FORMATEO DE RESULTADOS OFRECIDOS MEDIANTE XML

Lenguaje de marcas

Antonio Ferrer Lopez

Índice

1.Introducción.

a. Exposición de la práctica.

Mediante esta práctica se pretende realizar la exposición ordenada de los datos entregados en un archivo formateado en Xml.

En primera instancia se presentará un formateo limitado y sin filtrar de los datos empleando UNICAMENTE css. Puesto que dicha técnica es limitada en cuanto a presentación, ordenación y filtrado de datos, en una segunda parte procederemos a la creación de una aplicación realizada en React con la que daremos estructura, orden y filtrado de los datos. Por último, implementaremos unas gráficas a nuestra aplicación mediante la librería ChartJs-2.

Puesto que dicho archivo carece de un formato entendible para el humano procederemos a realizar el formateo del mismo y la presentación de los datos de forma amigable.

b. Visita la aplicación en funcionamiento.

Puesto que la tecnología de implementación escogida es React, y para facilitar el acceso a la visualización de la misma EN FUNCIONAMIENTO y poder descargar el código fuente, se publica en Github la misma pudiendo acceder a la aplicación desde:

https://antonioferrerlopez.github.io/covidXmlReact/

También se puede descargar el código fuente desde:

https://github.com/AntonioFerrerLopez/covidXmlReact

Nota:
También se adjunta el código fuente de la aplicación en la entrega.

2. Presentando datos mediante css (visualmente limitado).

Como primera práctica, realizaremos un formateo simple de los datos simplemente empleando css.

A continuación se presenta el archivo css, Cabe destacar algunos elementos empleados para añadir texto que le otorgue cierto sentido a los datos como es el uso de *'content'*, el cual nos permite añadir texto, en este caso lo empleamos junto con *:before* para añadirlo ANTES de la etiqueta seleccionada , con lo que conseguimos concatenar los resultados.
Por otro lado, se ha añadido un efecto zoom cuando se pasa el ratón sobre cada uno de los elementos para enfatizar los puntos de atención.

casos.css

```css
records:before{
    content: "LISTADO DE CASOS : \A ";
    white-space: pre;
    margin: 1em;
}

record{
    display: flex;
    justify-content: space-around;
    align-items: center;
    flex-wrap: wrap;
    margin : 1em;
    width: 50%;
    border: 1px solid black;
    background: #ecfff3;
}
record:hover{
    font-size: 1.1em;
}
record:nth-child(odd){
    background: #c6ffe0;
}
countriesAndTerritories:before {
    content: "País: ";
}
dateRep{
    font-size: large;
    font-weight: bold;
    display: block;
}
cases:before{
    content: "Casos confirmados: ";
}
deaths:before{
    content: "Fallecidos: ";
}

day , month ,year,geoId,countryterritoryCode,popData2018 {
   display: none;
}
```

Para poder hacer uso de dicho css, es necesario indicarle al archivo xml que debe emplear esta hoja de estilos, para lo cual se debe añadir una única línea de código al xml:

```
2    <?xml-stylesheet href="casos.css"?>
```

A continuación, se muestra una captura de pantalla del resultado:

Estos resultados son poco manejables puesto que no nos permite filtrar los datos de forma adecuada y es complicado extraer conclusiones sobre dichos datos. Para mejorar esto se propone realizar una aplicación que nos permita realizar un filtrado sobre los mismos pudiendo mostrar los datos por cada país y nos muestre una gráfica para cada uno de ellos según la selección.

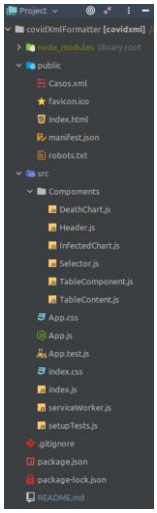
3. Presentando datos mediante React.

React es una librería de JavaScript que nos permite la creación modular de aplicaciones web, dichos componentes se pueden cargar, descargar o alterar en tiempo real permitiendo gran flexibilidad.
Si bien es cierto que las características que pretendemos se pueden implementar mediante tecnologías más simples, he creído interesante exponer este método por su flexibilidad, facilidad de modificación y porque considero una buena forma de practicar dicha tecnología.

A continuación, revisaremos la estructura del proyecto y los componentes.

a. Estructura de la aplicación.

Revisemos la estructura del programa a grandes rasgos.



Los archivos contenidos en **Public** los empleará el compilador para la realización de la "build".
Dentro del mismo podemos encontrar el archivo Index.html el cual contiene la estructura html base y el elemento raíz sobre el que se irán cargando los componentes a modo de lego con forme lo precisemos.

