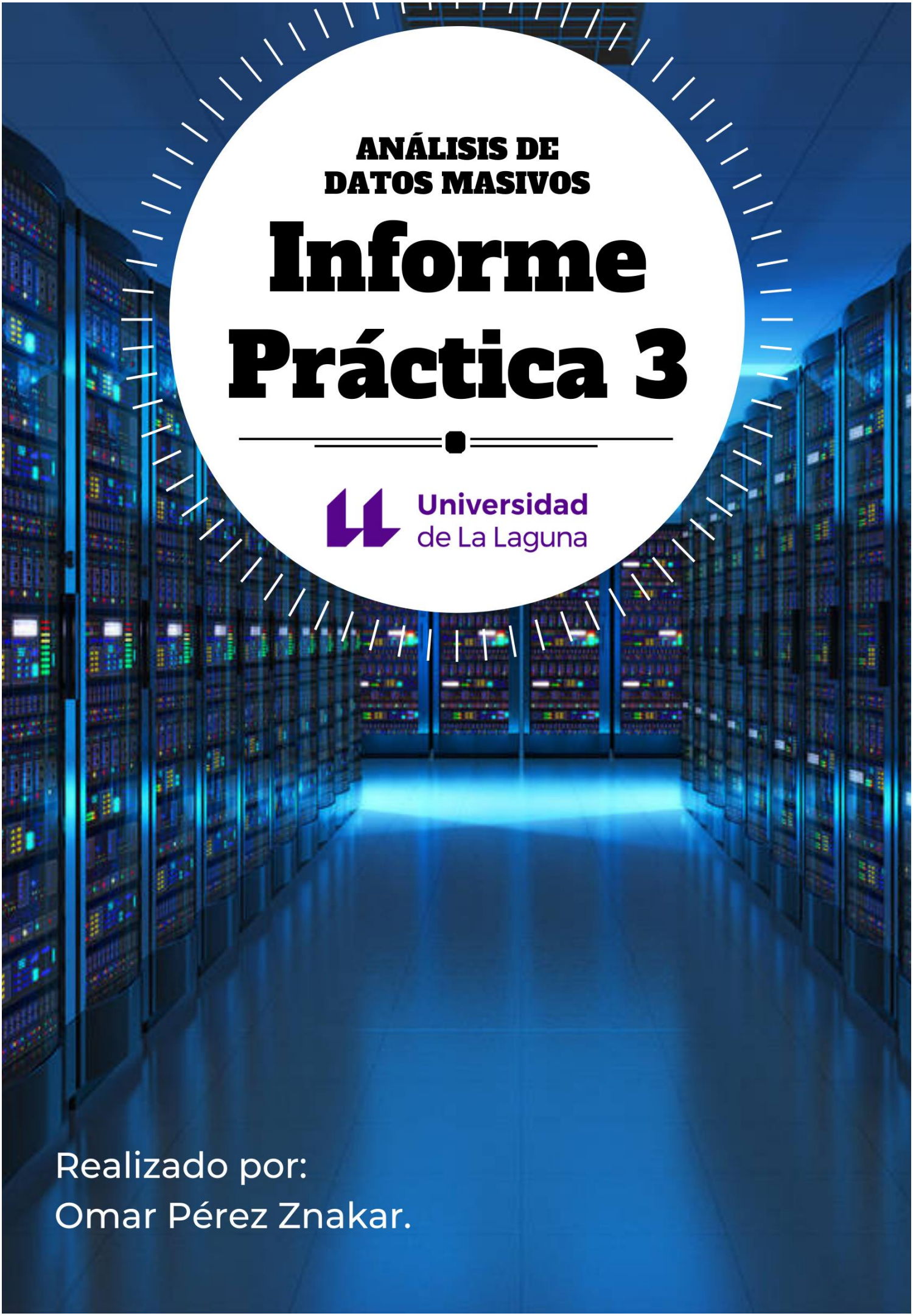


ANÁLISIS DE DATOS MASIVOS

Informe Práctica 3

Universidad de La Laguna

Realizado por:
Omar Pérez Znakar.

