Java Developers - 2021-09.lnk'*
'Ejercicio - trabajo con imagenes/'
'Ejercicio Docker - redes/'
'Ejercicio Docker.md'
'Ejercicio Docker.pdf'
'Ejercicio docker-compose.md'
'Ejercicio docker-compose.pdf'
'Ejercicios UT2 (VI).pdf'
JSFWebAppEx.war
MaterialBase/
'Microsoft Edge.lnk'*
'Microsoft Teams.lnk'*
'Visual Studio Code.lnk'*
cavanosa/
chuletasExamen/
credenciales-surfworpress.PNG
desktop.ini
'tarea video.md'
uria_rodriguez_javier_examenEv1/
uria_rodriguez_javier_examenEv1.rar
workspace_Practicas/

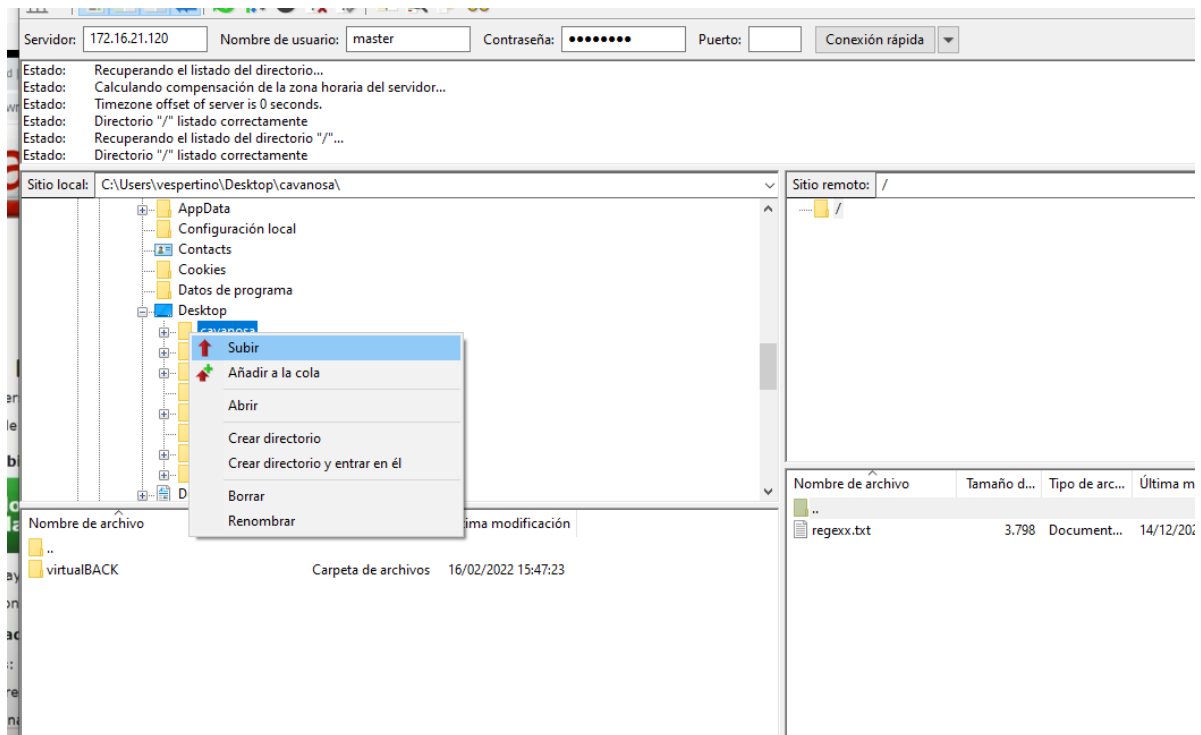
vespertino@DESKTOP-I6A768Q MINGW64 ~/Desktop
$ cd cavanosa

vespertino@DESKTOP-I6A768Q MINGW64 ~/Desktop/cavanosa
$ git clone https://github.com/cavanosa/virtualBACK.git
Cloning into 'virtualBACK'...
remote: Enumerating objects: 44, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 44 (delta 1), reused 44 (delta 1), pack-reused 0
Receiving objects: 100% (44/44), 59.90 KiB | 1.22 MiB/s, done.
Resolving deltas: 100% (1/1), done.