En el directorio **Src** encontraremos los archivos fuente con los que vamos a trabajar.
El punto desde el que parte la carga de nuestra aplicación es en App.js.
En esta carpeta también podemos encontrar los archivos css implicados en los componentes raíz.

La carpeta **Components** contiene cada uno de los componentes que vamos a ir construyendo para componer nuestra página.

En el directorio raíz del proyecto también podemos encontrar archivos de configuración como son package.json que se emplea para configurar NPM que nos permitirá instalar, actualizar nuestros módulos y desplegar o testear nuestra aplicación mediante scripts.

b. Los componentes.

App.js:

Es el componente padre desde el que se cargan el resto de componentes, este se encarga de recoger los datos del archivo y distribuirlo hacia el componente Selector.

Selector.js

Este componente recibe los datos de App.js y pinta un select-option en nuestra web, cargando primero los paises para poder realizar de forma dinámica la lista de "options". Una vez se ha seleccionado un país, este le envía al componente TableComponent.js los datos filtrados de ese país y carga dicho componente.

Si no se ha seleccionado un País, pinta sobre pantalla una indicación en vez de las tablas.

TableComponent.js

Se encarga de cargar una tabla en la que se pintarán una a una las líneas de esta, siendo cada línea un componente diferente (TableContent). Para ello le pasa los datos a TableContent.

TableContent.js

Este componente tiene varias funciones:

- Carga la tabla con los datos del país.
-Calcula los totales (Infectados y fallecidos) y los envía al componente Chip perteneciente a MaterialUi que nos pinta el resultado en formato píldora con un icono representativo.
-Envía los datos hacia los componentes de Gráficas (InfectedCharts.js y DeathsCharts.js).