INDICE

1.- Introducción **3**

2.- Mejoras propuestas por el profesor. **4**

2.1 Mostrar las opciones elegibles. 4

2.2 Nuevas Gráficas. 4

2.2.1 Histograma. 4

2.2.1.1 Histograma único. 4

2.2.1.2 Histograma múltiple. 5

2.2.2 Cajas y Bigotes. 6

2.3 Nueva fuente de datos. 8

3.- Mejoras propuestas por el alumno. **8**

3.1 Gestión de grupos. 8

4.- Librería Seaborn. **11**

4.1 Explicación librería. 11

4.2 Gráficas implementadas. 11

4.2.1 Histograma. 12

4.2.2 Cajas y Bigotes. 13

4.2.3 Diagrama de Violín. 13

5.- Extracción de conclusiones. **14**

5.1 Tomé Cano (Tenerife) 15

5.2 Coronavirus. 16

5.3 Accidentes España. 16

6.- Servidor Computación en la nube. **18**

6.1 Arquitectura. 18

6.2 Mejoras. 19

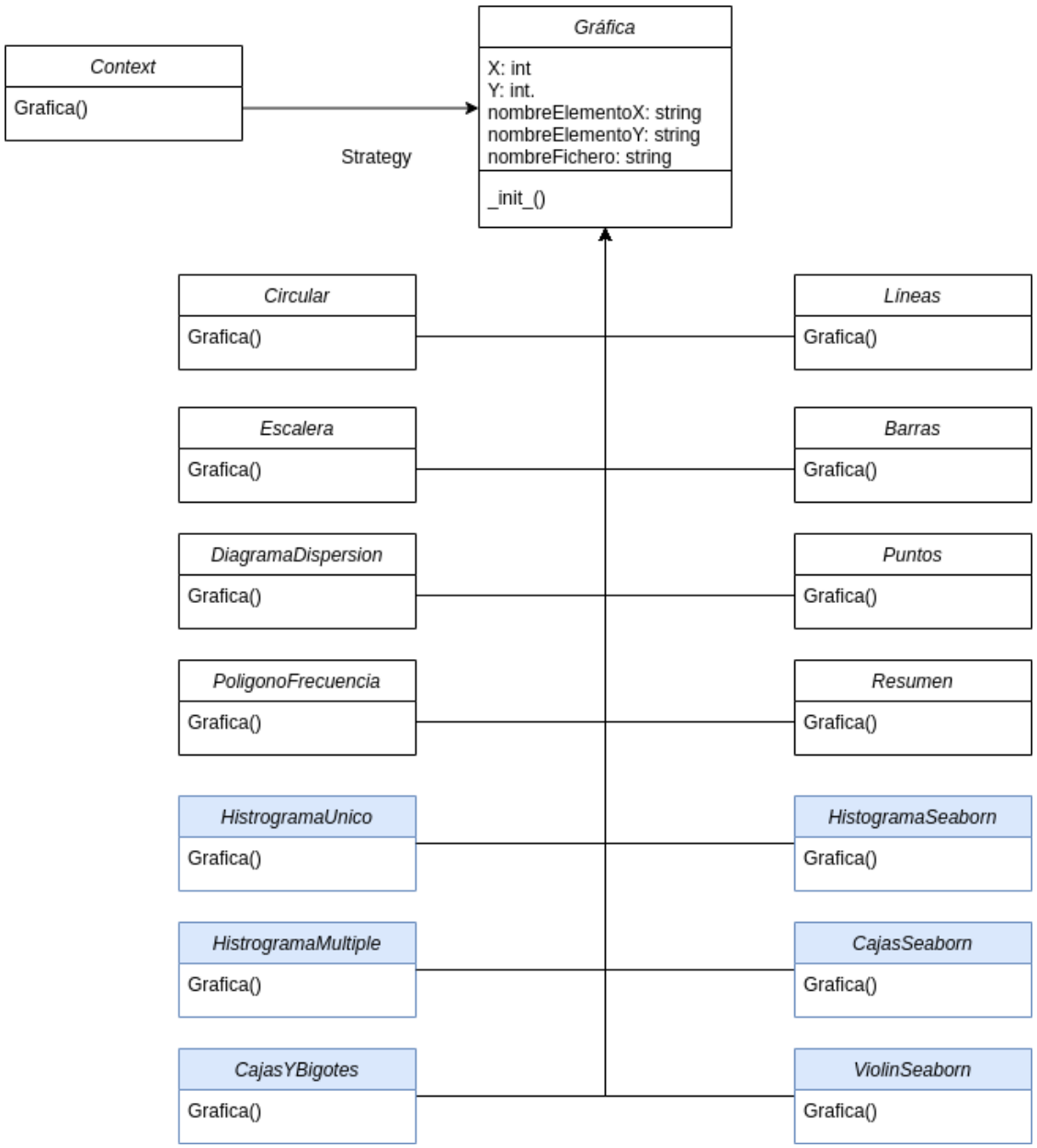
6.2.1 Ejecución de ejemplo para los algoritmos. 19

6.2.2 Mejora explicación de cada campo. 21

7.- Visión de futuro. **22**

1.- Introducción

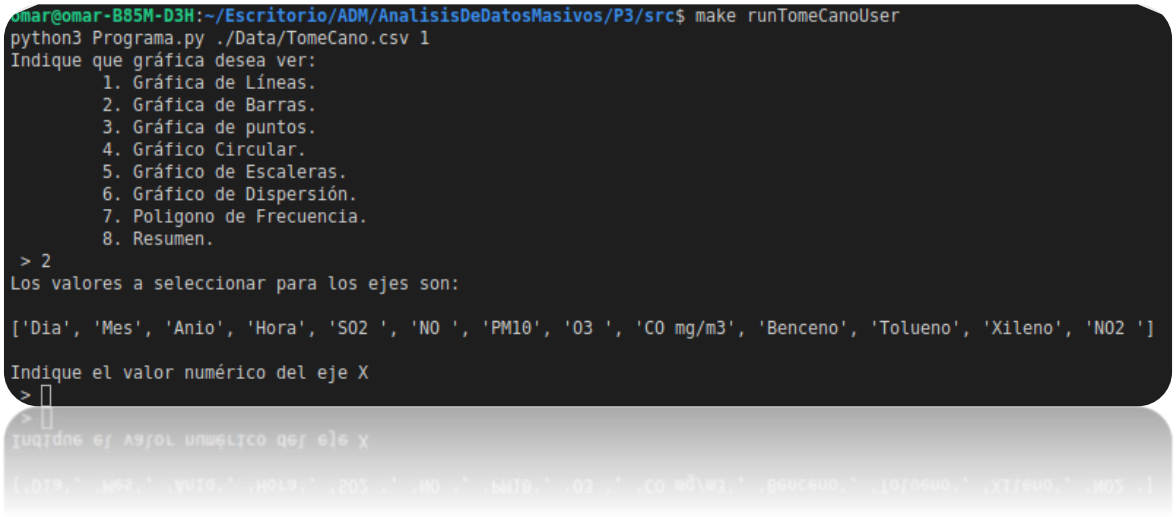
En esta práctica tocaremos diversos temas propuestos por el profesor. De igual forma, realizaremos las mejoras propuestas y, por tanto, nuestro diagrama de clases de gráficas quedará modificado correspondiendo con el que veremos a continuación (nuevas aportaciones en azul):



2.- Mejoras propuestas por el profesor.

2.1 Mostrar las opciones elegibles.

Tras la revisión de la semana pasada, el profesor me propuso poder ver las posibles opciones para facilitar la selección al usuario. Para ello, se ha realizado la siguiente aportación:



2.2 Nuevas Gráficas.

A continuación, veremos diversas gráficas realizadas con la librería “Matplotlib” usando Python.

2.2.1 Histograma.

Para este apartado se ha creado un nuevo tipo de diagrama. Este es el denominado como “Histograma”. Un histograma es la representación gráfica en forma de barras, que simboliza la distribución de un conjunto de datos. Sirven para obtener una "primera vista" general, o panorama, de la distribución de la población, o de la muestra, respecto a una característica, cuantitativa y continua.
En cuanto al programa generado se optado por realizar dos tipos de Histograma. Estos los podremos ver en los puntos siguientes.

2.2.1.1 Histograma único.

Esta representación está formada por un único histograma con todos los datos del archivo. Para realizar esto, se han seguido los siguientes pasos:

- Añadir este diagrama a la lista que se le pide al usuario.
- Crear una nueva clase.
- Llamar a la función de este diagrama con el valor del eje Y pedido con anterioridad.

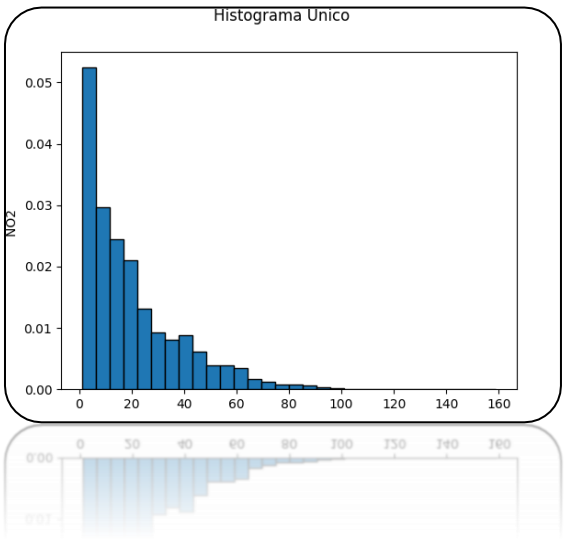
Podremos ver lo realizado con anterioridad a continuación:

Código

```
class HistogramaUnico(Grafica):
  def grafica(self, df):
    fig = plt.figure()
    fig.suptitle('Histograma Unico')
    plt.ylabel(self.nombreElementoY)
    plt.hist(self.Y, density=True, bins=30, alpha=1, edgecolor = 'black',  linewidth=1)
    if self.nombreFichero:
      plt.savefig(self.nombreFichero)
    else:
      plt.show()
```

Gráfica

Se ha creado un Histograma a partir de los datos de Tomé Cano. Para ello, se ha escogido el valor NO2.