vespertino@DESKTOP-I6A768Q MINGW64 ~/Desktop/cavanosa
$ |

```

Una vez tengo la carpeta del back, abro el filezilla para subirla al servidor, que ya tiene instalado y configurado el vsftpd para poder realizar la transferencia



Nombre de archivo	Tamaño d...	Tipo de arc...	Última modificaci...	Permisos	Propietario/Grupo
..					
cavanosa		Carpeta de...			
regexx.txt	3.798	Document...	14/12/2021 15:31:00	-rw-r--r--	1000 1000

```
master@daw-120: ~/cavanosa
```

```
master@daw-120:~$ pwd
/home/master
master@daw-120:~$ ls
cavanosa  regexx.txt
master@daw-120:~$ cd cavanosa
master@daw-120:~/cavanosa$ ls
virtualBACK
master@daw-120:~/cavanosa$
```

Una vez tengo el proyecto, genero el .jar mediante el comando "mvn install", que me genera una nueva carpeta llamada target

```
master@daw-120:~/cavanosa/virtualBACK$ ls
HELP.md  mvnw  mvnw.cmd  pom.xml  src  target
master@daw-120:~/cavanosa/virtualBACK$
```

Donde tengo el .jar generado

```
master@daw-120:~/cavanosa/virtualBACK/target$ ls
classes          maven-archiver  test-classes
generated-sources  maven-status    virtual-0.0.1-SNAPSHOT.jar
generated-test-sources  surefire-reports  virtual.jar
master@daw-120:~/cavanosa/virtualBACK/target$
```

Ahora, pruebo a ejecutar dicho .jar

```

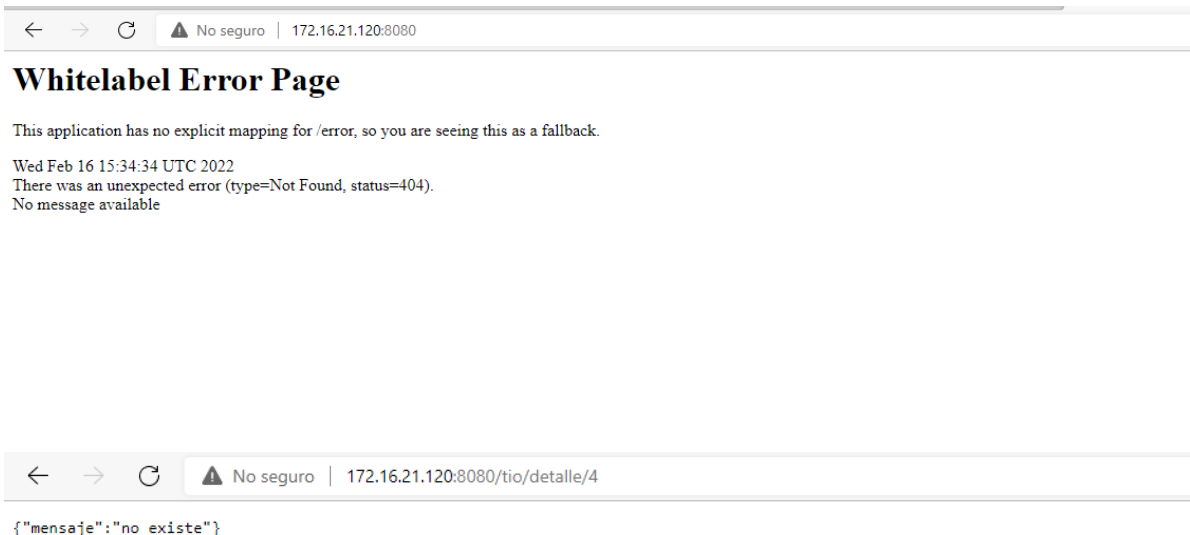
master@daw-120:~/cavanosa/virtualBACK/target$ cd ..
master@daw-120:~/cavanosa/virtualBACK$ ls
HELP.md mvnw mvnw.cmd pom.xml src target
master@daw-120:~/cavanosa/virtualBACK$ java -jar target/virtual.jar