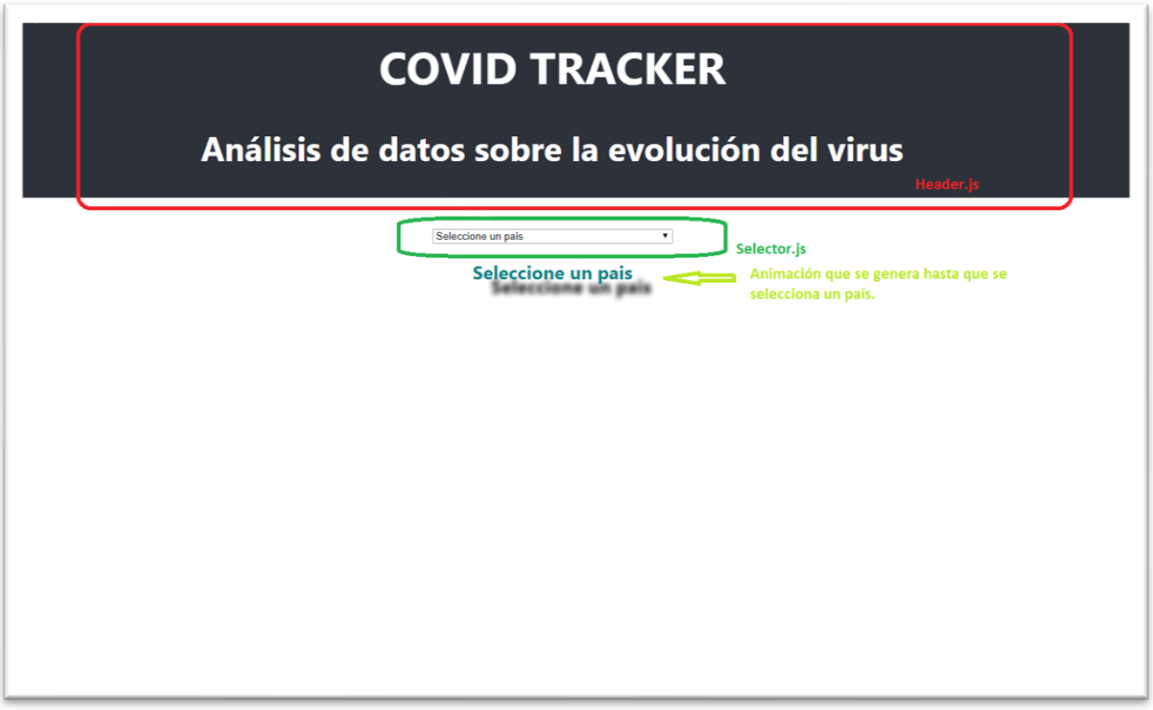
InfectedCharts.js y DeathsCharts.js

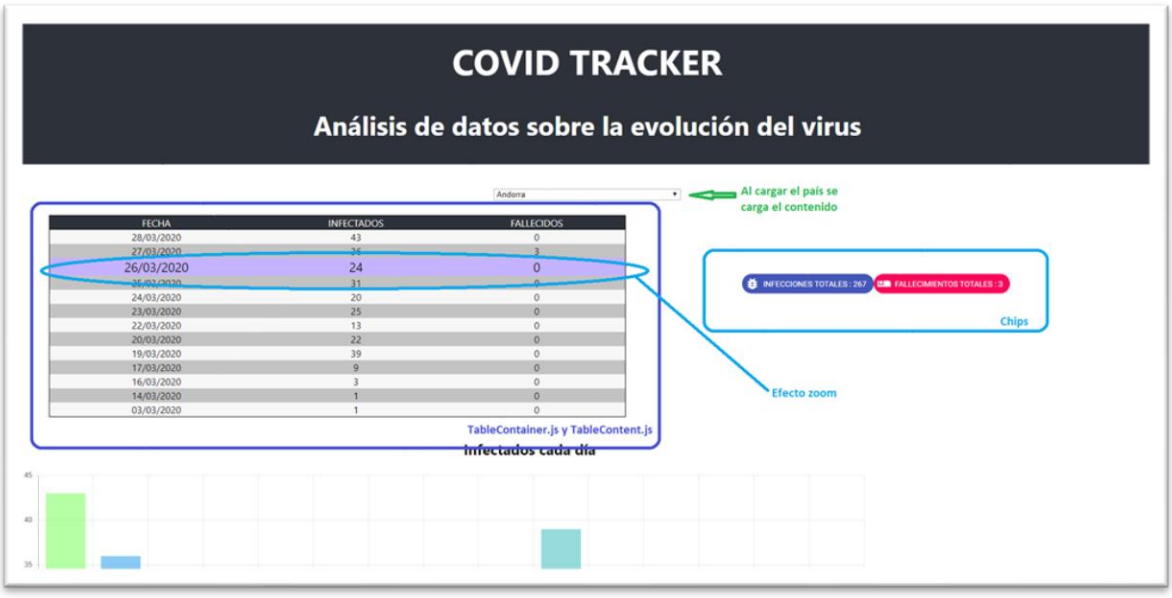
Cargan los gráficos mediante la librería externa Chartjs-2 de forma dinámica.

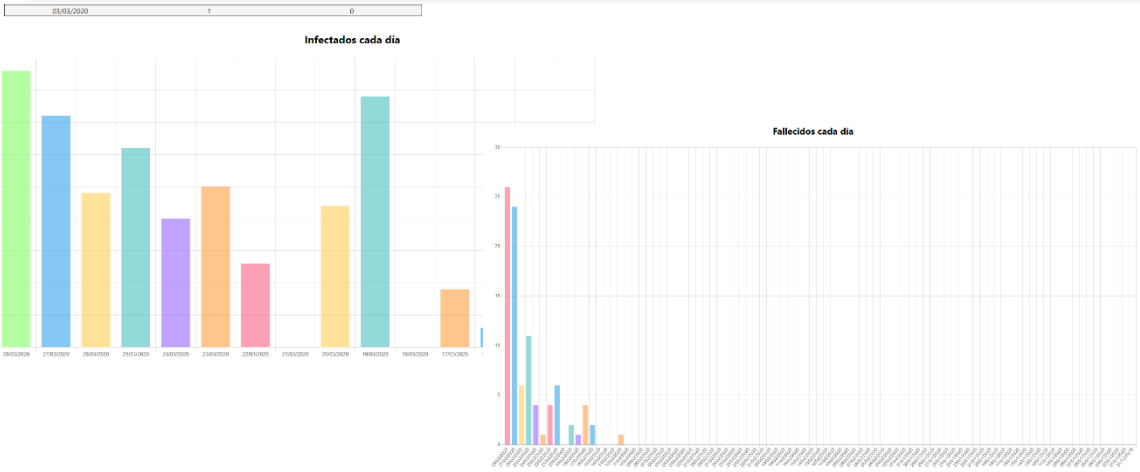
Header.js

Este componente pasivo, se carga en todas las páginas y contiene la cabecera de nuestra aplicación con su título.

c. Capturas de pantalla.







NOTA: esta última gráfica corresponde a otro país diferente a la tabla mostrada arriba.