2.2.1.2 Histograma múltiple.

Esta representación está formada por varios Histogramas para cada elemento seleccionado que se unen en un único. Esta representación, es muy útil para comparar gráficos. Para realizar esto, se han seguido los siguientes pasos:

- Añadir este diagrama a la lista que se le pide al usuario.
- Crear una nueva clase.
- Crear un vector único para cada elemento del eje x. Estos vectores se introducirán dentro de otro general, creando una estructura de vector de vectores.
- Llamar a la función de este diagrama con el valor calculado con anterioridad.

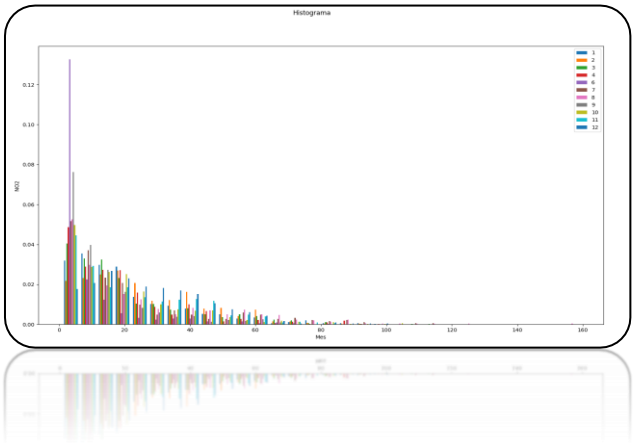
Podremos ver lo realizado con anterioridad a continuación:

Código

```
class HistogramaMultiple(Grafica):
  def grafica(self, df):
    fig = plt.figure()
    fig.suptitle('Histograma')
    plt.xlabel(self.nombreElementoX)
    plt.ylabel(self.nombreElementoY)
    listFinal = list()
    listProvisional = list()
    elements = dict.fromkeys(self.X).keys()
    for element in elements:
      listProvisional.append(element)
      elementosEjeX = df[self.nombreElementoX].isin(listProvisional)
      listFinal.append(df[elementosEjeX].loc[:,self.nombreElementoY])
      listProvisional = list()
    plt.hist(listFinal, density=True, bins=30, label=elements)
    plt.legend()
    if self.nombreFichero:
      plt.savefig(self.nombreFichero)
    else:
      plt.show()
```

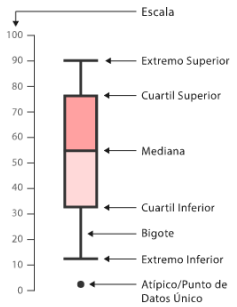
Gráfica

Se ha creado un gráfico que dispone de varios Histogramas cogiendo los datos de Tomé Cano. Para ello, se ha escogido el mes (como eje x) y NO2 (eje y).



2.2.2 Cajas y Bigotes.

Para este apartado, se ha creado un nuevo tipo de diagrama, denominado diagrama de cajas y bigotes. Un diagrama de cajas y bigotes es una manera conveniente de mostrar visualmente grupos de datos numéricos a través de sus cuartiles. Esto lo podremos ver a continuación:



Para poder realizar esta representación dentro de nuestro programa, se han seguido los siguientes pasos:

- Añadir este diagrama a la lista que se le pide al usuario.
- Crear una nueva clase.
- Crear un vector único para cada elemento del eje x. Estos vectores se introducirán dentro de otro general, creando una estructura de vector de vectores.
- Llamar a la función de este diagrama con los valores calculados.

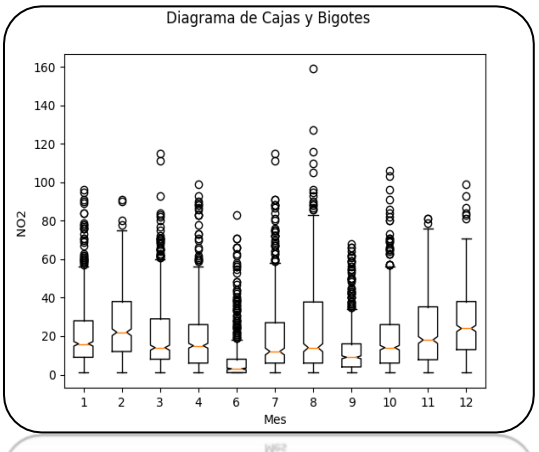
Podremos ver lo realizado con anterioridad a continuación:

Código

```
class Cajas(Grafica):
  def grafica(self, df):
    fig = plt.figure()
    fig.suptitle('Cajas y Bigotes')
    plt.xlabel(self.nombreElementoX)
    plt.ylabel(self.nombreElementoY)
    listFinal = list()
    listProvisional = list()
    elements = dict.fromkeys(self.X).keys()
    for element in elements:
      listProvisional.append(element)
      elementosEjeX = df[self.nombreElementoX].isin(listProvisional)
      listFinal.append(df[elementosEjeX].loc[:,self.nombreElementoY])
      listProvisional = list()
    plt.boxplot(listFinal, notch=True, sym="o", labels=elements)
    if self.nombreFichero:
      plt.savefig(self.nombreFichero)
    else:
      plt.show()
```

