UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de una aplicación web con información sobre el Covid a partir de datos abiertos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autores: Carlos Arauz García

Tutor: Manuela Albert Albiol, María Victoria Torres Bosch

Curso 2020-2021

Resumen

El Trabajo de Fin de Grado consistirá en la realización de una aplicación web que recuperará datos abiertos de COVID de distintas entidades públicas periódicamente (por ejemplo, casos por cada 100.000 habitantes, casos activos, casos que se han recuperado, etc.) y los mostrará en distintos formatos (tabla, gráfico, etc) según el usuario lo requiera. La aplicación se desarrollará de forma que la interfaz se adapte a distintos tamaños de dispositivos y se utilizarán funcionalidades que permitirán cambiar algunas características de la interfaz (como un modo nocturno para evitar el cansancio visual). Además durante el desarrollo del proyecto se harán uso de buenas prácticas tanto de codificación (por ejemplo, separación de contenido y diseño) como de organización de proyecto (por ejemplo, uso de control de versiones estructuradas). La implementación de la aplicación utilizará tecnologías web como Angular, Javascript, Typescript, SCSS y Bootstrap.

Palabras clave: Angular, Covid, TFG, Javascript, Typescript, SCSS, Bootstrap

Abstract

This work will consist on making a web application that will get COVID open data from several public repositories (such as, active infections, vaccines received, etc) and it will show them on different ways, as for example, tables, graphics and charts. The application will be responsive, so it will adapt to smartphones, tablets, laptops and desktop computers.

Furthermore, there will be added some features such as dark mode(in order to avoid eye weariness).

In addition, during the course of the development, we'll be doing use of good practices, in terms of design and structuration, and we will use control version too.

The implementation will use tools such as Angular, Jquery, Typescript, SCSS, Bootstrap

Keywords : Angular, Covid, TFG, Javascript, Typescript, SCSS, Bootstrap

Tabla de contenidos

Índice de ilustraciones

1. Introducción

1.1 Motivación

El Covid-19 es una realidad que nos atañe a todos, y al ser un peligro tan grande para nuestra sociedad, lo mejor es tomar precauciones y estar bien informado.

Por otra parte, el mundo del desarrollo es un mundo en constante cambio, y cada vez hay más tecnologías y herramientas para desarrollar un producto cada vez más completo y funcional.

Dentro del sector del desarrollo web, la parte de frameworks frontend es una parte muy interesante, puesto que está constantemente en desarrollo, cada día es más flexible y ofrece una gran cantidad de maneras para realizar una misma tarea, puesto que implementa un gran número de herramientas y librerías que complementan y mejoran mucho la calidad y las opciones que tiene un desarrollador a la hora de hacer una web.

Como bien sabemos, una web es algo que utilizamos a diario. Al haber visitado tanta cantidad de webs a lo largo de nuestra vida, se han podido ir observando los defectos que tiene cada web. Como desarrolladores, cuando vemos un defecto, se piensa en una solución.

A raíz de estas primeras premisas nace mi motivación para realizar este trabajo de fin de grado, que consistirá en una aplicación web realizada con el framework Angular que integre diferentes servicios y herramientas, y los muestre de una forma muy clara, concisa y útil.

1.2 Objetivos

El objetivo de esta web es claro:

Mostrar datos de covid de la comunidad valenciana: globales, por localidades, por rangos de edad, etc, de una manera clara para cualquier tipo de usuario, tenga experiencia visitando webs o no.

En concreto, los objetivos de datos a mostrar son:

- Mostrar los datos de vacunación de la Comunidad Valenciana, tales como gente vacunada parcialmente o con las dos dosis, tanto en dato numérico como en porcentaje.
- Mostrar el porcentaje de vacunados por rango de edad, especificando también si han sido vacunados parcialmente o completamente.
- Mostrar la cantidad de dosis que se han puesto de cada marca, con número y porcentaje.
- Mostrar los datos de casos positivos de PCR y otros como casos positivos acumulados los últimos 14 días, defunciones...etc.
- Mostrar los datos de contagio en la comunidad por género y ordenados por intervalos de edad.
- Mostrar tabla que muestre los datos de contagios y fallecidos por cada municipio de la Comunidad Valenciana.

Todas estas representaciones de datos deben de ser en un formato adecuado (por ejemplo, es más correcto poner unos datos sobre porcentaje de vacunados por rangos de edad en una gráfica de barras que en un mapa de arbol, puesto que hay que tener en cuenta las dos variables y se tienen que ver y entender fácilmente los datos).

Asimismo, la web deberá de tener una estructura clara en la que el usuario sepa en todo momento dónde se encuentra, y se podrá adaptar en el modo nocturno o diurno, para la mejor experiencia posible de usuario, debido a que a mucha gente le resulta molesto para la vista el modo diurno, y usan el nocturno.

1.3 Impacto Esperado

Me gustaría que los usuarios de esta aplicación puedan consultar los datos sin complicaciones. Que simplemente con un vistazo, sepan qué datos están viendo y qué significan, sin importar el dispositivo desde el que se esté visualizando ni la experiencia visitando webs del usuario que lo esté visitando.

1.4 Metodología

La metodología empleada en este proyecto abarca las siguientes fases:

1. Análisis y selección de datos: Analizar los distintos tipos de datos que nos daban las APIs y seleccionar aquellos que mejor se consideraron.
2. Diseño de interfaz: Hacer los primeros bocetos para ver qué estructura va a tener el proyecto.
3. Creación del proyecto de angular: En esta fase se crea el proyecto de angular, su estructura y las rutas.
4. Creación de interfaz: Desarrollo de los bocetos de la segunda fase, sobre el proyecto de angular creado en la tercera fase.
5. Conseguir los datos de las APIs. En esta fase conseguiremos rescatar y dejar los datos en el formato que deseemos para más tarde, mostrarlos donde deseemos.
6. Integrar datos en la aplicación: Ahora, conectaremos los datos extraídos en el paso anterior.
7. Optimización de estilos: Se mejorarán los estilos de la página.
8. Depuración: Por último, se buscan los errores que pueda dar la página y se depurarán.

1.5 Estructura

Este proyecto se divide en los siguientes apartados:

1. Introducción

   En este apartado comentaremos el motivo del proyecto, sus objetivos, las motivaciones que llevaron a realizarlo e introduciremos brevemente su finalidad.

2. Estado del arte.

   Análisis de la situación actual de webs similares a la que vamos a realizar, comparativa y diferencias con la que vamos a desarrollar.

3. Análisis del problema

   Analizaremos la tarea a realizar, para resaltar los puntos críticos e identificar mejor el orden en el que realizar las cosas y la estructura a realizar.

4. Diseño de la solución

   Basándonos en el apartado anterior, diseñaremos la aplicación acorde a los requisitos previos, tanto a nivel de lógica como visual.

5. Desarrollo de la solución propuesta

   Desarrollaremos la aplicación a partir del análisis y diseño realizados.

6. Implantación

   Se mostrarán los resultados que vaya dando el proceso de desarrollo de nuestra aplicación web.

7. Conclusiones

   Se hará una valoración del proyecto y se comentará su relación con algunas asignaturas del grado.

8. Trabajos futuros

   En este apartado se comentarán apartados que no se hayan podido realizar por algún motivo y que en el futuro se podrían implementar.

9. Anexo

   En este apartado estarán aquellos procesos de instalación que no tenían cabida en otros apartados porque no eran tan relevantes, pero que aún así son importantes.

10. Referencias

    Libros y enlaces de webs de donde he obtenido información que he usado en este proyecto.

2. Estado del arte

2.1 Crítica al estado del arte

En la situación en la que nos encontramos, puede resultar bastante impactante que en ciertas comunidades autónomas no haya la misma facilidad de acceso a datos del Covid-19 que en otras.

Cuando se buscan datos del Covid-19 por repositorios globales del estado, se puede apreciar claramente la disparidad entre diferentes Comunidades. Mientras hay comunidades que muestran los datos con una fecha, bien ordenados y en varios formatos para que sea más flexible a la hora de tratar datos, hay otras comunidades que muestran datos muy escasos, sin fecha y en un sólo formato (y en algunos casos, mal formateado).

A continuación, se revisarán dos ejemplos sobre datos de Covid-19 de dos comunidades autónomas diferentes(Murcia y Comunidad Valenciana), y uno de vacunación a nivel nacional.

2.1.1 Murciasalud

La página es la siguiente:

https://www.murciasalud.es/pagina.php?id=458869&idsec=6575

Figura 1: Opciones Murciasalud.es

En esta página, se proporciona la información, diaria y semanal, por zonas de Murcia y con datos tanto en PDF como en CSV.

Con esta página se puede comprobar rápidamente que es fácil orientarse y llegar al resultado que se está buscando. La información está bien separada por tipo de información, por localidades y por fecha. A continuación, veremos un ejemplo:

Si hacemos click en “Situación actual” podremos observar la siguiente pantalla:

Figura 2: Datos Murciasalud

En esta pantalla, se pueden ver los datos de la región de Murcia. Como se puede apreciar, están actualizados, y están mostrados de un modo muy simple y entendible para cualquier tipo de usuario.

No es exactamente el método de representación de datos que se busca para este proyecto, pero se acerca bastante en cuanto a la claridad a la hora de mostrar la información.

2.1.2 Coronavirus.san.gva

En cambio, en la de la Comunidad Valenciana, cuyo enlace es https://coronavirus.san.gva.es/es/estadisticas , podemos observar claramente el problema:

Figura 3: Datos coronavirus.san.gva

Podemos observar que en un espacio muy reducido se muestra gran cantidad de datos, lo cual no es práctico para prácticamente ningún tipo de usuario.

Los números se ven muy pequeños, no se aprecian bien en algunas localizaciones, y en las gráficas se solapan los números y resultados.

Probablemente se estén mostrando los mismos datos que en la página de Murcia, o incluso más, pero visualmente es excesivo cuando el objetivo del usuario es informarse de manera rápida y precisa.

2.1.3 Covid-vacuna app

Una página que muestra con bastante acierto la idea que persigue este proyecto es la siguiente:

https://covid-vacuna.app/

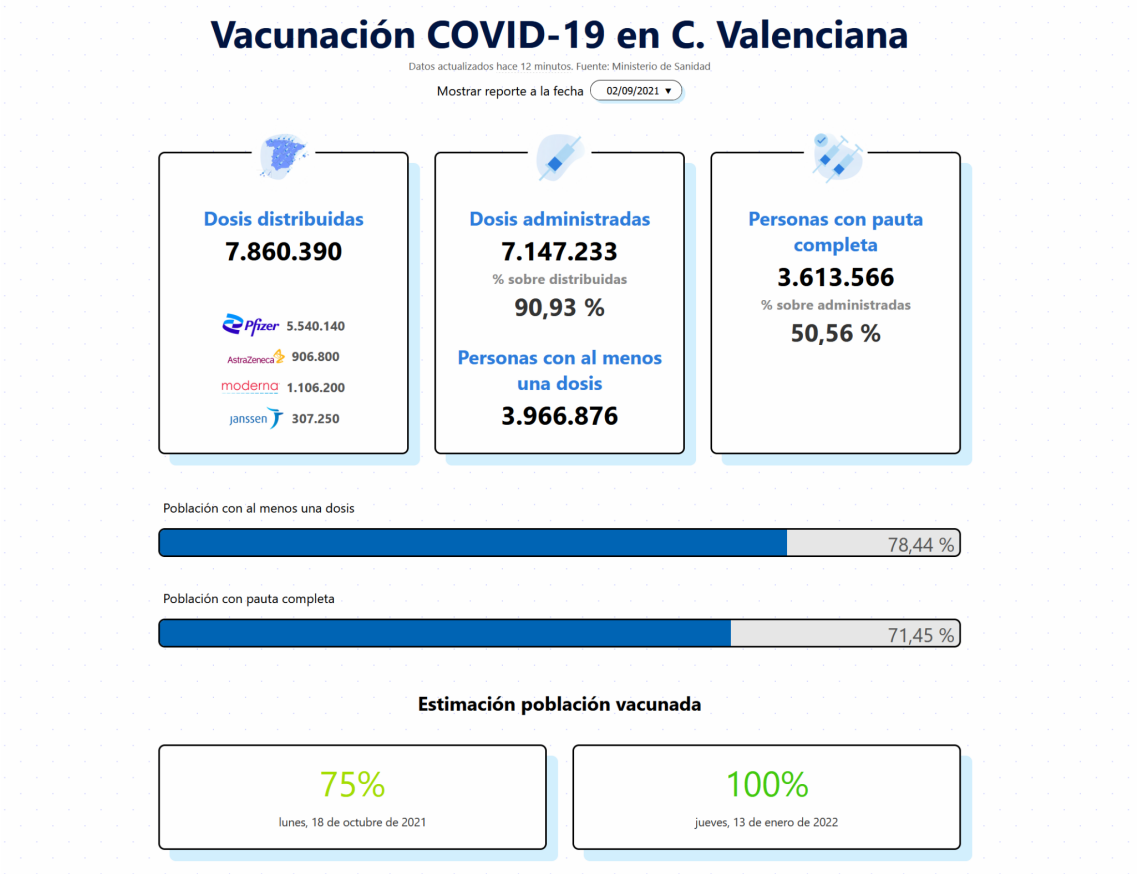


Figura 4: Datos Covid-Vacuna

Como podemos observar, en esta página se muestran los datos de un modo muy fácil de entender, y ese suele ser un factor bastante determinante para que la use cualquier tipo de usuario.