  ____ _
 / ___ \| | | |
/ /___ \| |_| |
\___\___\__\__|_|_|_|
:: Spring Boot :: (v2.2.1.RELEASE)

2022-02-16 15:12:23.235 INFO 13590 --- [main] com.cavanosa.virtual.V
irtualApplication : Starting VirtualApplication v0.0.1-SNAPSHOT on daw-120 with
PID 13590 (/home/master/cavanosa/virtualBACK/target/virtual.jar started by mast
er in /home/master/cavanosa/virtualBACK)
2022-02-16 15:12:23.237 INFO 13590 --- [main] com.cavanosa.virtual.V
irtualApplication : No active profile set, falling back to default profiles: de
fault
2022-02-16 15:12:24.364 INFO 13590 --- [main] .s.d.r.c.RepositoryCon
figurationDelegate : Bootstrapping Spring Data repositories in DEFAULT mode.
2022-02-16 15:12:24.454 INFO 13590 --- [main] .s.d.r.c.RepositoryCon
figurationDelegate : Finished Spring Data repository scanning in 82ms. Found 1 r
epository interfaces.
2022-02-16 15:12:25.011 INFO 13590 --- [main] trationDelegate$BeanPo
stProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTrans
actionManagementConfiguration' of type [org.springframework.transaction.annotati
on.ProxyTransactionManagementConfiguration] is not eligible for getting processe
d by all BeanPostProcessors (for example: not eligible for auto-proxying)
2022-02-16 15:12:25.466 INFO 13590 --- [main] o.s.b.w.embedded.tomca
t.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-02-16 15:12:25.490 INFO 13590 --- [main] o.apache.catalina.core
.StandardService : Starting service [Tomcat]
2022-02-16 15:12:25.491 INFO 13590 --- [main] org.apache.catalina.co
re.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.27]

```

Con ello, tendré la aplicación desplegada ejecutándose en el puerto 8080. Para comprobarlo, accederé desde el equipo real a la ip del servidor con dicho puerto (cabe recordar que puede ser necesario abrir dicho puerto en el cortafuegos -ufw allow-)



Y vemos que la aplicación está desplegada

Sabiendo que la aplicación funciona correctamente, lo que vamos a hacer ahora es convertirlo en ejecutable, de modo que nada más arrancar el sistema se ejecute el jar para no tener que hacerlo a mano. Para ello, primero hay que crear el servicio

```
master@daw-120: ~/cavanosa/virtualBACK
GNU nano 2.9.3 /etc/systemd/system/spring.service

[Unit]
Description=my spring boot app

[Service]
Restart=always
ExecStart=/home/master/cavanosa/virtualBACK/target/virtual.jar
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

Y luego, habilitarlo

```
master@daw-120: ~/cavanosa/virtualBACK
master@daw-120:~/cavanosa/virtualBACK$ sudo systemctl enable spring.service
Created symlink /etc/systemd/system/multi-user.target.wants/spring.service -> /e
tc/systemd/system/spring.service.
master@daw-120:~/cavanosa/virtualBACK$
```

Y por ultimo, iniciarlo y comprobar que esté activo

```
master@daw-120: ~/cavanosa/virtualBACK
master@daw-120:~/cavanosa/virtualBACK$ sudo systemctl enable spring.service
Created symlink /etc/systemd/system/multi-user.target.wants/spring.service -> /e
tc/systemd/system/spring.service.
master@daw-120:~/cavanosa/virtualBACK$ sudo systemctl is-enabled spring.service
enabled
master@daw-120:~/cavanosa/virtualBACK$ sudo systemctl is-active spring.service
inactive
master@daw-120:~/cavanosa/virtualBACK$ sudo systemctl start spring.service
master@daw-120:~/cavanosa/virtualBACK$ sudo systemctl is-active spring.service
active
master@daw-120:~/cavanosa/virtualBACK$
```

Y con esto, tendremos la aplicación corriendo siempre que el servicio esté activo. Con esto, ya estaría configurado spring correctamente. Ahora toca la parte del frontend.

El primer paso sería instalar apache. Luego, al igual que hicimos con el back de la aplicación, clonamos el front al equipo local y, por filezilla, lo subimos al vps. No obstante, antes de subirlo, habrá que realizar unos cambios al proyecto a través del visual studio.