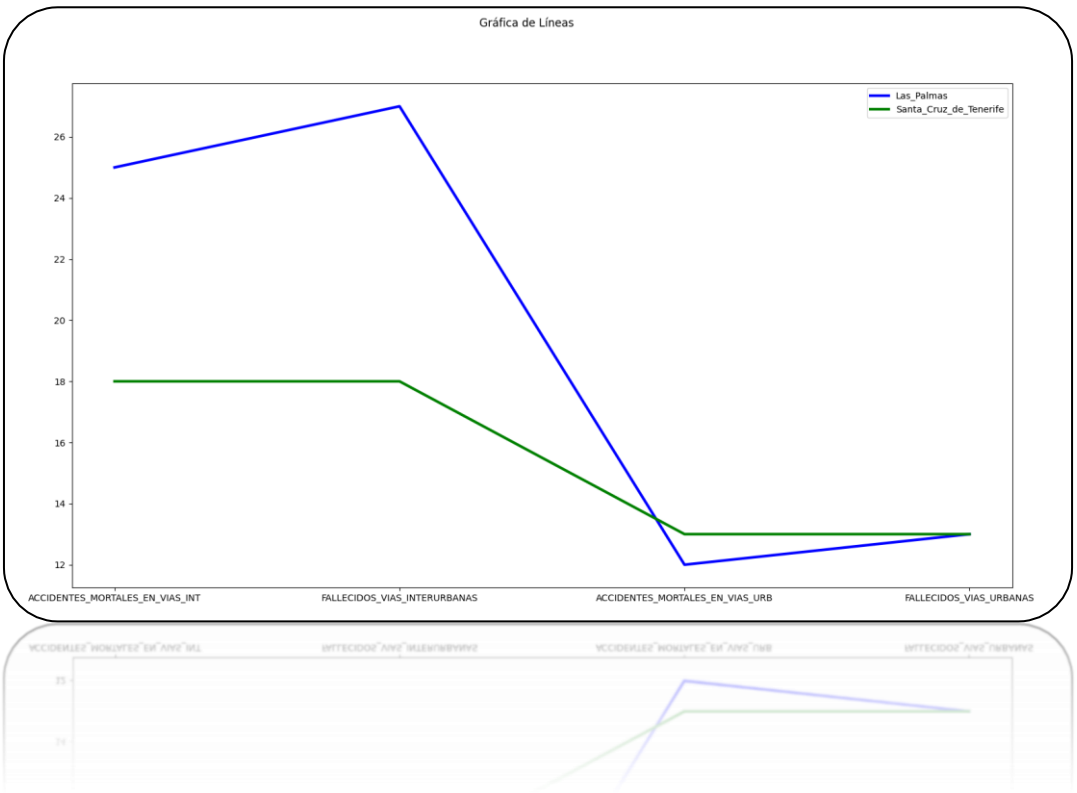
Ejemplo

Se ha creado un diagrama de cajas y bigotes usando la fuente de datos de los datos meteorológicos de la zona de Tomé Cano. De igual forma, se ha escogido el mes (eje x) y “NO2” (eje y).



2.3 Nueva fuente de datos.

Para este punto se ha realizado una búsqueda exhaustiva de datos para buscar unos interesantes para poder sacar conclusiones reales. Por ello, se ha ingresado en la página "https://opendata.esri.es/" y se escogido los datos de accidentes de tráfico en España durante el 2016. Posteriormente, se han probado usando un diagrama de líneas para comparar entre Tenerife y Las Palmas. Este lo podremos ver a continuación:



3.- Mejoras propuestas por el alumno.

3.1 Gestión de grupos.

Esta aportación consiste en la posibilidad de agrupar por un elemento de la tabla y poder realizar diversas acciones de interés. Un ejemplo de esto, es la posibilidad de agrupar por sexo (dentro de los datos del coronavirus) o por mes (en el diagrama de los datos de la calidad del aire). Sin embargo, a la hora de agrupar existen diversas posibilidades, es decir, ¿Qué hacemos a la hora de agrupar? Por ello, se ha propuesto las siguientes acciones:

- **Suma**: sumar los elementos que coincidan. Por ejemplo, en caso de agrupar por sexo (en unos datos de muertes del coronavirus) sumaremos todas las muertes de las mujeres y del hombre por separado y, realizaremos el gráfico que el usuario nos ha pedido.
- **Máximo:** coger el valor más grande. Por ejemplo, en los datos de la calidad del aire si agrupamos los datos por día, podremos coger el valor máximo del día y, así podremos dictaminar si en ese día tenemos una buena o mala calidad del aire (según

la OMS se tiene que coger el valor más alto y comprobar si está dentro de unos valores predeterminado).

- **Mínimo**: coger el valor más pequeño. Por ejemplo, en los datos de la calidad del aire si agrupamos los datos por días, podremos determinar a qué horas tenemos la mejor calidad del aire y, así podremos dictaminar la mejor franja horaria para hacer deporte.

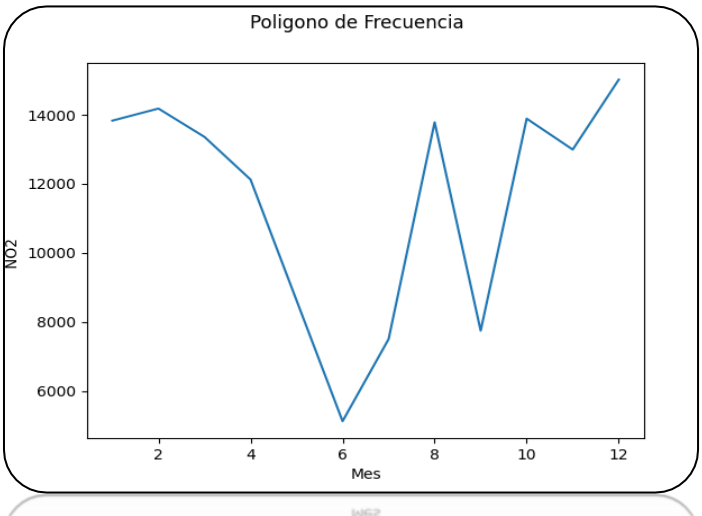
Esto lo podremos ver a continuación:

Código

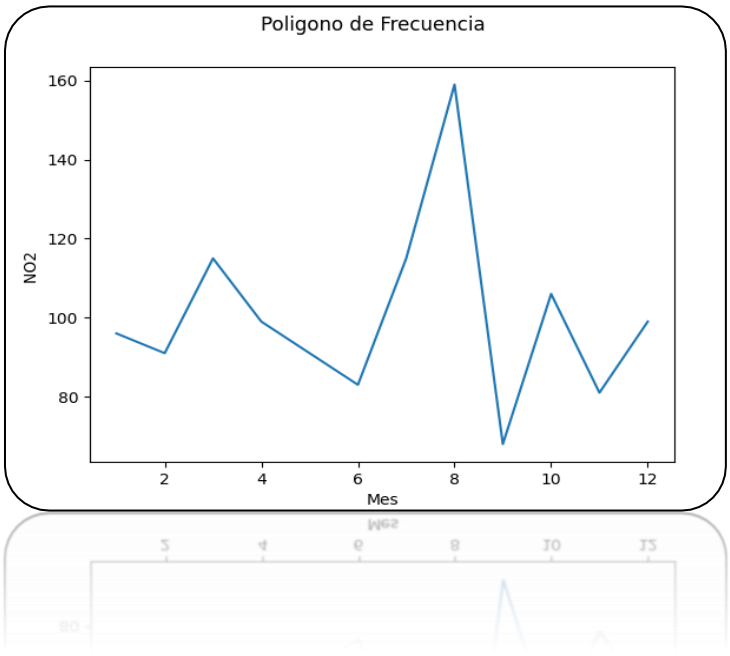
```
Indique que gráfica desea ver:
        1. Gráfica de Líneas.
        2. Gráfica de Barras.
        3. Gráfica de puntos.
        4. Gráfico Circular.
        5. Gráfico de Escaleras.
        6. Gráfico de Dispersión.
        7. Poligono de Frecuencia.
        8. Resumen.
 > 2
Los valores a seleccionar para los ejes son:

['Dia', 'Mes', 'Anio', 'Hora', 'SO2 ', 'NO ', 'PM10', 'O3 ', 'CO mg/m3', 'Benceno', 'Tolueno', 'Xileno', 'NO2 ']

Indique el valor numérico del eje X
 > 1
Indique el valor numérico del eje Y
 > 12
Indique el elemento por el que desea agrupar. Si no desea agrupar clicke enter
 > 1
En caso de colisión de datos similares al agrupar ¿Que desea hacer?:
        1. Suma.
        2. Máximo.
        3. Mínimo.
        4. Ninguno.
```