En este caso, la aplicación es sólamente de datos de vacunación, y los ordena por marca, por porcentaje de dosis administradas sobre las distribuidas, por gente vacunada con una o dos dosis, etc.

2.2 Propuesta

¿Qué se busca lograr en esta web que la haga necesaria? ¿En qué se diferenciará exactamente?

Como se ha podido observar en las anteriores páginas, en dos de ellas se mostraban datos de infectados y en la otra de vacunas.

La idea detrás de este proyecto consiste en unir esos dos tipos de datos en una misma web y desarrollarla para que sea una web de una sóla página mediante la cual se podrán ver ver más tipos de datos pero conservando la delicadeza de una web simple. Como hemos podido apreciar en la web de la Comunidad Valenciana, demasiada complejidad no tiene por qué ser buena.

Como conclusión de la propuesta, se puede decir que lo que se busca es realizar una aplicación web de angular que sobre la misma página, muestre el tipo de dato requerido, sin tener que recargar cada vez toda la web entera.

3. Análisis del problema

3.1 Identificación de problemas

Durante el desarrollo de esta aplicación, se va a requerir instalar, aparte de angular:

- Librerías para gráficas: Se requerirá del uso de librerías externas a angular para representar los datos deseados.
- Librerías para tablas: Se requerirá el uso de librerías externas para representar los datos en tablas.
- Varios mecanismos de recuperación de datos de las APIs. Puesto que se va a tratar con diferentes APIs de diferentes fuentes, se deberá encontrar un modo correcto para conseguir mostrar esos datos en el sitio deseado.

3.2 Solución propuesta

Después de revisar varias opciones, se ha optado por las librerías NG2-Charts, NGX-Charts, PrimeNG y Angular Material para la representación de gráficos, tablas y uso de estilos. Se usarán a conveniencia según el tipo de datos que se vayan a utilizar.

Para la recuperación de APIs, se usarán peticiones http o fetch, y en caso de ser necesario, un backend de node con express.

3.3 Plan de trabajo

La parte más primordial en este proyecto son los datos.

El plan de trabajo consistirá pues en recuperar los datos lo antes posible y crear las estructuras de datos donde vayamos a incluirlos. Después, habrá que integrar esos datos en las gráficas o tablas que les correspondan, y más adelante añadir funcionalidades extras y estilos.

En resumen, se trabajará por nivel de importancia, de más a menos.

Realmente, sabemos que todo es importante, pero resulta más crítico que una web no tenga datos o que no funcione correctamente, que una web que lo que le falte sea pulir los estilos.

4. Diseño de la solución

4.1 Arquitectura del sistema

La arquitectura del sistema contará con los siguientes apartados:

- La sección “shared”: En esta sección se colocarán aquellos componentes que se van a utilizar globalmente en la aplicación. En este caso, contamos con el navbar y el footer.
- La sección “gráficas”: Esta sección estará compuesta por:
    - Components: Comprenderá todos los métodos de representación de datos, tales como tablas y gráficas.
    - Interfaces: Contendrá aquellas interfaces que se deseen utilizar para tratar un tipo de dato definido por el desarrollador en esa interfaz.
    - Pages: Comprenderá las 3 páginas principales a mostrar. Estas páginas son componentes pero se deben tener separados para tener una buena estructuración del proyecto.
    - Services: Es el apartado dedicado a la conexión de datos y de tratado de las APIs. En este apartado se definen las variables que contienen las URLs de las APIs, y se crean métodos con las llamadas a estas APIs, para más tarde ser consultados por algún componente que requiera esos datos.

4.2 Diseño Detallado

4.2.1 Módulo App.Routing

Para que el usuario se pueda mover entre las distintas páginas, se deben configurar las rutas para que cuando se desee cambiar entre vacunación, infectados y datos por municipio, cambie el contenido central.

Esta conexión se realiza mediante el módulo de app-routing, en el que se tendrán que escribir las rutas de las ventanas que deseemos que la aplicación web muestre.

En concreto, se debe importar RouterModule de angular, y escribir el siguiente código:

```
const routes: Routes = [
  {
    path: '',
    component: VacunacionComponent,
    pathMatch: 'full'
  },
  {
    path: 'infectados',
    component: InfectadosComponent
  },
  {
    path: 'datosMunicipio',
    component: DatosMunicipioComponent
  },
  {
    path: '**',
    redirectTo: ''
  }
];
```

Figura 5: Configuración de Rutas

- En el caso del path vacío, será la ventana principal de la web, que se mostrará cuando se cargue la web con la URL base.

- En el caso del path “**”, es la encargada de redirigirnos a otra ruta en caso de no encontrar la ruta indicada. En este ejemplo, redirigirá a la web a la ventana de Vacunados, que es la principal.
- Por último, se debe importar en app.module el módulo de AppRouting, para que detecte las direcciones correctamente.

  Si se realizan estos pasos correctamente, cuando se coloque el router-outlet en el html, se obtendrá un resultado como el siguiente:

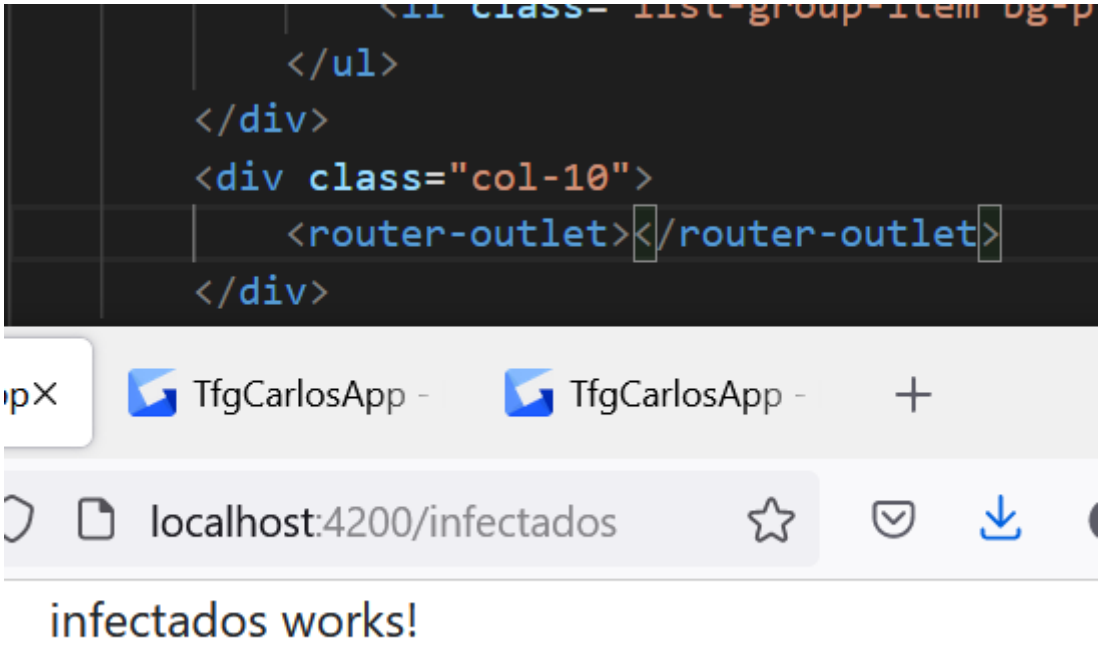


Figura 6: Funcionamiento de Rutas

Para configurar correctamente el navbar y sus rutas, hay que añadir la clase routerLink con la dirección a la que apunta. Si se realiza correctamente, al efectuar un click sobre las opciones del navbar, se debería mostrar por pantalla la sección deseada.

```
<a class="nav-item nav-link " routerLink="" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}">Vacunación <span class="sr-only"></span></a>
<a class="nav-item nav-link" routerLink="infectados" routerLinkActive="active">Infectados</a>
<a class="nav-item nav-link" routerLink="fallecimientos" routerLinkActive="active">Casos por Municipios</a>
```

Figura 7 : Rutas en en navbar

A continuación, necesitaremos el apartado de páginas, donde deberán ir esas 3 páginas de infectados, vacunación y fallecimientos.

Las páginas son componentes, pero hay que estructurarlo de este modo para no confundir con otros componentes y llevar una buena organización de los componentes de la web.

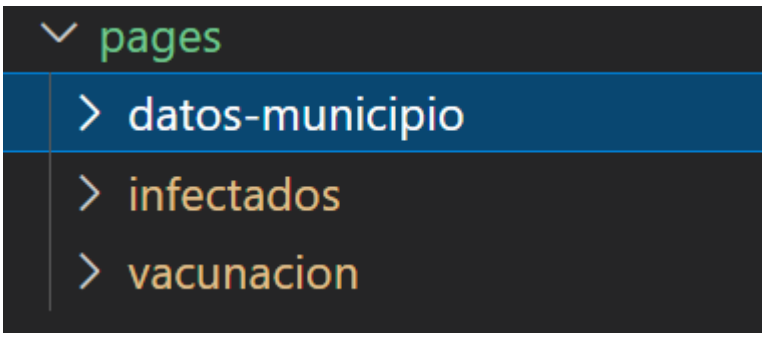


Figura 8 : Estructura de páginas

Cada página se compondrá de un HTML, SCSS Y TS, como cualquier componente.

Dentro de cada una de estas páginas, se llamará a los componentes que requiramos, usando el correspondiente selector, como veremos a continuación.

:El modo en el que se debe de implementar es el siguiente:

- Se debe buscar qué tipo de representación de datos se desea mostrar. En esta explicación se mostrará como ejemplo el desarrollo de una gráfica de tipo Gauge.
- Se debe crear un componente con el nombre del tipo de representación que se vaya a utilizar(en este ejemplo, gauge-chart).

- Accediendo a la documentación oficial, se obtendrá el código HTML y TS y se deberá copiar a los ficheros correspondientes del componente recién creado.
- Cambiar las variables de ese componente como se desee, e indicarle cómo se va a llamar la variable de la que coja los datos.
- En el HTML, en [results], se le asignará el nombre deseado a esa variable.
- Acceder al documento padre de esta gráfica, e indicarle el nombre de esta variable del siguiente modo.

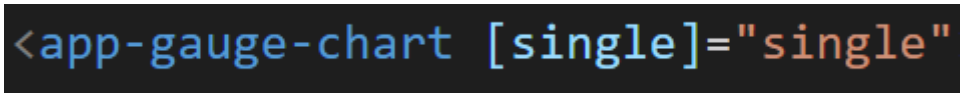


Figura 9: Argumento de Charts

- Por último, en el documento Typescript de la gráfica gauge, se creará un Input para recopilar esos datos,El procedimiento a seguir es muy similar para el resto de gráficas de la librería NGX, por lo que no explicaré el resto, ya que siguiendo los mismos pasos y depurando algún posible error, funcionan del mismo modo.

4.2.2 Módulo Shared

El módulo shared se compone de aquellos componentes que se vayan a usar en las diferentes páginas, como podrían ser un footer, un header o un sidebar.

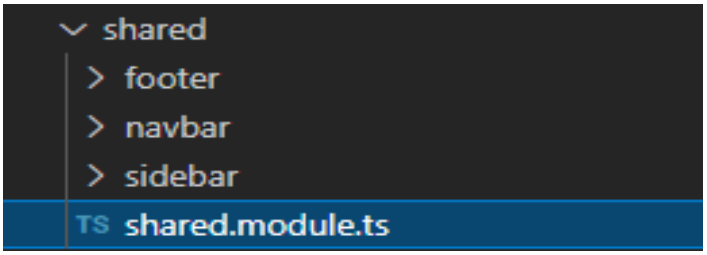


Figura 10: Módulo Shared

En este proyecto,el módulo shared va a estar compuesto por los componentes:

Navbar: será responsive, y seré el encargado de y cuando estemos en un dispositivo móvil, añadirá un botón que controlará el burger menú.

Será el encargado de hacer que el usuario se desplace por las 3 páginas principales.

Esta tarea se conseguirá mediante

Footer: Contendrá información de contacto sobre el desarrollador de esta web.

Para usar estos componentes fuera de este módulo, deberán ser añadidos como declarations y exports en shared.module, como se muestra en la siguiente figura:

Figura 11: Declarations y exports en Shared.Module

4.3 Idea y planificación

En este apartado, se explicarán los pasos a seguir a nivel teórico y el código escrito para conseguir ese objetivo. No se encontrarán aquí procesos de instalación de librerías ni de otras dependencias, eso se comentará en el apartado “Anexos”.