En primer lugar abrimos la terminal y ejecutamos un npm update para crear unos node modules que faltan.

```
PS C:\Users\vespertino\Desktop\chuletasExamen\virtualFRONT-master> npm update
```

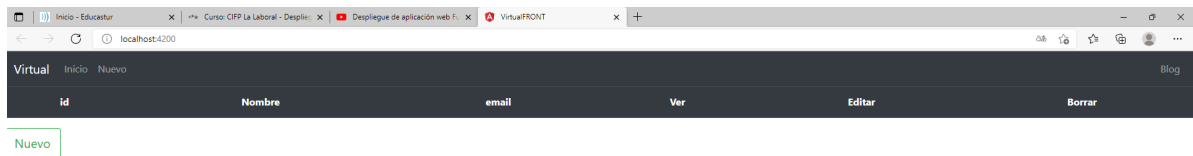
Luego ejecutamos un `ng serve -o` para que nos abra el navegador con `localhost:4200`

```
PS C:\Users\vespertino\Desktop\chuletasExamen\virtualFRONT-master> npm run ng serve -o
```

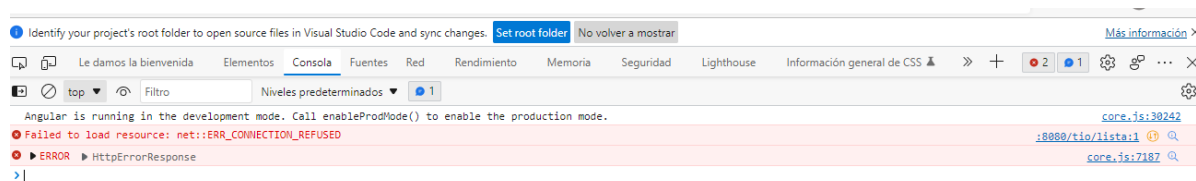
```
> virtual-front@0.0.0 ng C:\Users\vespertino\Desktop\chuletasExamen\virtualFRONT-master
> ng "serve"
```

```
Browserslist: caniuse-lite is outdated. Please run next command `npm update`
(node:5516) Warning: Accessing non-existent property 'cat' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)
(node:5516) Warning: Accessing non-existent property 'cd' of module exports inside circular dependency
(node:5516) Warning: Accessing non-existent property 'chmod' of module exports inside circular dependency
(node:5516) Warning: Accessing non-existent property 'cp' of module exports inside circular dependency
(node:5516) Warning: Accessing non-existent property 'dirs' of module exports inside circular dependency
```

Y veo que tengo la aplicación corriendo



Aunque todavía presenta un error, y es que no encuentra un recurso en `localhost:8080`



Esto es porque necesito cambiar la dirección que usa para acceder a ese recurso a la del VPS

```

VIRTUALFRONT-MASTER
├── e2e
├── node_modules
├── src
│   ├── app
│   │   ├── menu
│   │   ├── model
│   │   └── tio
│   │       ├── # actualizar-tio.component.css
│   │       ├── <> actualizar-tio.component.html
│   │       ├── TS actualizar-tio.component.ts
│   │       ├── # detalle-tio.component.css
│   │       ├── <> detalle-tio.component.html
│   │       ├── TS detalle-tio.component.ts
│   │       ├── # lista-tio.component.css
│   │       ├── <> lista-tio.component.html
│   │       ├── TS lista-tio.component.ts
│   │       ├── # nuevo-tio.component.css
│   │       ├── <> nuevo-tio.component.html
│   │       ├── TS nuevo-tio.component.ts
│   │       └── TS tio.service.ts
└── ...

src > app > tio > TS tio.service.ts > TioService > lista
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Observable } from 'rxjs';
4  import { Tio } from '../model/tio';
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class TioService {
10
11    tioURL = 'http://localhost:8080/tio/';
12
13    constructor(private httpClient: HttpClient) { }
14
15    lista(): Observable<Tio[]> {
16      return this.httpClient.get<Tio[]>(this.tioURL + 'li:
17    }
18
19    detalle(id: number): Observable<Tio> {
20      return this.httpClient.get<Tio>(this.tioURL + `deta:
21    }
22
23    nuevo(tio: Tio): Observable<any> {
24      return this.httpClient.post<any>(this.tioURL + 'nuev:

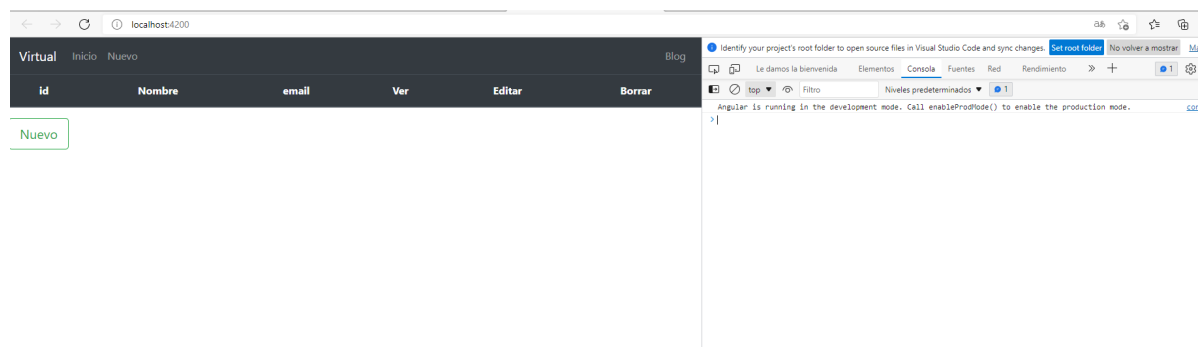
```

```

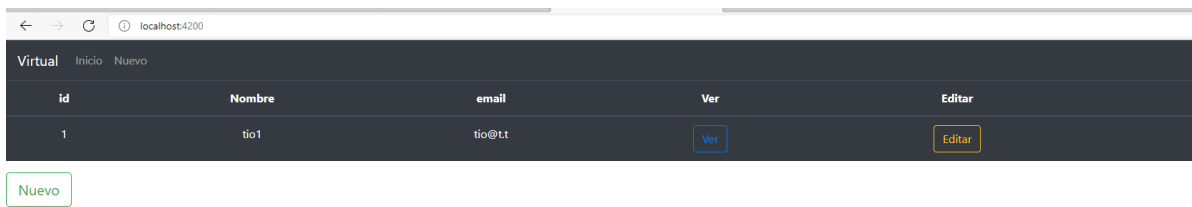
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class TioService {
10
11    tioURL = 'http://172.16.21.120:8080/tio/';
12
13    constructor(private httpClient: HttpClient) { }
14
15    lista(): Observable<Tio[]> {
16      return this.httpClient.get<Tio[]>(this.tioURL + 'list:

```

Y ahora ya puedo ver que el error ha desaparecido



Y, si pruebo por ejemplo a crear un nuevo registro, funciona correctamente

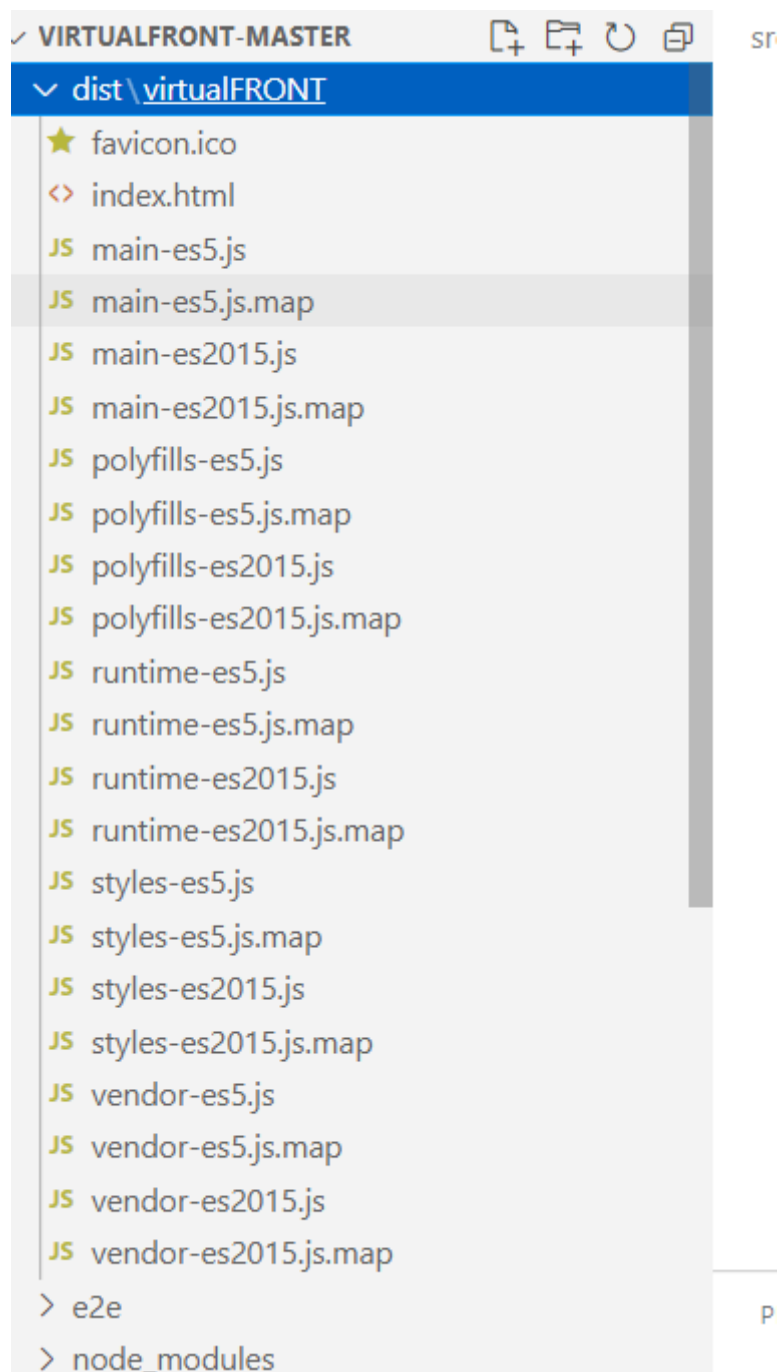


Ahora que sé que este proyecto, el front, funciona correctamente, voy a subirlo a producción. Para ello, abro una terminal nueva y ejecuto el `ng build --prod`

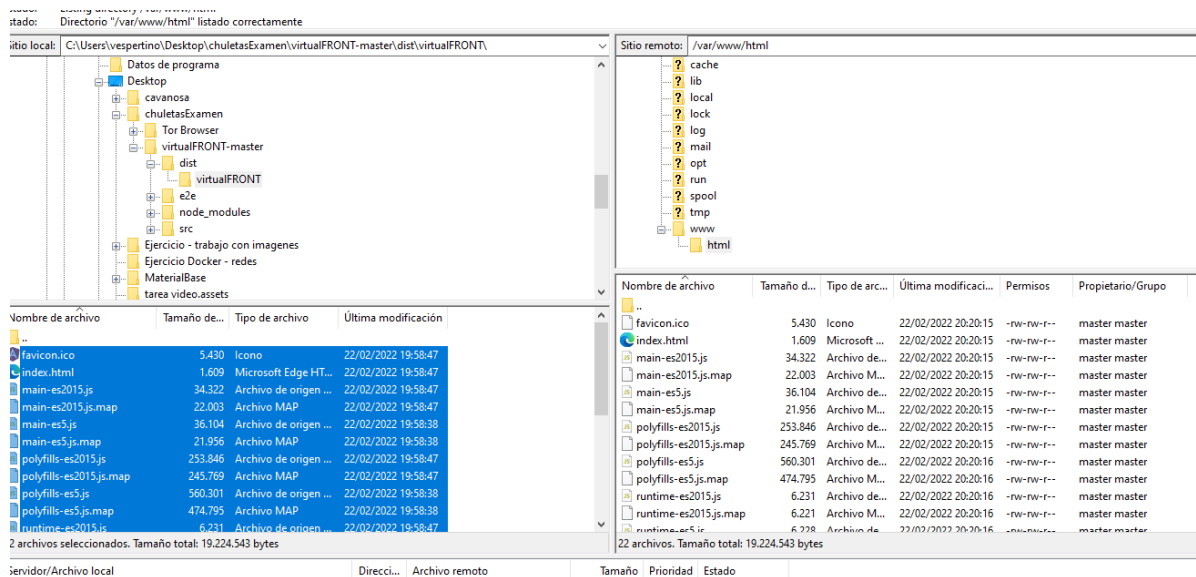
```
PS C:\Users\vespertino\Desktop\chuletasExamen\virtualFRONT-master> npm run ng build --prod
> virtual-front@0.0.0 ng C:\Users\vespertino\Desktop\chuletasExamen\virtualFRONT-master
> ng "build"

Browserslist: caniuse-lite is outdated. Please run next command `npm update`
10% building 3/3 modules 0 active(node:2868) Warning: Accessing non-existent property 'cat' of module exports inside circular dependency
(node:2868) Warning: Accessing non-existent property 'cd' of module exports inside circular dependency
(node:2868) Warning: Accessing non-existent property 'chmod' of module exports inside circular dependency
(node:2868) Warning: Accessing non-existent property 'cp' of module exports inside circular dependency
(node:2868) Warning: Accessing non-existent property 'dirs' of module exports inside circular dependency
```