Gráfica (datos meteorológicos Tomé Cano)

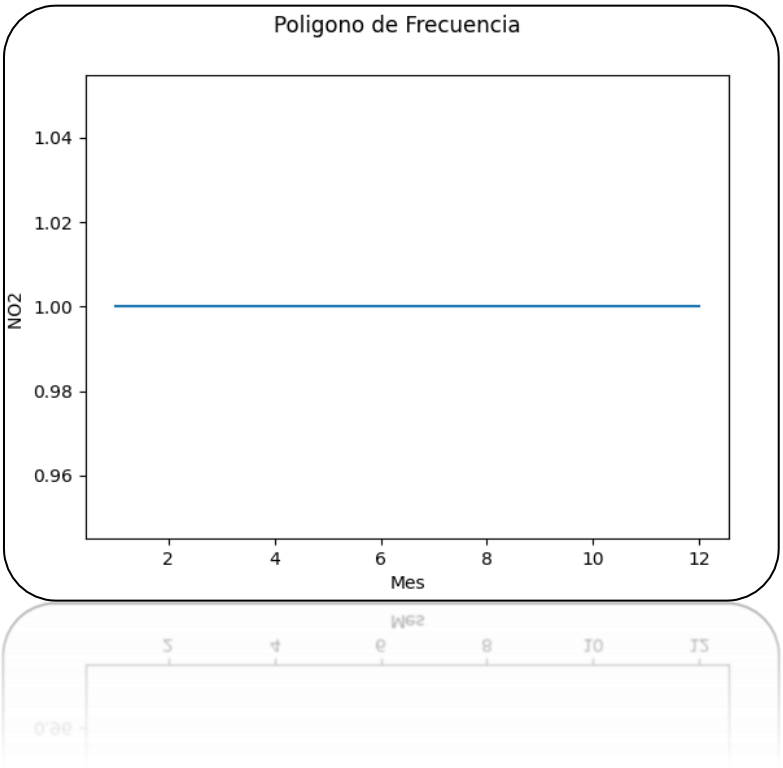
Agrupando por Suma



Agrupando por Máximo



Agrupando por Mínimo



4.- Librería Seaborn.

4.1 Explicación librería.

Seaborn es una librería de Python que nos permite generar gráficos de forma muy sencilla. Esta, está basada en Matplotlib y, por tanto, proporciona una interfaz de más alto nivel, y por consiguiente, resulta más fácil de usar.

Para saber qué gráficos podremos implementar con esta librería bastaría con entrar en la siguiente URL: "https://seaborn.pydata.org/examples/index.html" . Algunos ejemplos los podemos ver a continuación:

4.2 Gráficas implementadas.

Para poder entender mejor esta librería, ¿Qué mejor que probarla? Así que a continuación veremos cómo se ha implementado 3 gráficas (Histograma, Cajas y Bigotes y Diagrama de Violín).

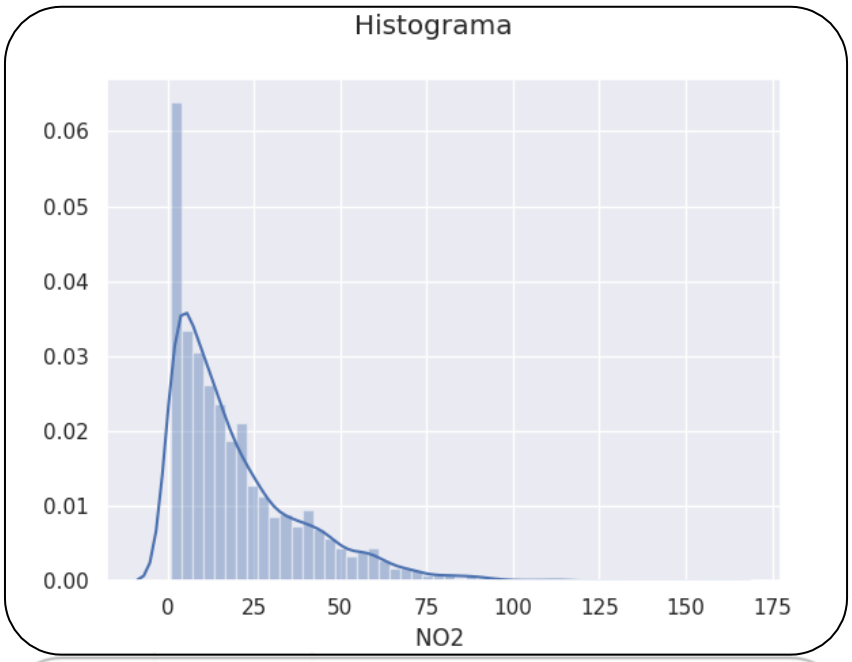
4.2.1 Histograma.

Aunque ya se ha implementado un histograma usando la librería “Matplotlib”, se ha querido crear este tipo de diagrama con la librería “Seaborn” y, así poder comparar de mejor forma el resultado obtenido en ambas librerías. A continuación, podremos ver el código y un ejemplo de la implementación:

Código

```
class HistogramaSeaborn(Grafica):
  def grafica(self, df):
    sns.set()
    fig = plt.figure()
    fig.suptitle('Histograma')
    x = np.random.randn(100)
    print(df.loc[:,self.nombreElementoY])
    sns.distplot(df.loc[:,self.nombreElementoY])
    if self.nombreFichero:
      plt.savefig(self.nombreFichero)
    else:
      plt.show()
```

Ejemplo



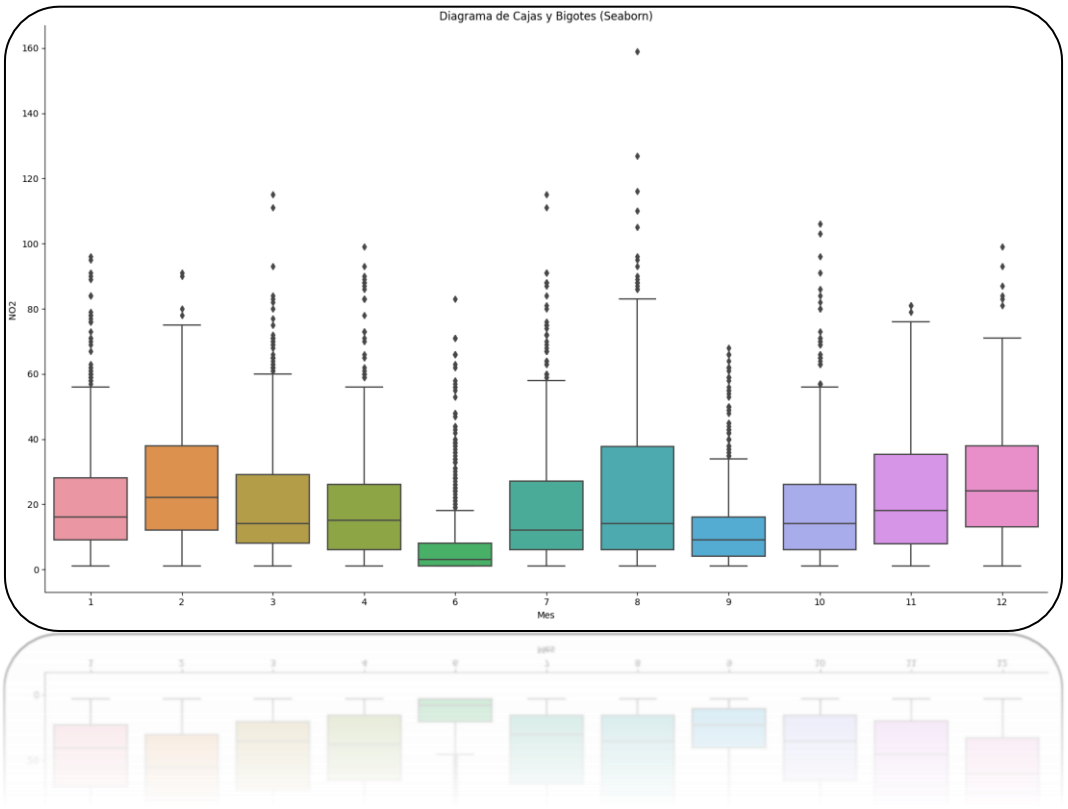
4.2.2 Cajas y Bigotes.