Para desarrollar una aplicación web, lo primero que se debe tener es una idea de la organización que se le desea dar, ya que facilita mucho la parte inicial de desarrollo de la aplicación.

En este caso, durante el proceso de pensar en la interfaz y en los primeros mockups, se identificó que una forma correcta podría ser la siguiente:

- Colocación de un selector de zonas(ya fuera un navbar o un sidebar), para poder cambiar entre las distintas zonas.
- Colocación de una parte central con el contenido.
- Información general y compartida en el pie de página

El primer mockup que se realizó de la parte principal con la barra lateral y la cabecera fue éste, para intentar darle forma a esas ideas.

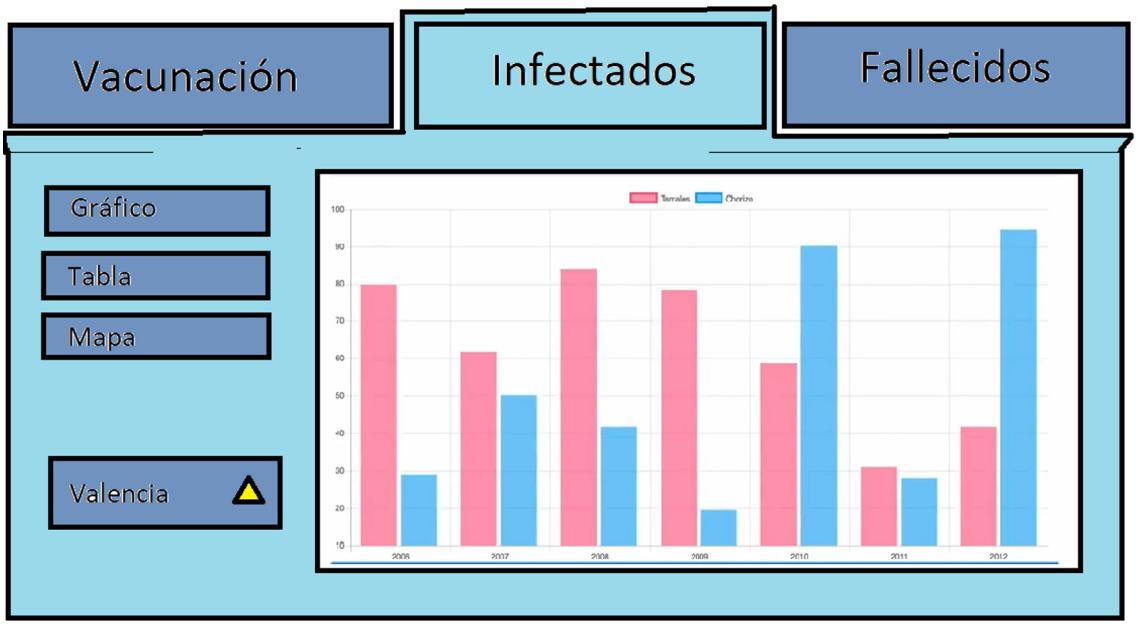


Figura 12: Boceto de diseño

También se realizó un mockup para la versión móvil, que contaría con un menú estilo burger que contendría los apartados del Header, y el resto de apartados, al ser responsive, se adaptarán al tamaño de móvil.

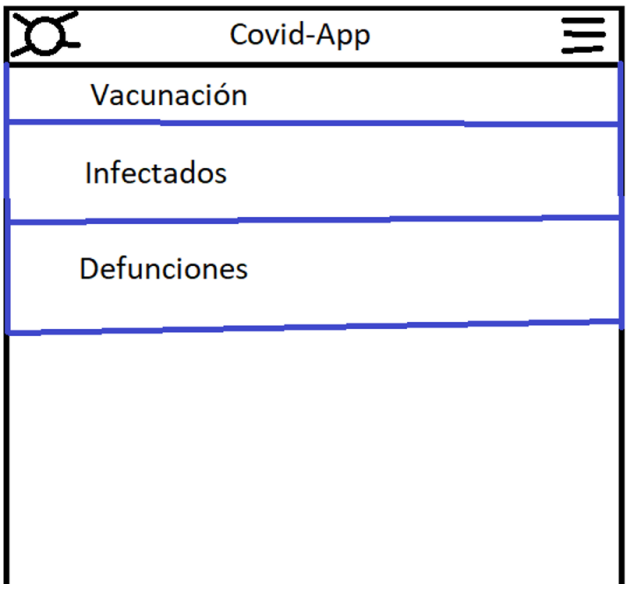


Figura 13: Boceto de diseño móvil

Con este diseño ya se podía empezar a realizar una aplicación. La idea inicial era tener esos 3 apartados (Vacunación, Infectados y Fallecidos), y las 3 opciones de representación(Gráfico, Tabla y Mapa), y que se pudiera mostrar para Valencia, Castellón o Alicante.

5. Desarrollo de la solución propuesta

5.1 Tecnologías utilizadas

5.1.1 Angular

Angular es un entorno de trabajo(framework) de código abierto para la realización de aplicaciones webs y móviles.

5.1.1.1 Historia

Angular.js o Angular 1 fue lanzado inicialmente el 20 de octubre de 2010. El 15 de septiembre de 2016, se lanzó Angular 2, y actualmente estamos por Angular 12.1.4.

Respecto a las versiones se debe hacer una breve aclaración: Angular 1 y Angular 2 no son compatibles. De Angular 2 en adelante, sí que lo son.

El motivo por el que Angular 1 no es compatible con Angular 2 es porque no es una simple actualización con mejoras, es una reinvención de Angular desde sus cimientos, por lo que fue una migración prácticamente absoluta.

Angular 1 utilizaba Javascript, y en cambio, Angular 2 utiliza Typescript, que utiliza un tipado más fuerte que Javascript. Una de las grandes desventajas era que Angular 1 no estaba creado para el desarrollo móvil, y contaba con un rendimiento bastante por debajo del de Angular 2.

En general, Angular 1 era menos organizado que Angular 2, y por eso mismo Angular 2 lo sustituyó.

De aquí en adelante cuando se mencione a Angular, no nos referiremos a su primera versión, sino que haremos referencia a Angular 2, que actualmente se le conoce como Angular.

5.1.1.2 Fortalezas

Angular es un framework que se basa en componentes.
Los componentes son clases usadas para hacer las uniones pertinentes de datos.

Una de sus mayores ventajas es que es un framework constantemente actualizado. Recibe una nueva versión cada 6 meses, y está en constante evolución.

Angular cuenta con una gran flexibilidad y permite crear webs mediante el uso de los estándares webs más avanzados. También tiene soporte en todos los navegadores modernos, y como ya comenté, está pensado también para el desarrollo móvil.

Como se ha comentado previamente, Angular utiliza Typescript, que es una mejora de javascript que cuenta con mayor seguridad y las tareas de depuración son mucho más sencillas con él, ya que te avisa del error que estás cometiendo e incluso , en ocasiones, tiene “quick fixes” para arreglarlo, pero el simple hecho de que avise al desarrollador de cuál es el error, facilita el reconocimiento del error y el hecho de poder arreglarlo. Aún así, en caso de no poder, como tendremos un código de error, podemos buscar en algún foro en el que la duda ya esté realizada y resuelta, por lo que el desarrollador se ahorrará bastante tiempo.

5.1.1.3 Estructura

Angular basa su estructura en módulos, componentes, interfaces y servicios, aunque no es lo único que se puede utilizar.

A continuación, se explicarça brevemente para que tengamos una idea clara y estructurada de qué es y cómo funciona Angular. Más adelante, durante el proceso de creación del proyecto y del uso de cada una de estas partes de la estructura, se explicará más detalladamente su uso.

- ➔ Los módulos son cada uno de los “bloques” de los que se compone angular. Cada aplicación web tendrá como mínimo un módulo que se generará automáticamente al crear nuestra aplicación web de angular, llamado app.module.ts.

Dentro de estos módulos, tendremos los siguientes apartados :

- Importaciones, de donde se traen otros módulos o servicios que han sido declarados en otro módulo y necesitemos usar en el módulo actual.
- Declaraciones: Se usan para declarar componentes, directivas y pipes del módulo en el que estemos situados y también consigue que estén disponibles para otros módulos
- Exportaciones: Consigue que los componentes, directivas y tuberías estén disponibles en módulos en los que se invoquen mediante import.
- Providers: Es una instrucción para obtener un valor de una dependencia. Normalmente son servicios que se tengan que inplementar en un componente.
- Bootstrap: Sólo está presente en el módulo raíz.

La apariencia de un módulo es la siguiente:

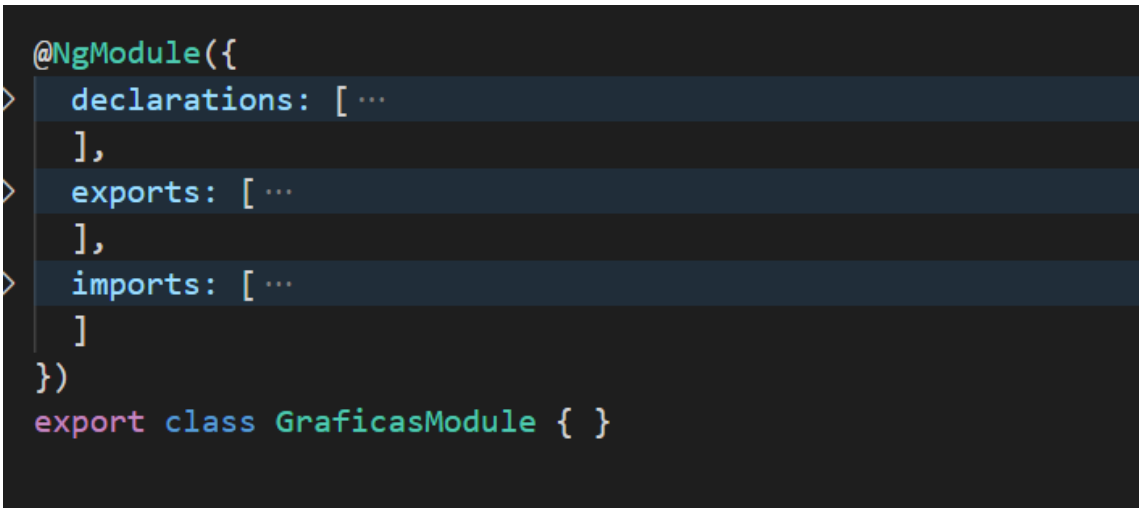


Figura 14: Estructura módulo

Como podemos observar, aquí sólo encontramos declarations, exports e imports, pero no usamos providers ni bootstrap. Sólo se utilizará lo necesario para nuestra aplicación, puesto que cada apartado tiene su función propia.

➔ Los componentes, por otra parte, son cada una de las partes (visualmente hablando) de las que se compone nuestra aplicación. Veámoslo más cómodamente con un ejemplo visual:

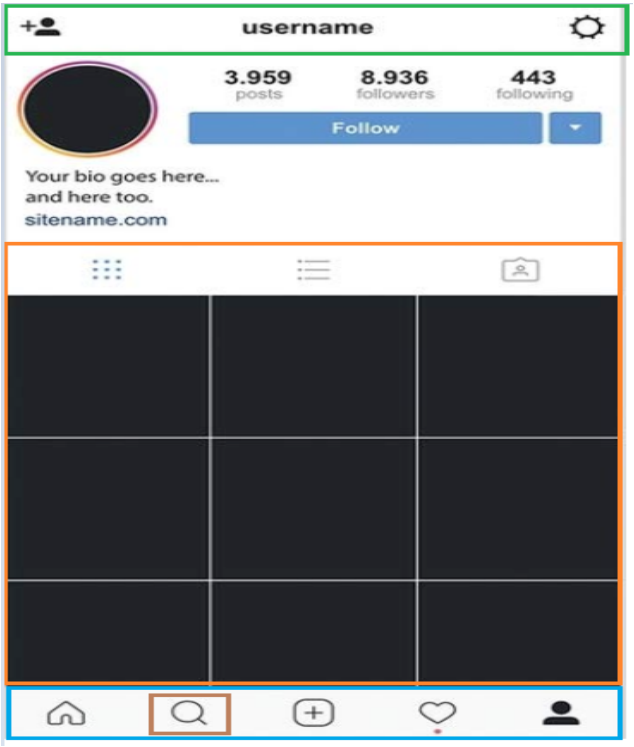


Figura 15: Distribución de componentes

Como observamos en esta imagen de ejemplo, podemos diferenciar varios componentes. Se nombrarán varios para que quede claro el ejemplo, pero la elección de componentes la hace el desarrollador, por lo que dos desarrolladores distintos pueden separarlo de manera diferente.

A continuación, analizaremos un par de ejemplos de la foto:

Primeramente, encontramos una cabecera(marcada en verde) donde tenemos nuestro nombre de usuario, un icono de herramientas y otro icono. Por otra parte, en el pie de página vemos un componente footer(marcado en azul). Dentro de este componente distinguimos un botón(señalado en marrón) que también podría ser un componente. Y

por último, en el centro se encuentra situada la parte principal de las fotos del perfil(marcado en naranja).