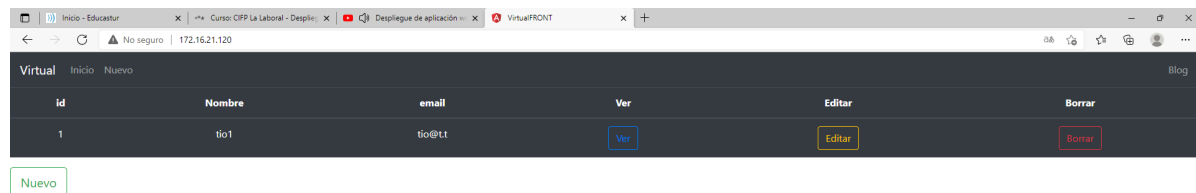
Y con ello, nos genera una carpeta llamada `dist`, que a su vez tiene una carpeta con el nombre del proyecto, que es la que, ya sí, queremos subir al VPS junto con el back.



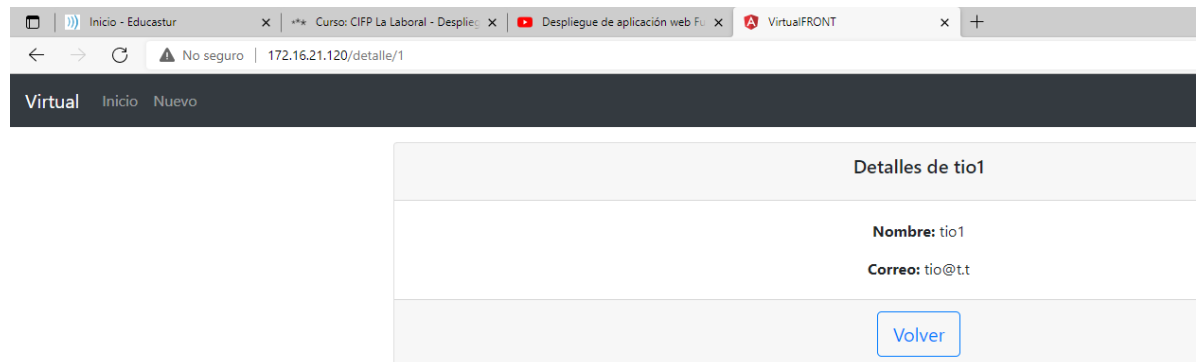
Esta vez, la carpeta (o mejor dicho, sus contenidos) la meteremos en /var/www/html/ en vez de donde teníamos la del back.



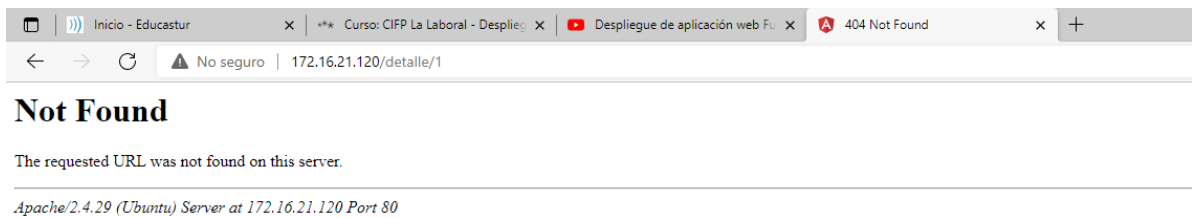
Y ya tendré el front de la aplicación desplegado en el VPS



Y voy probando las distintas páginas del sitio, viendo que funcionan todas (ej: detalle de un tio)



No obstante, todavía hay un fallo. La aplicación solo tiene un html (el index) ya que es una Single Page Application y el resto de páginas que vemos realmente no existen, por lo que, si recargásemos la página estando en una de estas páginas "inexistentes", nos daría un 404.



Para corregir esto, tenemos que volver al proyecto en el visual studio, y en un archivo llamado `app-routing.module.ts`, añadimos lo siguiente

```
};

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

Ahora, toca volver a producir la carpeta de `dist` y volver a subir su contenido al VPS. Con esto, el problema de recargar la páginas que no son `index.html` queda resuelto.

Y habiendo llegado hasta aquí, ya tenemos desplegada la aplicación al completo (back y front) en el VPS, completamente funcional.