Aunque ya se ha implementado un histograma usando la librería “Matplotlib”, se ha querido crear este tipo de diagrama con la librería “Seaborn” y, así poder comparar de mejor forma el resultado obtenido en ambas librerías. A continuación podremos ver el código y un ejemplo de la implementación:

Código

```
class CajasSeaborn(Grafica):
  def grafica(self, df):
    sns.catplot(x = self.nombreElementoX, y = self.nombreElementoY, data=df, kind = "box").set(title = 'Diagrama de Cajas y Bigotes (Seaborn)')
    if self.nombreFichero:
      plt.savefig(self.nombreFichero)
    else:
      plt.show()
```

Ejemplo



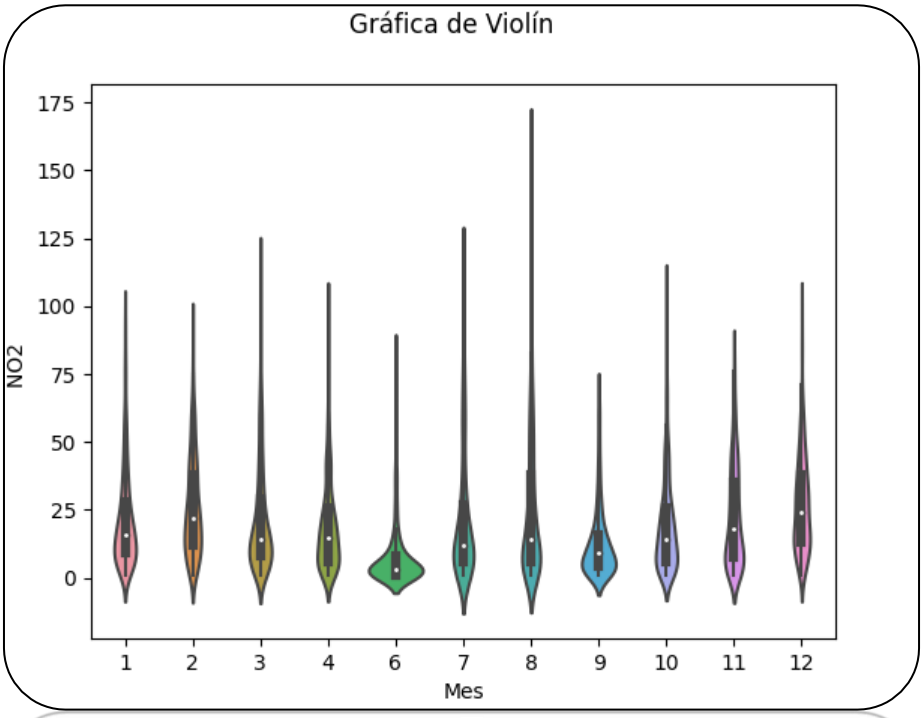
4.2.3 Diagrama de Violín.

Un diagrama de violín se utiliza para visualizar la distribución de los datos y su densidad de probabilidad. Este gráfico es una combinación de un diagrama de cajas y bigotes y un diagrama de densidad girado y colocado a cada lado, para mostrar la forma de distribución de los datos. La barra negra gruesa en el centro representa el intervalo Inter cuartil, la barra negra fina que se extiende desde ella, representa el 95 % de los intervalos de confianza, y el punto blanco es la mediana. A continuación, podremos ver el código y un ejemplo de la implementación:

Código

```
class ViolinSeaborn(Grafica):
  def grafica(self, df):
    fig = plt.figure()
    fig.suptitle('Gráfica de Violín')
    sns.violinplot(x = self.nombreElementoX, y = self.nombreElementoY, data=df)
    if self.nombreFichero:
      plt.savefig(self.nombreFichero)
    else:
      plt.show()
```

Ejemplo



5.- Extracción de conclusiones.

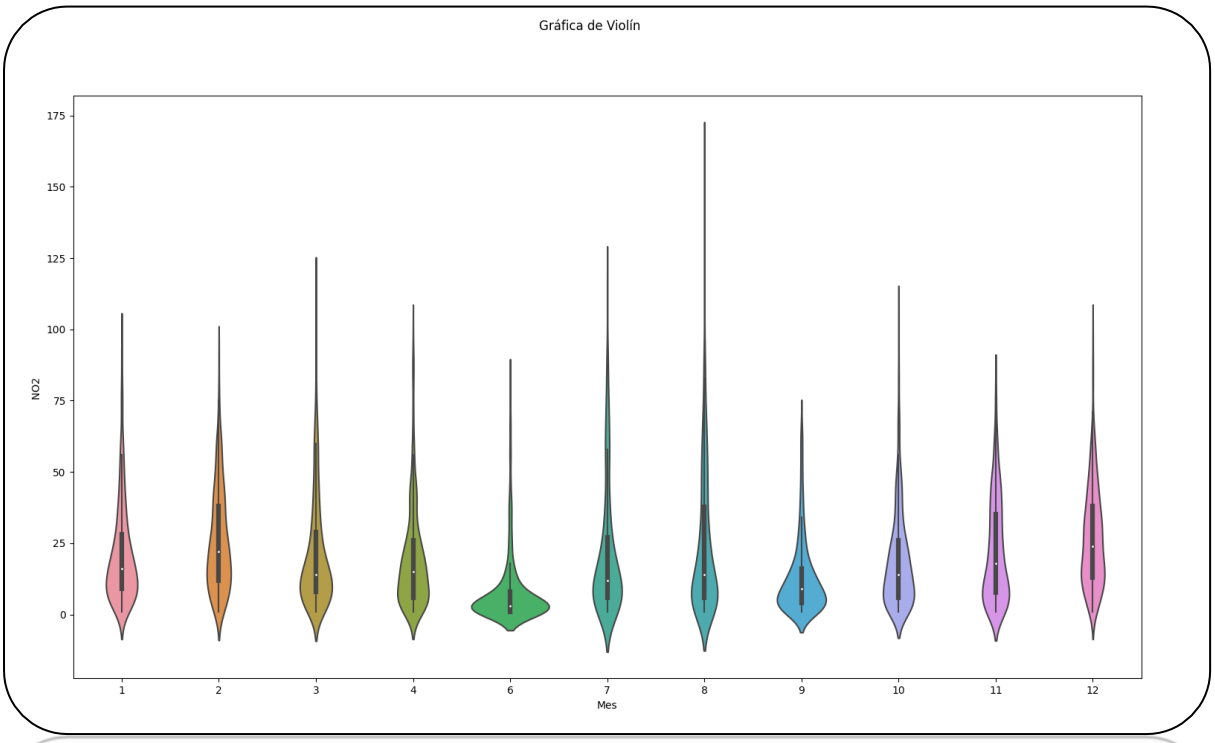
En este apartado, intentaremos sacar conclusiones de los datos buscados : Tomé Cano, Coronavirus y Accidentes de España. Esto lo podremos ver a continuación.