Como observamos, cada componente se puede diferenciar a simple vista del resto, pero se puede hacer con distintas organizaciones. Un mal diseño, por ejemplo, sería poner todo en un componente. Un buen diseño, en cambio, separará óptimamente por componentes, basándose en las necesidades de cada página.

Por lo general, si hay una parte de la aplicación que vas a reutilizar, tiene que tener su propio componente, por lo que será mucho más sencilla su organización, y a la hora de depurar queda todo mucho más claro.

- ➔ Interfaz: se usa para definir el tipo de los objetos que vas a usar. Por ejemplo, si nos encontramos en una aplicación que te pide los datos personales, una interfaz podría contener los campos:
  - -nombre: string
  - -primerApellido: string;
  - -segundoApellido: string;
  - -edad: number;
  - -dni: string

  Por lo que, una vez esa interfaz está creada, si a un objeto se le asigna el tipo de esta interfaz, la interfaz detecta automáticamente que va a necesitar esos elementos. Por lo que en caso de tener que depurar errores, Typescript nos facilitará mucho el proceso.

- ➔ Servicios: Es la capa que nos entrega los datos se quieran usar en nuestra aplicación. En resumen, es nuestro proveedor de datos. Los servicios son consumidos por los componentes, que usan la información que ellos les proporcionan.

5.1.2 Typescript

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft. Fue lanzado el 1 de octubre de 2012 y ha ido mejorando desde entonces. Es la evolución de Javascript, que como se comentó previamente en la introducción, mejora muchos errores que tenía Javascript. Cabe destacar que las aplicaciones web realizadas en Javascript también son válidas en Typescript.

TypeScript cuenta con un tipado de datos fuerte y estático. En Javascript, se puede definir una variable con var y no indicar el tipo, y no nos pediría nada. Incluso podríamos sobreescribir un tipo de datos con otro, y no obtendríamos ningún error por parte de javascript. En TypeScript tenemos restricciones que ayudan a la buena organización de código. Por ejemplo, al definir una variable de tipo String, esa variable va a aceptar sólamente String.

Este tipo de control del tipo de datos y el control de errores, tiene grandes ventajas, como la sugerencia de autocompletar el código con lo que detecta que, según su lógica interna, debería ir en un lugar determinado, o como la sugerencia de soluciones en los errores.

En resumen, es la evolución natural de Javascript, y cada vez lo usan más desarrolladores y frameworks. Para este proyecto, TypeScript viene nativo con Angular, por lo que no vamos a tener que hacer nada para instalarlo.

5.1.3 HTML5

HTML5 es la quinta versión de HTML, que es el lenguaje de marcas en el que se basan las páginas web. Se puede decir que HTML es el esqueleto de una página web. Esta versión es el nuevo estándar de desarrollo web, que incorpora muchas mejoras, tales como:

- La capacidad nativa de insertar archivos multimedia a la web sin necesidad de utilizar plugins.
- Mejora el rendimiento
- Permite almacenar muchos más datos en el navegador. Mientras que con HTML4 podíamos guardar información en muy pocos kilobytes, ahora podemos guardar información en hasta 10 megabytes de datos.

5.1.4 CSS3

CSS3 es la última versión del lenguaje de estilos CSS que complementa al lenguaje de marcas HTML5. En estos documentos, se definen las propiedades que queremos que tengan los elementos que hemos creado en el HTML. Se pueden aplicar por los siguientes modos:

- En un documento CSS externo.
- Aplicar directamente en la línea, añadiendo la etiqueta “style”.
- Aplicar incrustado en el documento HTML.

En cualquiera de estos tipos, podemos clasificar el modo de aplicar estilos en:

- Clases, que vienen precedidas por un punto(.ejemplo) y se pueden usar en distintos elementos.
- Id's, que vienen precedidos por un # (#ejemplo),
- Directamente sobre un tipo de elemento, por ejemplo img.

5.1.5 SCSS

SCSS es una adaptación de CSS, pero que permite organizar mejor el código. Mientras que en CSS solíamos tener un CSS por página, en este caso nos permitirá dividir esos estilos en partes mucho más pequeñas, para una mejor organización.
Por ejemplo: Una página que contenga un componente header, uno main y uno footer, tendrá 3 archivos SCSS, uno para cada uno. E incluso si el componente main está compuesto por más componentes, cada uno podrá tener su propio archivo SCSS.

Esta función busca exactamente lo mismo que Angular con su estructura: Facilitar y modularizar los componentes de una aplicación.

Aparte de estas funcionalidades, SCSS permite la anidación de selectores.

Por ejemplo: En una web compleja, resultará un proceso innecesariamente complejo si para darle estilos a un elemento concreto, tenemos que hacer una ruta larga.

En el caso de SCSS, se nos permite anidar uno dentro de otro. A continuación, veremos un ejemplo para entenderlo mejor:

Figura 16: Comparativa SCSS y CSS

Como se puede observar en esta imagen, con una ruta sólo de 3 elementos(que pueden ser muchos más), en el documento CSS tenemos que crear una regla para el

“section”, otra para el elemento con clase “class-one” dentro de section, y otro más de un elemento de clase “button” dentro de “class-one” dentro de “section”.

En el caso de SCSS, sólo es necesario crear una regla para section.

Dentro de esa regla, podemos crear la otra regla de “class-one” solamente, con sus reglas, y dentro de la misma, la clase “button”.

Como podemos observar, es bastante parecida a la estructuración por tabulaciones de un HTML.

Aparte de contar con estas funcionalidades, también podemos estructurar los documentos.

En el caso de tener el componente main de nuestro documento 3 componentes, podemos dejar el componente del main sin reglas, y que ese documento sólo importe a los otros 3. De tal modo, si se desea buscar algo en el main, al distinguir los componentes en los que está dividido, se puede buscar más fácilmente aquello que se desee modificar.

Los archivos SCSS también se pueden compilar en un archivo final CSS, que llevará toda la información a la web en un mismo documento.

5.1.6 Bootstrap

Bootstrap es una biblioteca de herramientas de CSS de código abierto para desarrollo web. Éste facilita mucho la creación de una web y los estilos de la misma, puesto que combina elementos CSS y JavaScript de modo que ahorra mucho trabajo al desarrollador que lo esté usando.

Su principal objetivo es que la aplicación web que está siendo creada sea responsive, de modo que se pueda utilizar la aplicación web en un ordenador, en una tablet o en un móvil.

Bootstrap está basado en un sistema GRID de 12 columnas, mediante las cuales organiza el contenido de la web, dándole las columnas necesarias a cada elemento.

Además, bootstrap cuenta con muchos componentes predefinidos como navbars, alertas, botones y otros.

5.1.7 Visual Studio Code

La aplicación se realizará en el programa Visual Studio Code, que es un editor de código fuente lanzado por Microsoft el 29 de abril de 2015 que ha ido ganando popularidad hasta ser uno de los editores más utilizados.

Es un editor multiplataforma que se mantiene actualizado constantemente, además de contar con una gran cantidad de lenguajes de programación.

Cuenta nativamente con la utilidad de remarcar el texto según la sintaxis del lenguaje que se esté usando, y cuenta con una gran cantidad de plugins para hacer más ágil y claro el proceso de escribir código.

5.1.8 Jquery

Jquery es una biblioteca de Javascript multiplataforma de código abierto que permite funciones como:

- Manipular el DOM
- Manejar eventos
- Implementar animaciones
- Desarrollar aplicaciones AJAX

Nació para complementar los defectos de javascript, por lo que Jquery nos brinda una API de funciones que permite hacer el uso de Javascript mucho más sencillo y ágil. Además, cuenta también con plugins para extender sus utilidades.

Actualmente el 78% de los sitios de internet utilizan jQuery. Y en los sitios web que usan Javascript, el 95% usan Jquery, por lo que tiene bastante peso actualmente.

5.1.9 Github

Github es una plataforma gratuita en la que se pueden crear repositorios públicos o privados, en los que se pueden compartir proyectos para trabajar en grupo en el proyecto.

Las fortalezas principales de github son las siguientes:

- La posibilidad de dividir el proyecto en ramas, mediante las cuales se pueden realizar distintas tareas al mismo tiempo sin que causen problemas entre ellas.
- Cuenta con control de versiones: Con github se puede llevar un control sobre el código con cada actualización de código que se realice, para comprobar los cambios e incluso revertirlos en caso de ser erróneos.
- Sobre el código, se puede añadir anotaciones a modo de revisión para que el creador del código u otro desarrollador que esté en el proyecto puedan revisarlas.

La página oficial de GitHub ofrece toda la información necesaria para una organización correcta de las ramas y todos los comandos necesarios para usarlo.

Actualmente, los IDE y los editores de código suelen integrar total o parcialmente github para hacer esta labor más liviana.

5.1.10 Postman

Postman es una herramienta gratuita disponible en Windows, Linux y OSX que permite crear peticiones a APIs para rescatar los datos que necesitemos, y la posibilidad de agruparlas por colecciones para tener un mejor control de las APIs.

Una de sus mayores fortalezas es poder crear entornos de trabajo en los que definir variables, y por ende, ganar una gran organización para poder tener todas las peticiones en una misma aplicación correctamente organizadas.

Otra ventaja es que se puede exportar a formato .JSON toda la configuración, por lo que se puede usar esa misma configuración en diferentes equipos, compartirla con otros compañeros o llevar un control de versiones en GitHub.

5.1.11 NGX-Charts

Es la librería de gráficas para Angular que se ha decidido usar en este proyecto.

Sus gráficas son bastante flexibles en personalización y en su página se pueden encontrar ejemplos de muchos tipos de representación de datos, por lo que resulta ideal para este proyecto en el que se va a necesitar distintos tipos de representación.

5.1.12 Angular Material

Angular Material es una librería diseñada específicamente para angular que cuenta con muchos componentes configurados para la creación de nuestra página. En este caso, se va a hacer uso la tabla para representar los datos de infectados por municipio.

El siguiente paso a realizar será subir la base de este proyecto a github, para controlar las versiones a partir de este momento.

5.2 Uso de Github

El siguiente paso a realizar es subir el proyecto a github e ir actualizándolo para llevar un buen control de versiones. Para lograr esto, se deberán seguir los siguientes pasos:

- Crear un proyecto nuevo en github .
- Tener una cuenta activa en github en caso de no disponer de uno.
- Configurar nuestro usuario y nuestro email .
- Clonar el repositorio que hemos creado en el equipo de trabajo.
- Hacer un commit con todos los cambios.
- Hacer un push para subirlo a github.

Si se han seguido estos pasos, ya se podrá empezar a usar github para controlar las versiones, y para poder revertir algún fallo de la página web.

5.3 Uso de Librerías

Es uno de los puntos delicados del proyecto, dado que es lo que le va a entrar por la vista al cliente al entrar en la página.

Es posible que los datos que queramos mostrar puedan haber sido obtenidos de un modo muy complejo, pero la gran mayoría de la gente no sabe cómo funciona por dentro ese tratado de datos, por lo que hay que elegir el método correcto de representación de datos.

5.3.1 NGX-Charts

Inicialmente, se iba a utilizar la librería NG2-Charts, pero al final se acabó eligiendo otra, llamada NGX-Chart, que resultó ser más flexible y más agradable visualmente.

A continuación, veremos un par de ejemplos de tipos de representación de datos de esta librería, sacados de su página web.

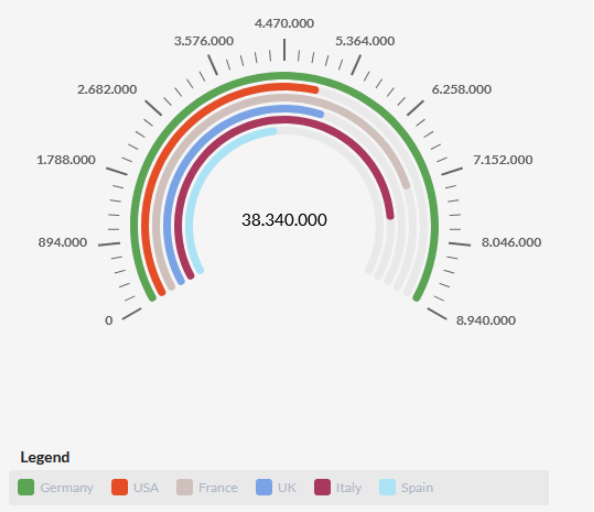


Figura 17: Ejemplo Gauge-Chart

Después de situar las primeras gráficas, el siguiente paso consiste en mostrar los datos correctos, no los de prueba.

Para la realización de esta tarea, teníamos que utilizar ya nuestras APIs.