5.1 Tomé Cano (Tenerife)

Estos datos representan la calidad del aire en la zona de Tomé Cano (municipio de Tenerife). En ellos, entraremos mediaciones de diversos apartados relacionados con la calidad del aire. Estos son:

- Día.
- Mes
- Año.
- Hora.
- SO2.
- NO
- PM10.
- O3.
- CO
- Benceno.
- Tolueno.
- Xileno.
- NO2.

Posteriormente, hemos realizado un diagrama de Violín cogiendo como valor principal el Mes y el NO2 (valor más representativo de la calidad de aire). La gráfica generada ha sido:

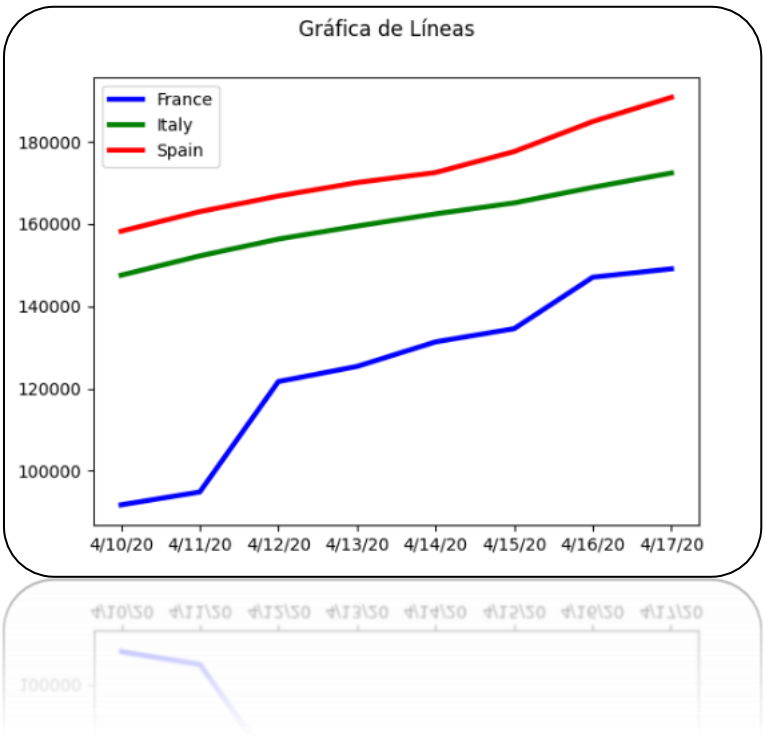


Como podemos ver en la gráfica anterior, podemos llegar a las siguientes conclusiones:

- El mes con mejor calidad del aire es junio. No obstante, el mes de septiembre también dispone de una buena calidad del aire.
- El mes con peor calidad del aire es diciembre. No obstante, tanto noviembre como febrero disponen de esta misma característica.
- El mes que dispone de peor calidad del aire en único día es septiembre. No obstante, Julio y marzo también poseen días cercanos a esta marca.

5.2 Coronavirus.

Para este apartado, se ha cogido el fichero del Coronavirus propuesto por el profesor. En él, disponemos de la cantidad de infectados en diferentes países. Por ello, compararemos los casos en España, Italia y Francia en un intervalo de fechas. El gráfico generado lo podremos ver a continuación:

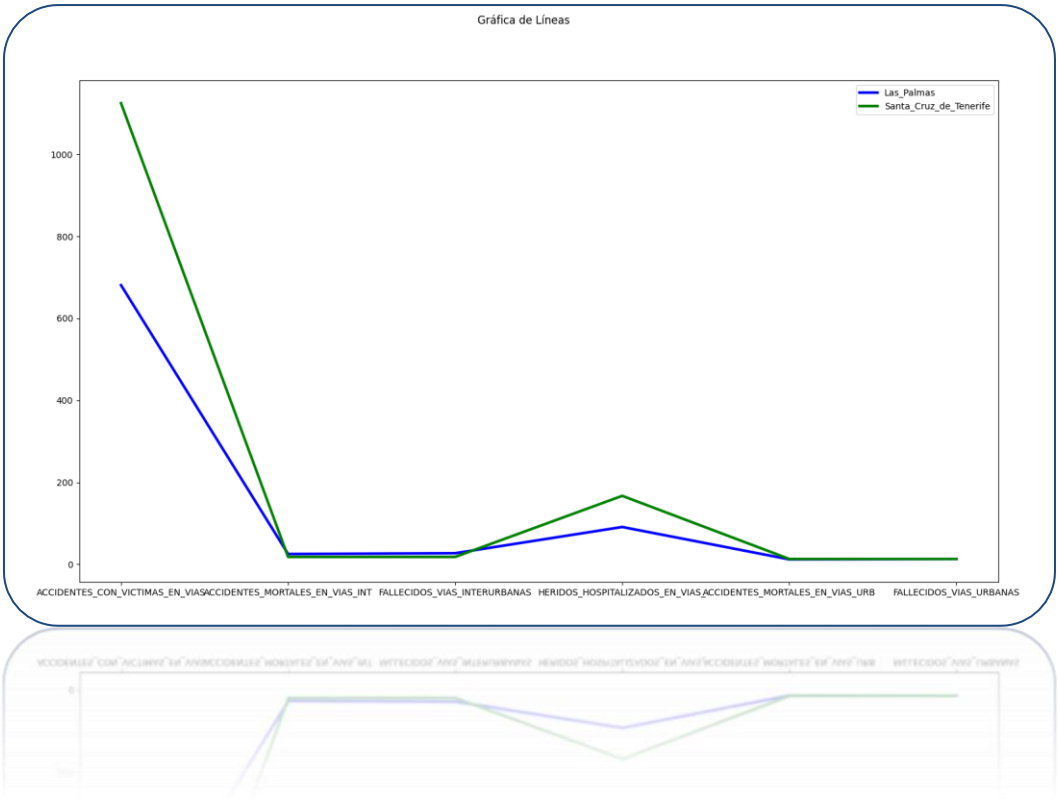


Como podemos ver en el gráfico anterior, España tiene más casos en total en dichas fechas. No obstante, Francia dispone de un crecimiento mucho más alto, lo cual nos hace pensar que podría superarla en un futuro no muy lejano.