El modo en el que se debe de implementar es el siguiente:

- Se debe buscar qué tipo de representación de datos se desea mostrar. En esta explicación se mostrará como ejemplo el desarrollo de una gráfica de tipo Gauge. (ver figura 17).
- Se debe crear un componente con el nombre del tipo de representación que se vaya a utilizar(en este ejemplo, gauge-chart).
- Accediendo a la documentación oficial, se obtendrá el código HTML y TS y se deberá copiar a los ficheros correspondientes del componente recién creado.
- Cambiar las variables de ese componente como se desee, e indicarle cómo se va a llamar la variable de la que coja los datos.
- En el HTML, en [results], se le asignará el nombre deseado a esa variable.
- Acceder al documento padre de esta gráfica, e indicarle el nombre de esta variable del siguiente modo.

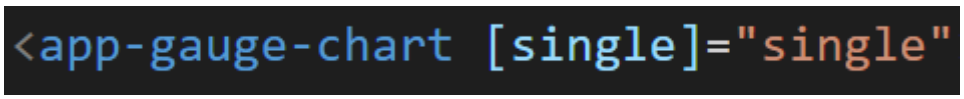


Figura 18: Argumento de Charts

- Por último, en el documento Typescript de la gráfica gauge, se creará un Input para recopilar esos datos,El procedimiento a seguir es muy similar para el resto de gráficas de la librería NGX, por lo que no explicaré el resto, ya que siguiendo los mismos pasos y depurando algún posible error, funcionan del mismo modo.

Al acabar de realizar estos pasos, nos encontraremos con la gráfica rellena con los datos que le asignamos.

Para que quede más claro, veremos un ejemplo de una gráfica de barras de esta librería:

```
<div #containerRef class="card-body">
    <ngx-charts-bar-horizontal-2d
      [view]="[containerRef.offsetWidth, 800]"
      [scheme]="colorScheme"
      [results]="generoEdad"
      [gradient]="gradient"
      [xAxis]="showXAxis"
      [yAxis]="showYAxis"
      [legend]="showLegend"
      [showXAxisLabel]="showXAxisLabel"
      [showYAxisLabel]="showYAxisLabel"
      [xAxisLabel]="xAxisLabel"
      [yAxisLabel]="yAxisLabel"
      (select)="onSelect($event)"
      (activate)="onActivate($event)"
      (deactivate)="onDeactivate($event)"
      >
    </ngx-charts-bar-horizontal-2d>
</div>
```

Figura 19: Estructura Interna NGX-Charts

Como se puede observar en la figura, se ha insertado la gráfica dentro de un div a modo de contenedor para poder aplicarle mejor los estilos deseados.

Se pueden modificar las variables y eventos de gráfica para darle diferente comportamiento.

5.3.2 Angular Material

Se va a utilizar la librería de Angular Material para la realización de una tabla con todo tipo de datos(PCR positivas, PCR positivas en los últimos 14 días, Fallecidos, Tasa de PCR Positivas, etc). La tabla proporcionaría los datos por municipios, por lo que complementará de buena manera la información más global del apartado “infectados”.

5.4 Uso de APIs

Para el tratado de APIs, se va a hacer uso de la herramienta postman, que nos ayudará a hacer las peticiones de datos.

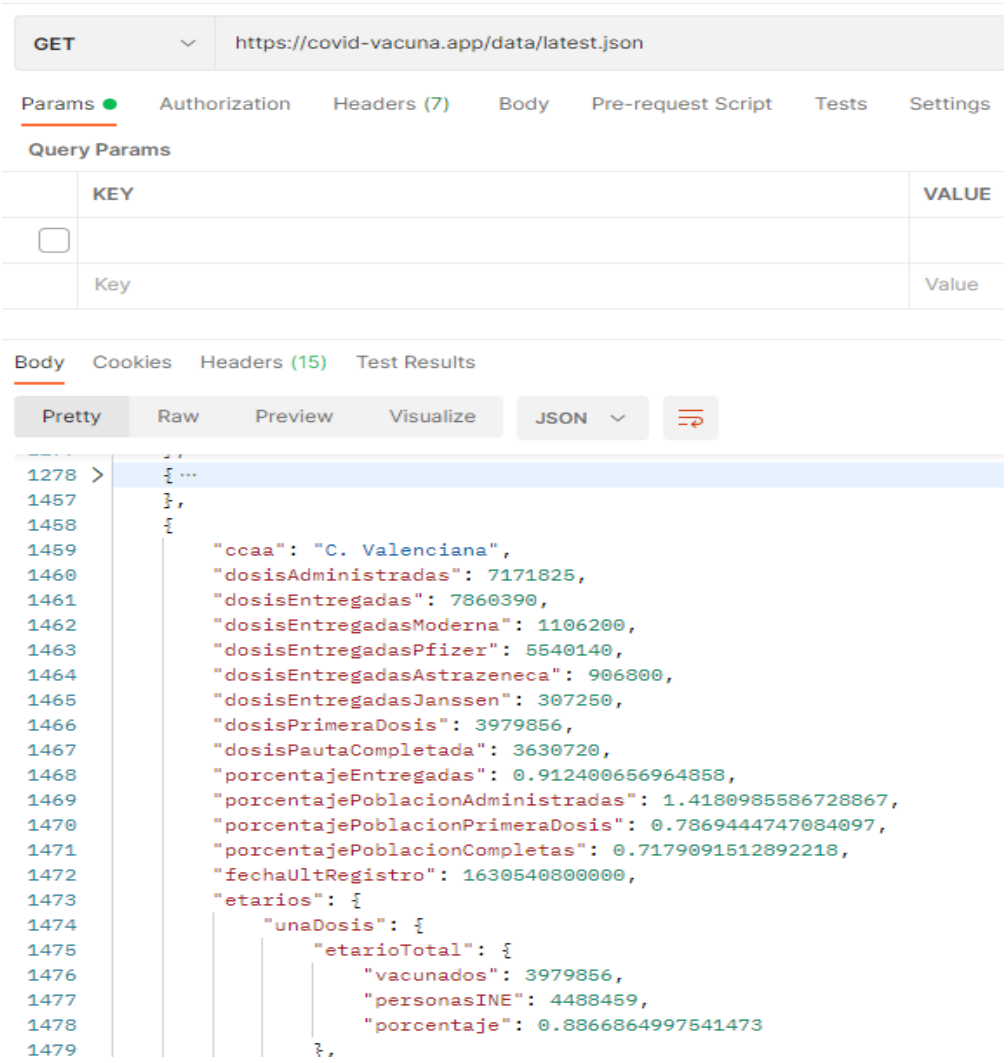


Figura 20 : Api en Postman

Como podemos observar en la figura 19, al escribir una API en su campo de búsqueda, devuelve el resultado de la consulta directa.

Para los datos referentes a la vacunación Covid-19, usaremos la api contenida en el enlace 1 en el apartado de anexos.

Al hacer la consulta, se obtendrá un JSON fácil de manejar, por lo que se tratarán esos datos para mostrarlos donde se requiera.

Para otro tipo de datos, usamos otras APIs, las de los datos del gobierno de España y de la Generalitat Valenciana(enlaces 2 y 3 en anexos). Para éstas, se encontraron más dificultades.

Primero, los métodos de consulta que ellos proporcionaban a modo de “tutorial” devolvían objetos anidados unos dentro de otros, de modo que tratándolo así con postman, se hizo imposible.

Se decidió entonces hacer un servidor de backend en Node para poder tratar mejor esos datos. Una vez se trataron con este servidor los datos, al hacer la consulta en postman a la dirección de ese servidor, y esa consulta devolvió un array con las URLs de los distintos ficheros CSV.

Esto habría sido muy interesante, porque podríamos haber hecho las consultas a cada elemento del array y montar un gráfico por días/semanas/meses. El problema residió en que no venía la fecha en el CSV, por lo que podría mostrar un gráfico aproximado, y no exacto.

Por otra parte, las direcciones no estuvieran ordenadas por fecha, por lo que la dirección válida, no se encontraba en el primer puesto, sino que iba variando según el día.

Al final, se optó por dejar de lado ese servidor backend, a la espera de encontrar otra API mejor(ya que otras APIs similares pero con otro tipo de datos, llevaban la fecha). Así que, por el momento, simplemente se utilizó la dirección al último CSV, por lo que no tendríamos gráficas por tiempo pero tendríamos los datos actualizados a diario, que también es muy práctico.

En el caso de estos CSV de la generalitat, hubo que tratarlos hasta que la consulta los devolviera en un formato con el que nos encontrásemos cómodos.

Veremos un bbreve ejemplo sobre una tabla hecha por género y edad sobre infecciones.

Primeramente, al tratar la API, obtuvimos como respuesta lo siguiente por consola:

Figura 21 : Datos de positivos por Edad y Sexo

Esta figura ya tiene una forma similar a la deseada. En este caso, nos interesaban los casos(en la figura no se ven, pero es la propiedad que hay después de “Percentatgecasos”).

```
searchEdad(){
this.graficasService.getEdadGenero().subscribe(data => {
  data = data.replace(/,/g, '.').replace(/;/g, ',').replace
  var lines=data.split("\n");
  var result = [];
  var headers=lines[0].split(",");


  for(var i=0;i<21;i++){

      var obj:any = {};
      var currentline=lines[i].split(",");

      for(var j=0;j<headers.length;j++){
          obj[headers[j]] = currentline[j];
      }

      result.push(obj);
      if(i >0 && i<11){
          this.mujeres.push(result[i].Casosacumulats);
      }else if(i>10){
          this.hombres.push(result[i].Casosacumulats);
      }

    }
```

Figura 22: Código para tratar CSV

Como se puede observar en esta figura, se hace la llamada a través del servicio gráficasService al método getEdadGenero, que devuelve en el return una petición http de tipo GET a la url de la API.

Se recorrieron estos datos y se montó un array para hombres y otro para mujeres, de los que se obtuvo lo siguiente:

```
hombres: 19188,33796,39869,34672,39541,33678,22896,14244,8187,1831
mujeres: 18720,34432,40416,37324,43038,36637,23254,15465,12129,4667
```

Figura 23: array hombres y mujeres

En este caso, cada array muestra unos números, que son los números de cada grupo de edad.

Y con estos datos, ya lo único que queda es ponerlos en los datos que coge la gráfica para representar

Estas APIs se han consultado mediante una petición HTTP, y en el caso de la JSON ya estaban los datos preparados, pero con el CSV de la otra petición tuve que tratar

los datos para darles la estructura deseada y pasarla a JSON, ya que me parece el formato ideal para rescatar datos.

Estas consultas HTTP a las APIs, se realizan en un servicio de angular. Los servicios son los encargados de hacer estas peticiones, y conectar los datos con las páginas deseadas. Es el encargado de tratar toda esa capa de lógica y de devolver esos datos para ser usados.

El modo de conectar estos datos es mediante los imports, exports y declarations del TypeScript del módulo de gráficas, para conectar las páginas y componentes creados anteriormente entre sí, y con los servicios creados..

Una vez se consiguió este objetivo, el siguiente paso era mostrarlo por pantalla.

Esta tarea se realizó mediante la creación de variables en los ficheros TypeScript de las páginas, y asignándole los datos deseados que se rescataron de las APIs.

Para realizar esta tarea, se ha llamado a las funciones de los servicios que contenían la llamada a la API deseada, y sobre esa función, con los datos devueltos, hemos creado las variables deseadas y se la hemos asignado en los lugares correspondientes a nuestros gráficos.

Para poner un ejemplo más claro, en este último caso la variable que se pasa para que se representen sus datos es “generoEdad”.

```
this.generoEdad = [
  {
    "name": "0-9",
    "series": [
      {
        "name": "Hombre",
        "value": this.hombres[0]

      },
      {
        "name": "Mujer",
        "value": this.mujeres[0]
      }
    ]
  },

  {
    "name": "10-19",
    "series": [
      {
        "name": "Hombre",
        "value": this.hombres[1]
      },
      {
        "name": "Mujer",
        "value": this.mujeres[1]
      }
    ]
  },
```

Figura 24: Datos a representar

Como podemos observar en esta figura, hay que pasarle la posición del array que nos interese, y se mostrará correctamente.

De esta gráfica se decidió sólo coger el dato numérico, y el resto “el género y el rango de edad” se colocaron a mano, porque al ser algo estático, no hace falta que funcione con variables, y así tiene menos peso.

En el caso de ser un objeto de enormes dimensiones, habría que haber generado estos datos dinámicamente, recorriéndolo y asignando tantos como elementos haya. Pero en este caso, no ha sido necesario por el ligero volumen de datos.

5.5 Extras

5.5.1 Modo día/noche

Finalmente, una vez conseguido todos estos pasos, se ha implementado un modo nocturno, debido a que actualmente muchos usuarios lo prefieren y es una función que ningún desarrollador debe pasar por alto.

Para esta funcionalidad, hemos utilizado el código de la función mat-slide-toggle para su realización.

```
<mat-slide-toggle (click)="switchTheme()"
class="example-margin"
[color]="color"
[checked]="checked"
[disabled]="disabled">
  {{ theme === 'light-theme' ? 'Modo día' : 'Modo noche' }}
</mat-slide-toggle>
```

Figura 25: Código Toggler

Finalmente, se decidió por mostrar otro tipo más de datos, los *iframes.*

5.5.2 Iframes

Existe otro sitio web que proporciona datos a nivel nacional, y que proporciona exactamente los datos que queremos mostrar. La inconveniencia que presenta esta página es que el JSON sólo se puede descargar, pero no muestran la URL de la API, por lo que sólo podríamos acceder manualmente a esa información, descargando el fichero cada día y consultándolos.

Aún así, existe dentro de esa misma página una alternativa, los *iframes*.

Esta página proporciona *iframes* de gráficas temporales de muchos tipos de datos, por lo que se ha decidido usar esos iframes.

Los iframes tienen la peculiaridad de que son muy poco personalizables(por no decir nada). Aún así, si la tabla ofrece lo que nosotros buscamos, es una opción bastante buena.

Por último, como ya se ha realizado toda la estructura y el flujo de datos es el deseado, se ha estilizado la página mediante los documentos SCSS para darle una paleta de colores, de formas y efectos deseados.

6. Implantación

En este apartado se comentará el orden de pasos a seguir para poder lanzar esta aplicación en cualquier equipo.

- Tener cuenta en git, como ya se comentó en apartados anteriores.
- Crear una carpeta en tu equipo
- Realizar la clonación del repositorio del proyecto

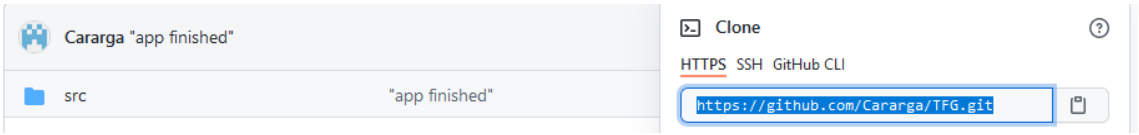


Figura 26: Repositorio a clonar

```
D:\TFG>git clone https://github.com/Cararga/TFG.git
Cloning into 'TFG'...
remote: Enumerating objects: 61, done.
remote: Counting objects: 100% (61/61), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 61 (delta 8), reused 57 (delta 8), pack-reused 0R
Receiving objects: 100% (61/61), 137.10 KiB | 1.37 MiB/s, done.
Resolving deltas: 100% (8/8), done.
```

Figura 27: Clonación del proyecto

- Desde la carpeta de proyecto, ejecutar la orden npm -i para generar la carpeta node_modules y que se sincronicen las dependencias con las del proyecto.

Figura 28: Ejecutar npm install

- Finalmente, ejecutaremos ng serve -o para ejecutar el proyecto.

```
PS D:\TFG\TFG> ng serve -o
? Port 4200 is already in use.
Would you like to use a different port? Yes
⠋ Generating browser application bundles (phase: setup)...Compiling @angular/core : es2015 as esm2015
Compiling @angular/animations : es2015 as esm2015
Compiling @angular/cdk/keycodes : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/cdk/observers : es2015 as esm2015
Compiling @angular/animations/browser : es2015 as esm2015
Compiling @angular/cdk/collections : es2015 as esm2015
Compiling @angular/cdk/platform : es2015 as esm2015
Compiling @angular/cdk/bidi : es2015 as esm2015
```

Figura 29: Lanzar la aplicación

Figura 30: Resultado

Si se han realizado los pasos correctamente, se habrá conseguido clonar y ejecutar el proyecto.

7. Validación de la web

En este apartado se mostrarán los resultados por los que fuimos pasando en el apartado anterior.

Al usar las librerías mencionadas anteriormente y enlazarle los datos de las APIs, el resultado de la página sería el siguiente:

- Página “Infectados”:

Figura 31 : Gauge con datos

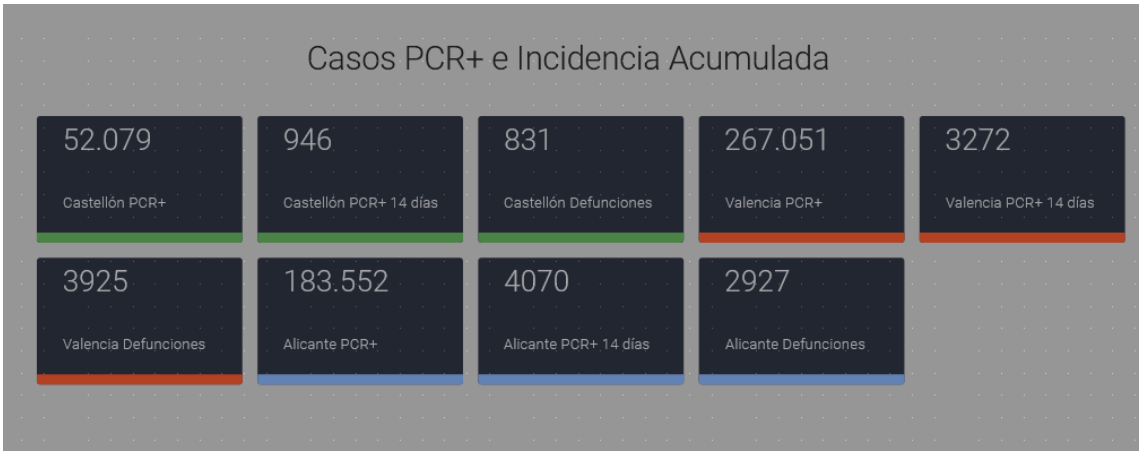


Figura 32: Casos PCR e Incidencia Acumulada

En esta figura, hubo que tocar la paleta de colores, de modo que los datos de cada provincia salieran del mismo color, para facilitar la comprensión visual por parte del cliente. En caso de no ir agrupados, y repetirse algún color en otra provincia, podría dar lugar a confusión.

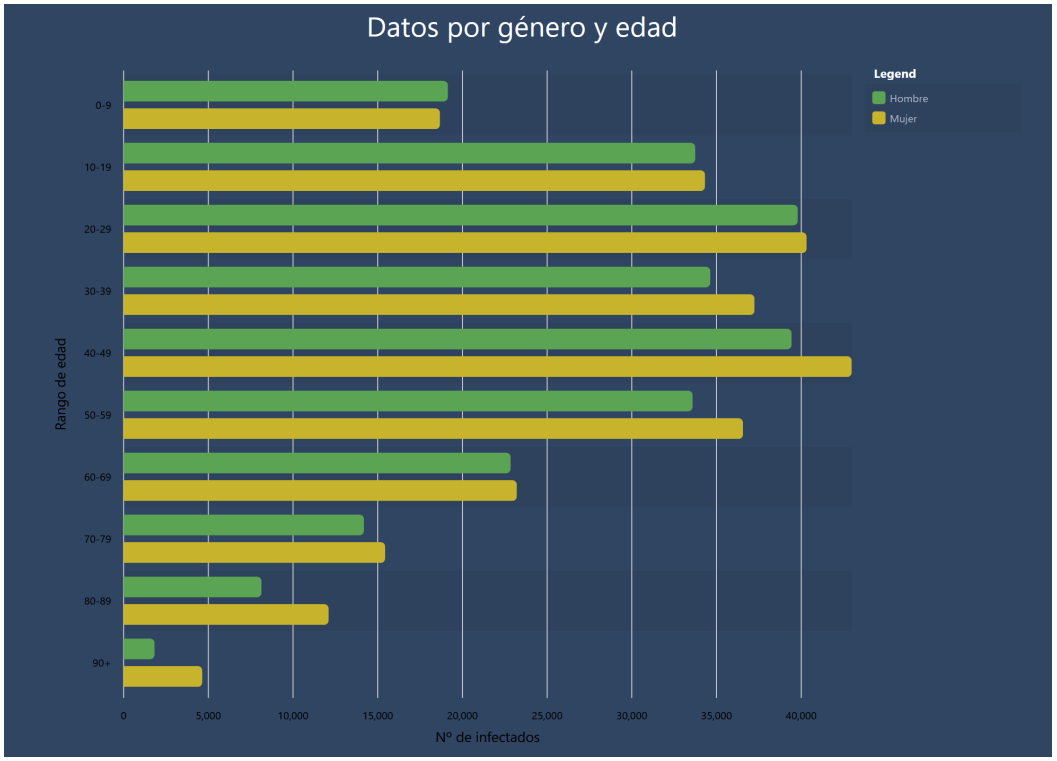


Figura 33: Resultado gráficas barras

Este procedimiento es el mismo que se ha seguido para todas las APIs de la generalitat, así que no explicaremos aquí el resto, puesto que es la misma implementación.

Como se comentó en el apartado anterior, íbamos a hacer uso de los *iframes*.

En este caso, sacamos para la página de infectados este iframe:

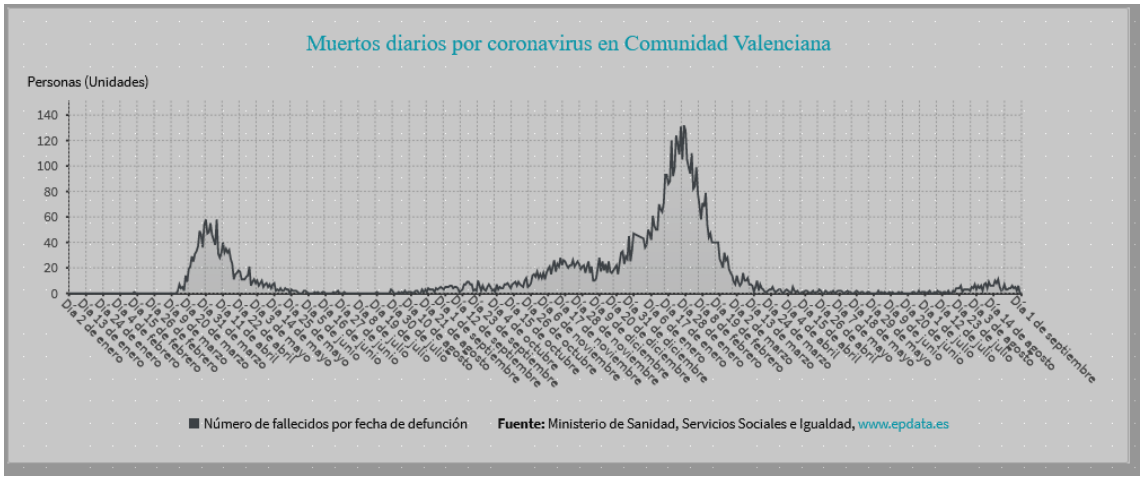


Figura 34: Iframe de defunciones

Como vemos, los iframes son una opción muy buena cuando el dato que queremos mostrar coincide con el que se nos proporciona, siempre y cuando no queramos aplicarle estilos propios.

Con esta implementación, hemos insertado ya datos de 3 modos diferentes, por lo que hemos podido comprobar lo importante que es abarcar varias opciones y no cerrarse solo en un tipo.

- Página “Vacunación”

En esta página tratamos con la API JSON, y los resultados fueron los siguientes:

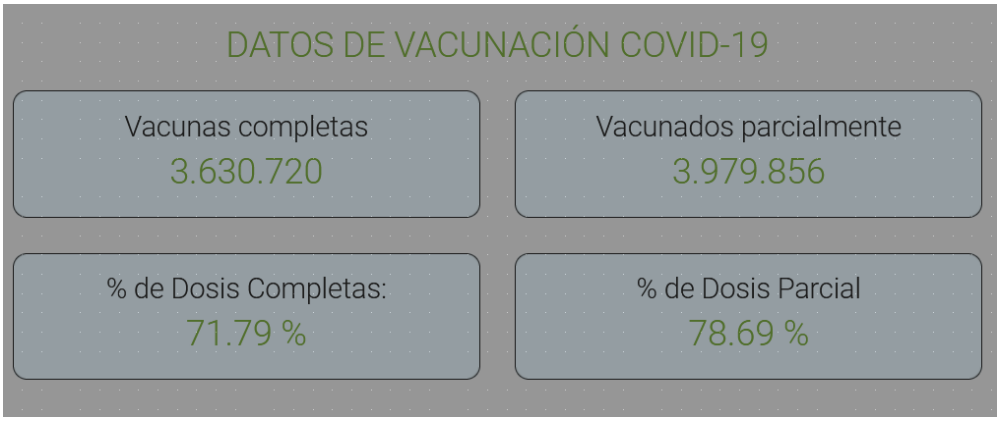


Figura 35: Datos Vacunación

Para implementar estos datos, se realizó lo mismo que ya vimos en el CSV: asignar el dato recuperado a una variable, y esa variable que se muestre. Con la diferencia de que esta consulta es bastante más rápida, ya que simplemente con el poco código que vimos en el apartado anterior, conseguimos sacar estos datos, y los de las dos siguientes figuras.

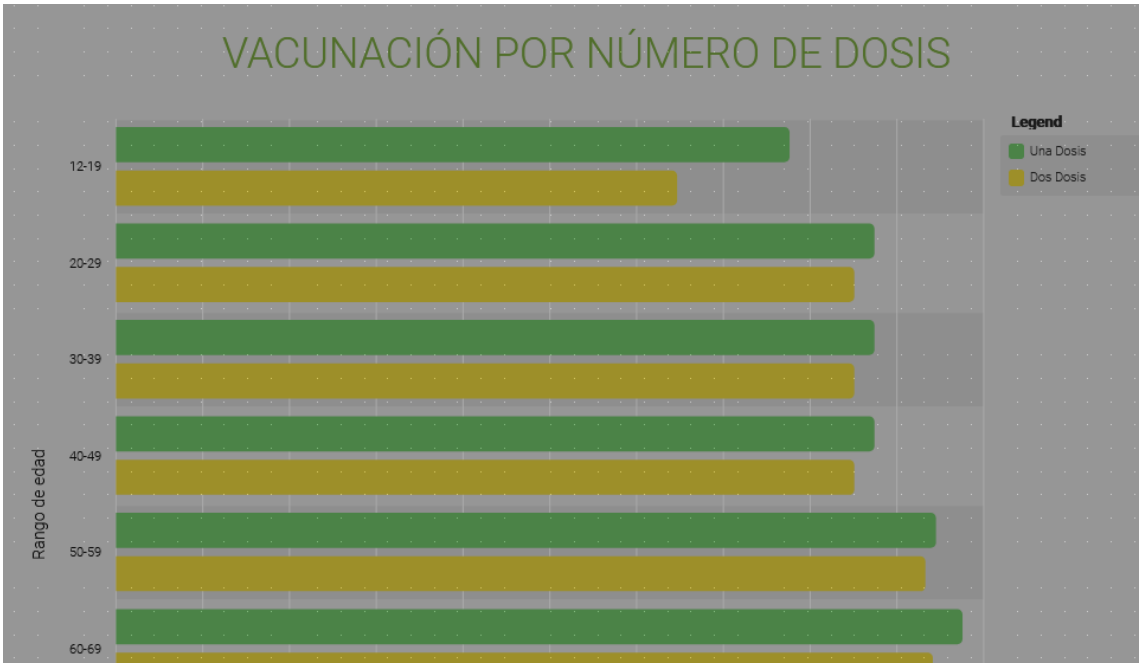


Figura 36: Datos de vacunación por edad

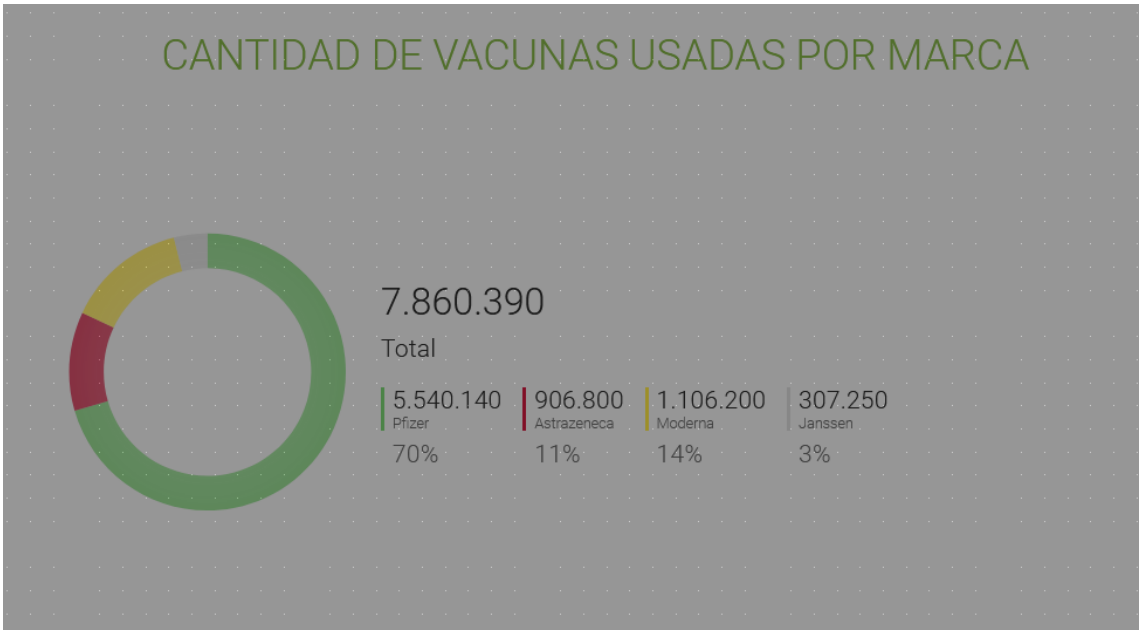


Figura 37 : Datos de marcas de vacuna

Figura 38: Iframe temporal de vacunados

Como vemos, estas librerías de métodos de representación de datos son muy útiles y flexibles.

- Página “Casos por Municipios”

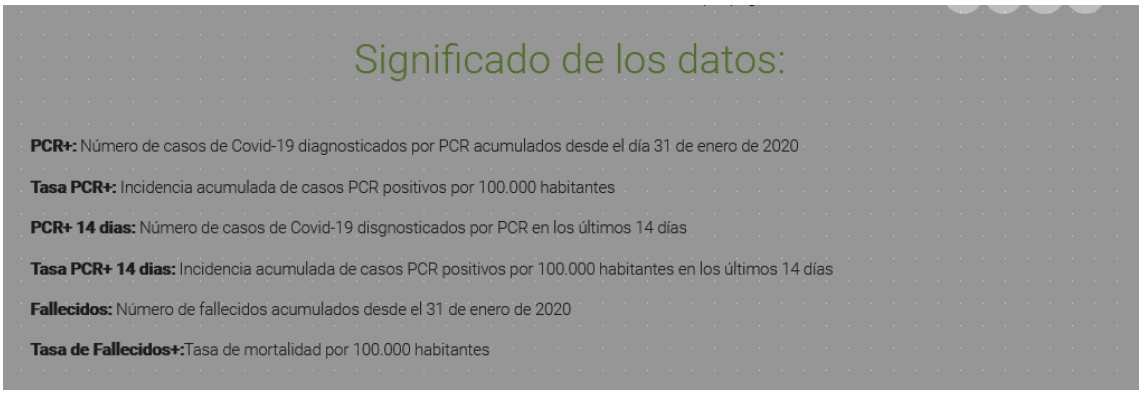
En el caso de esta página, podemos observar la table con los datos ordenados por municipio (aunque se puede ordenar por casos también).

Casos de COVID-19 por municipio en la Comunidad Valenciana

Municipio	PCR+	Tasa PCR+	PCR+ 14 dias	Tasas PCR+ 14	Fallecidos	Tasa de Fallecidos
Ademuz	202	19479.27	18	1735.78	1	96.43
Ador	189	10924.86	3	173.41	2	115.61
Adsubia	19	3109.66	0	0.0	1	163.67
Agost	380	7926.57	4	83.44	11	229.45
Agres	36	6382.98	0	0.0	1	177.3
Agullent	99	4145.73	0	0.0	3	125.63
Aielo de Malferit	412	8908.11	0	0.0	5	108.11
Aielo de Rugat	1	625.0	0	0.0	0	0.0

Figura 39: Tabla de datos por municipio

Como alguno de estos datos puede resultar un poco liante, se ha decidido crear un apartado bajo de la gráfica, para explicar cada tipo de datos lo que significa.



Significado de los datos:

PCR+: Número de casos de Covid-19 diagnosticados por PCR acumulados desde el día 31 de enero de 2020

Tasa PCR+: Incidencia acumulada de casos PCR positivos por 100.000 habitantes

PCR+ 14 dias: Número de casos de Covid-19 disgnosticados por PCR en los últimos 14 días

Tasa PCR+ 14 dias: Incidencia acumulada de casos PCR positivos por 100.000 habitantes en los últimos 14 días

Fallecidos: Número de fallecidos acumulados desde el 31 de enero de 2020

Tasa de Fallecidos+: Tasa de mortalidad por 100.000 habitantes

Figura 40: Explicación de los datos

- Header

En la cabecera, pusimos un navbar y una foto separadora como background.

En el caso del header, cambiamos el diseño de los botones, y se ha añadido el modo Día/Noche, para que los contenidos principales de la página sean más claros o más oscuros.

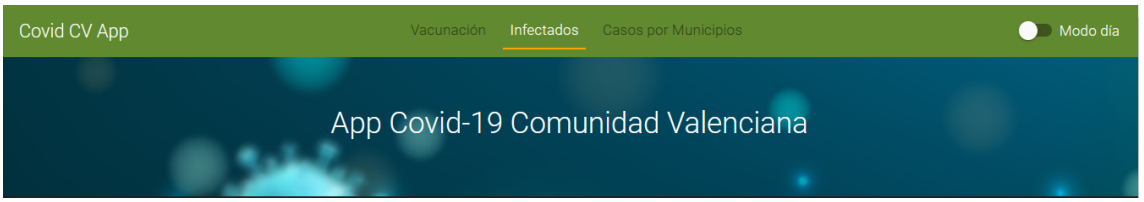


Figura 41: Navbar del Header

- Footer

En el pie de página, creé un div con los datos personales míos y de las dos tutoras que he tenido durante el proyecto, junto con nuestro nombre, apellidos y email de la universidad.

Figura 42: Footer

8. Conclusiones

Podemos decir que ha sido un proyecto muy interesante y especialmente útil, debido a que he utilizado herramientas y tecnologías que durante el grado no había tocado.
De las asignaturas que más me gustaron a lo largo de los 4 cursos, fueron justamente las que más tenían que ver con este tipo de desarrollo, y ha sido un placer entrar en el mundo de Angular, descubrirlo y crecer dentro de él.

Al no conocerlo, fue duro, especialmente al principio, pero es un muro que hay que conseguir sobrepasar en cualquier lenguaje y/o tecnología en la que nos adentremos.

Así que como conclusión, personalmente me encuentro muy satisfecho con el trabajo realizado. Me habría gustado tener más tiempo para ampliarlo más, pero ya me ha dado pie para empezar a hacer proyectos por mi parte, dado a que ya estoy familiarizado con este framework.

Respecto a la aplicación en sí, me gusta su resultado final, ya que al final cumple con lo que yo buscaba al principio: sencillez.

Quería una aplicación que cualquier usuario pudiera entender fácilmente con poco más que un vistazo, y creo que lo he conseguido.

8.1 Relación del trabajo desarrollado con los estudios cursados

Durante la realización de este proyecto, se han utilizado tecnologías vistas en diferentes asignaturas, o tecnologías similares a las mismas. A continuación, acompaño con un listado de asignaturas que han sido las que han hecho que me cante por este proyecto.

1. **Interfaces Persona Computador (IPC):**

Interfaces Persona Computador fue aquella asignatura que me abrió la mente hacia el código (puesto que anteriormente no me había llamado la atención ninguna asignatura que requiriera escribir código) y me gustó diseñar interfaces y llevarlas a cabo, por lo que fue el primer paso que si en el diseño de interfaces para aplicaciones/webs.

2. **Ingeniería del Software (ISW):**

Aunque no rescate mucho en este proyecto de mi paso por Ingeniería del Software, la primera experiencia real con el desarrollo de una aplicación sentí que fue en esta asignatura. Fue la primera vez que usé el control de versiones, y, aunque fue un poco caótica la organización de las ramas, me parece que fue también una de las cosas que más útiles son en un trabajo real. El código de ISW poco tiene que ver con el de este proyecto realizado en angular, pero aunque el lenguaje cambie, la experiencia se queda, y me parece que esta asignatura fue una de las que más aprendí, tanto a nivel de conocimientos como a nivel de buenas prácticas.

3. **Desarrollo Web (DEW):**

Desarrollo Web fue la siguiente asignatura que me hizo sentirme parte de un proyecto grande, y esta asignatura puede que sea la que más tiene que ver con este proyecto. Me habría gustado ver en esta asignatura algún framework como angular, pero puesto que se llegaba a este punto de la carrera sin haber tocado casi esta rama, lo considero normal. De esta asignatura, saco para este proyecto todo el uso de maquetación de HTML con CSS, el uso de Bootstrap, y el JQuery.

4. **Desarrollo Centrado en el Usuario (DCU):**

De DCU saco todas las buenas prácticas a la hora de diseñar una web. Aunque en el diseño web siempre se puede mejorar y también es algo que evoluciona constantemente,es una asignatura más que obligatoria para la realización de cualquier interfaz.

5. **Integración de Aplicaciones (IAP):**

De IAP destaco notablemente el trato de datos CSV y JSON, que es una cosa con la que te vas a encontrar con total seguridad en cualquier aplicación que use datos de una API.

9. Trabajos futuros

En el futuro me gustaría seguir ampliando esta aplicación, o incluso utilizar la base para crear otros proyectos similares.

Una de las cosas que más me habría gustado poner, habría sido un mapa, dividido por municipios, en los que al pasar el ratón por encima, mostrasen los datos de ese municipio, para poder verlo más claramente.

El problema de esa tarea es que era muy compleja y requería de mucha dedicación, y preferí optimizar el tiempo en otras tareas, y esa información volcarla en una tabla, que era menos costoso y no perdía información.

Como he comentado ya varias veces, me gustaría realizar proyectos similares, ya que tratar con diferentes APIs ha sido una experiencia compleja pero a la vez muy satisfactoria, y me gustaría poder optimizar estos procesos lo máximo posible.

10. Referencias

1. Angular - De cero experto - Edición 2021. Fernando Herrera. Disponible en : https://www.udemy.com/course/angular-fernando-herrera/ .
2. Entendiendo Providers en Angular. Cristian Flores. Disponible en: https://medium.com/@cristianflores.ee/providers-en-angular-1832e8093e2a
3. Claves para entender Angular - Qué es y cómo se utiliza? Ro. Disponible en https://www.acontracorrientech.com/claves-para-entender-angular-que-es-y-como-se-utiliza/
4. TypeScript para Principiantes. Daniel Diaz Suarex. Disponible en: https://tutorialesenpdf.com/typescript/
5. Servicios en Angular. Disponible en: https://desarrolloweb.com/articulos/servicios-angular.html
6. Angular: componentes y módulos. Disponible en: https://jotagep.com/blog/angular-modulos-componentes
7. Angular charts with ngx-charts. Genka. Disponible en https://www.youtube.com/watch?v=IGSFqEZzc5A
8. Complete angular material documentation. Disponible en: https://material.angular.io/
9. Comunidad Valenciana - Bacunas contra el coronavirus en cada comunidad, en datos y gráficos. Ministerio de sanidad. Disponible en: https://www.epdata.es/datos/vacunas-coronavirus-comunidades-autonomas-datos-graficos-mapas/573/comunidad-valenciana/299
10. Complete NGPrime documentation. Disponible en: https://www.primefaces.org/primeng/
11. CSS Examples. Disponible en: https://www.w3schools.com/css/
12. Learning Angular Material. Disponible en https://riptutorial.com/Download/angular-material.pdf
13. 10 Best Angular DataTables with Pagination, Sorting and Filter. Ankit Prajapati. Disponible en: https://www.ngdevelop.tech/best-angular-tables/
14. How to call another components function in angular2. Disponible en: https://stackoverflow.com/questions/37587732/how-to-call-another-components-function-in-angular2

15. Express.js Deep API Reference. Disponible en: https://www.programmer-books.com/express-js/
16. Fetch - Javascript - Cómo mostrar datos desde una API. Disponible en: https://www.youtube.com/watch?v=G-j5SI7Qitk

11. Anexo

11.1 Enlaces de interés

Enlace 1: https://covid-vacuna.app/data/latest.json

Enlace 2:
https://dadesobertes.gva.es/dataset/3b0b53a4-2383-4d04-afb9-5dca8cda9b5c/resource/af0ce7ce-fd6a-41e7-9449-2a956baa3760/download/covid-19-datos-de-casos-y-personas-fallecidas-por-grupo-de-edad-y-sexo-acumulados-desde-el-31-01.csv

Enlace 3:
https://dadesobertes.gva.es/dataset/38e6d3ac-fd77-413e-be72-aed7fa6f13c2/resource/170ef5da-e93f-448c-82cb-0fa15e81530c/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv

11.2 Otras Instalaciones

En este apartado comentaremos todas las instalaciones requeridas para realizar este proyecto, y comentaremos posibles errores que nos ocurran.

- Instalación Angular.

Para la instalación de Angular, tendremos que escribir en una terminal: ng new “nombreProyecto”.

A la pregunta de si queremos el routing, le decimos que sí.

A la que nos preguntan qué tipo de estilos vamos a utilizar, seleccionamos SCSS.

Con esto, el proyecto estaría creado. Si ejecutamos ng serve -o, se ejecutará y se abrirá en nuestro navegador, y veremos una ventana como esta:

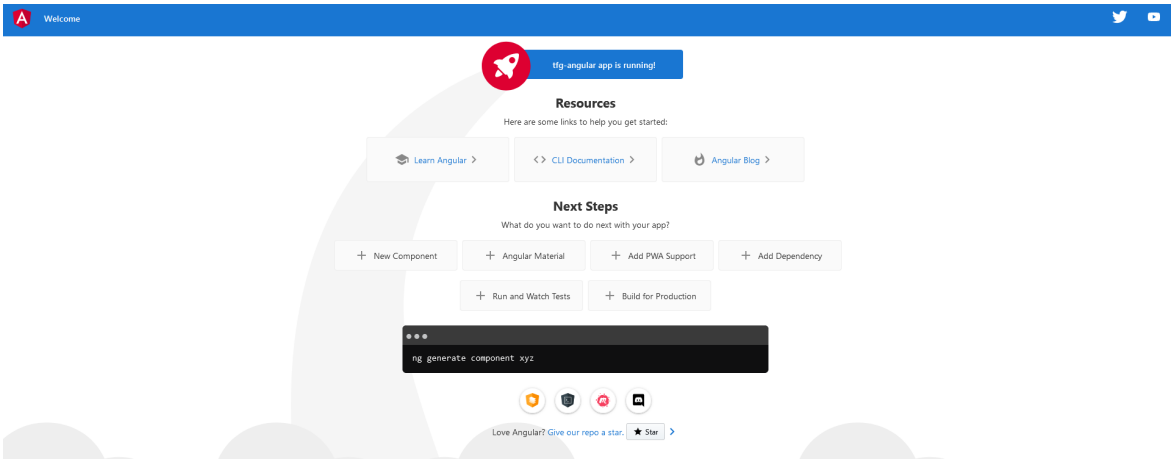


Figura 43: Página base Angular

- Instalación Bootstrap

Para instalar bootstrap hay que ir a su web oficial, y tenemos que coger la link href del css y el script de javascript que ponen con la documentación y referenciarlas en nuestro Index.html, o bien descargar la última versión de bootstrap, colocarla en assets(o donde se desee) y referenciarla desde el index.

En el caso de este proyecto, se optó por la primera opción.

- Instalación Ngx Charts

Esta librería la instalé mediante el comando “npm install @swimlane/ngx-charts --save”, y se tiene que importar en el app.module para que funcione.

- Instalación Jquery

Con Jquery tenemos varias opciones para instalarlo. Podemos hacerlo por npm, o referenciándolo con un enlace en el Index.html, así que hemos optado por esa opción, ya que al no pesar mucho, el hecho de referenciarlo por enlace no entorpece su uso.

- Configuración Github

Para esta configuración, lo primero que tenemos que hacer es crear un repositorio en angular, y en el ordenador una carpeta donde vamos a meter el proyecto, y clonar ahí el repositorio. con el enlace que te proporcionen ellos. Si todo está bien, se clonará.

Debemos tener una cuenta configurada en el PC. Si no, necesitamos nuestro usuario y email de github y tendremos que escribir lo siguiente en una consola:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Figura 44: Configurar cuenta de github

A continuación, haremos muestro primer commit con “git commit -m ‘Primer commit’ “, donde el parámetro -m será el mensaje de identificación que le queramos dar al commit. El commit para lo que sirve es para guardar los cambios y dejarlos pendientes de subir a github.

Si tenemos el commit hecho, lo podremos subir a github con un “git push”.

Si hemos realizado estos pasos correctamente, tendremos nuestro directorio configurado y podremos trabajar sobre él con control de versiones.

- Configuración Angular Material

Primeramente, debemos instalar el Angular material, Angular CDK y las Animaciones.

Esto se hará con el comando “npm install @angular/material @angular/cdk @angular/animations --save” y posteriormente, usaremos el comando “ng add @angular/material” para actualizar el proyecto con las dependencias correctas, y podremos seleccionar un tema predeterminado.

Por último, hay que importar los módulos que queramos utilizar en el documento que corresponda.

- Configuración Servidor Node con Express

Por último, comentaré cómo se hizo el servidor de express que finalmente se pudo utilizar correctamente por los problemas mencionados anteriormente.

```
ex.js > ...
const express = require('express');
const ApiGob = require('./utils/ApiGob');
const PORT = 3000;
const app = express();

app.get('/distributions', async (req, res) => {
  const { body, params, query } = req;

  const api = new ApiGob();

  try {
    const distributionUrls = await api.getDistributions();
    const distributions = [];
    for (const url of distributionUrls) {
      const distribution =  api.parseCSV(url);
      distributions.push(distribution);

    }
    res.json({
      distributions,
    });
  } catch (e) {
    res.status(500).json({
      error: e.message,
    });
  }


});



  app.listen(PORT, () => {
    console.log('Server started on port', PORT);
  });
```

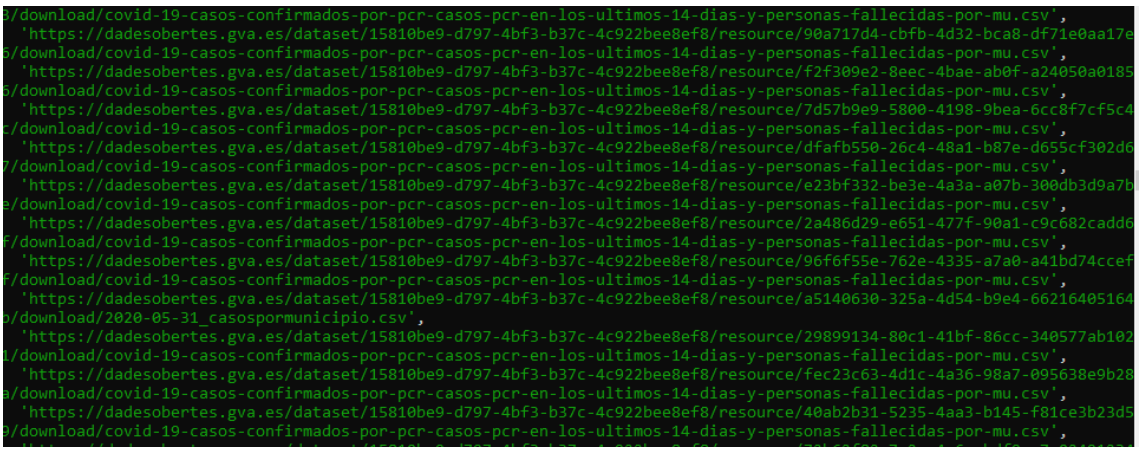
Figura 45: index.js de Express

Como observamos en la figura, hacía una llamada a ApiGob cuando se llamase al servidor localhost:3000, y se trataban los datos para recibir las URLs de la API.

```
const { json } = require('body-parser');
const fetch = require('node-fetch');
const http = require('http');


class ApiGob {
 urlApi = "https://datos.gob.es/apidata/catalog/dataset/a10002983-covid-19-casos-confirmados-po

   constructor() {
   }
   async getDistributions() {
     return fetch(this.urlApi)
       .then(res => res.json())
       .then(data=> {
         console.log('DATA', data);
         const { items } = data.result;
        // console.log('ITEMS', items);
         return items;
       })
       .then(items => {
         const distributions = [];
         for(const item of items) {
           for (const distribution of item.distribution) {
               distributions.push(distribution.accessURL);
               console.log(distributions);

           }
         }

         return distributions;
       }).catch(e => {
         console.error('ApiGob: error', e.message);
         throw e;
       });

   }
   async parseCSV(url) {
     const csvFilePath=url;
     const csv=require('csvtojson');
     csv({delimiter:';'})
     .fromFile(csvFilePath)
     .then((jsonObj)=>{
     console.log(jsonObj);

      return jsonObj;
```

Figura 46: ApiGob

En esta figura, como podemos observar, se usaba la URL que contenía todas las URLs de la API, por mala gestión de estas APIs por parte de la generalitat, al hacer la llamada, generaba lo siguiente:



```
3/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/90a717d4-cbfb-4d32-bca8-df71e0aa17e
6/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/f2f309e2-8eec-4bae-ab0f-a24050a0185
6/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/7d57b9e9-5800-4198-9bea-6cc8f7cf5c4
c/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/dfafb550-26c4-48a1-b87e-d655cf302d6
7/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/e23bf332-be3e-4a3a-a07b-300db3d9a7b
e/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/2a486d29-e651-477f-90a1-c9c682cadd0
f/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/96f6f55e-762e-4335-a7a0-a41bd74ccef
f/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/a5140630-325a-4d54-b9e4-66216405164
b/download/2020-05-31_casospormunicipio.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/29899134-80c1-41bf-86cc-340577ab102
1/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/fec23c63-4d1c-4a36-98a7-095638e9b28
a/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
  'https://dadesobertes.gva.es/dataset/15810be9-d797-4bf3-b37c-4c922bee8ef8/resource/40ab2b31-5235-4aa3-b145-f81ce3b23d5
9/download/covid-19-casos-confirmados-por-pcr-casos-pcr-en-los-ultimos-14-dias-y-personas-fallecidas-por-mu.csv',
```

Figura 47: Respuesta por consola

Como podemos comprobar en la figura 42, las URLs se rescataban correctamente, y esas URL funcionan, pero no estaban ordenadas por fecha, por lo que no podíamos

averiguar cuál era la última ni ordenarlas por fecha para realizar una gráfica, por lo que se abandonó este código, pero veía importante mostrarlo porque si la API hubiera estado ordenada