5.3 Accidentes España.

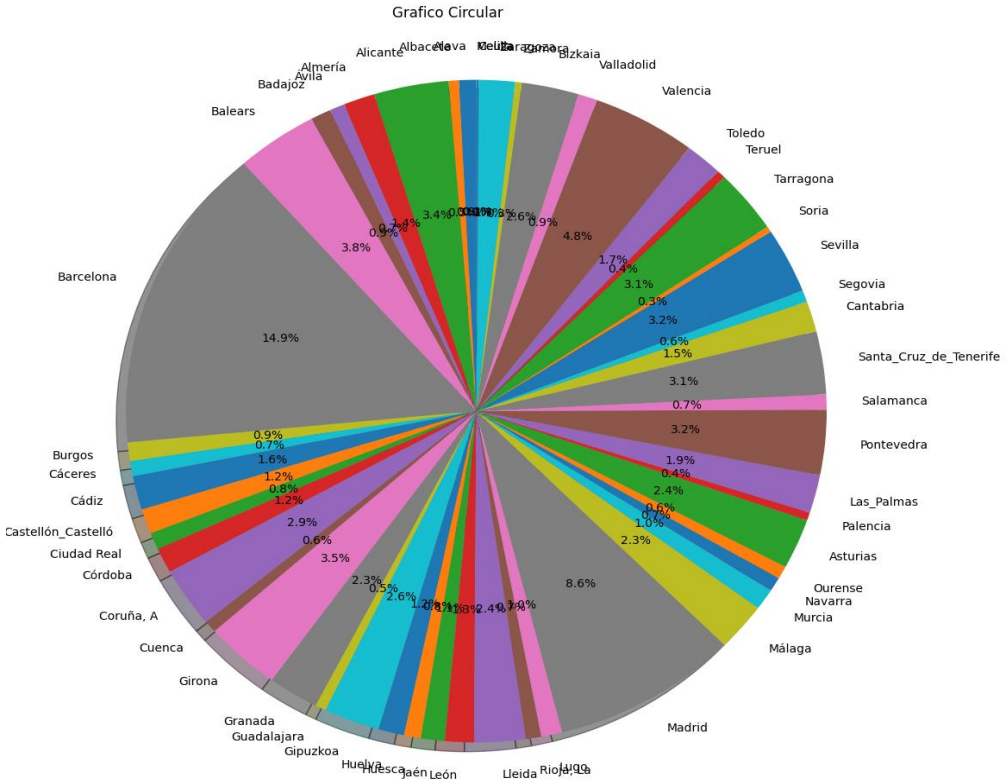
Para este apartado, usaremos los datos de accidentes de España en el año 2016. Estos datos son proporcionados por la DGT (fuente oficial).

Para este ejercicio, compararemos los accidentes en la provincia de Santa Cruz de Tenerife y Las Palmas. Por ello, realizaremos un diagrama de barras comparativo entre ambas localidades. Este lo veremos a continuación:



Como podemos ver en el gráfico anterior, en Tenerife hubieron menos accidentes de tráfico y, por consiguiente, menos fallecidos y heridos en estos.

De igual forma, se ha realizado un gráfico circular para comparar todas las localidades de España. Este lo podremos ver a continuación:

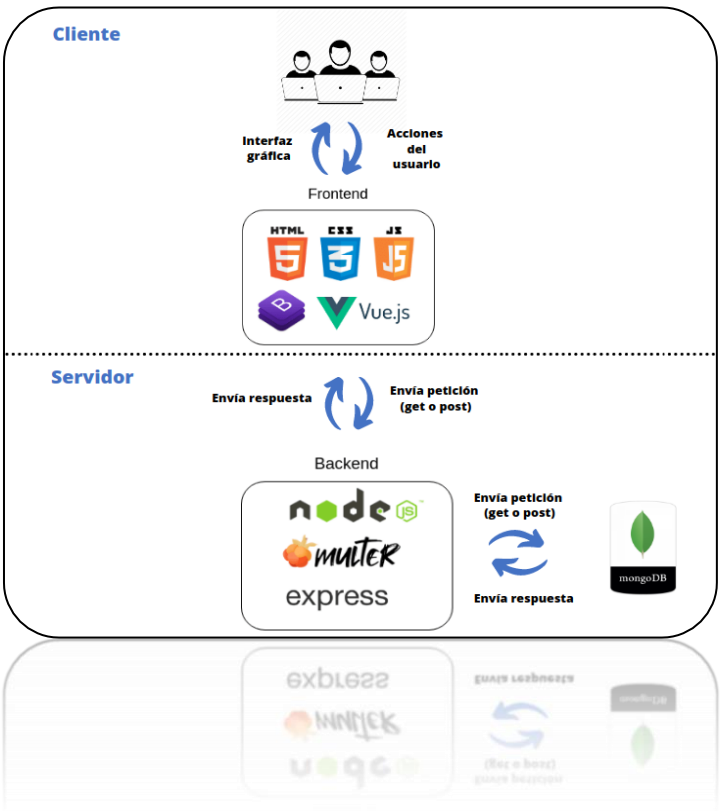


Como vemos en el gráfico, Barcelona sería la ciudad con más accidentes, le sigue de cerca Madrid, conclusión (que en un principio) veríamos lógica debido a la cantidad de personas que viven en estas provincias. No obstante, como podemos ver las Islas Baleares (3,8 %) estaría en el puesto número 4 (seguido de las mencionadas y por Valencia), dato ilógico si pensamos que los tienen un número muy bajo de habitantes (con respecto al resto de provincias). Una posible explicación de esto es la cantidad de turistas que las visitan al año.

6.- Servidor Computación en la nube.

6.1 Arquitectura.

Para el despliegue del servidor realizado en la asignatura de Computación en la Nube se ha utilizado la siguiente arquitectura:



Tras analizar detenidamente la cantidad de software existente, se ha escogido el que nos proporciona MEVN, esta elección se debe a que nos aporta muchas ventajas, entre las que cabe destacar: presentar previa experiencia en esta y su sistema de componentes. Este stack está compuesto de:

- **Mongodb:** base de datos de tipo noSQL que nos posibilita el manejo de grandes volúmenes de datos gracias a la tecnología de JSON.
- **Express:** framework que nos permite facilitar la creación de API REST, puesto que contiene muchas de las funcionalidades de estas y otras aportaciones que nos benefician.

- **Vue.js:** framework de desarrollo de la una página web que implementa la idea de usar Javascript en prácticamente todo el proyecto.
- **Node.js:** entorno de programación del lado del servidor que nos proporciona la posibilidad de mostrar una página web en el navegador.

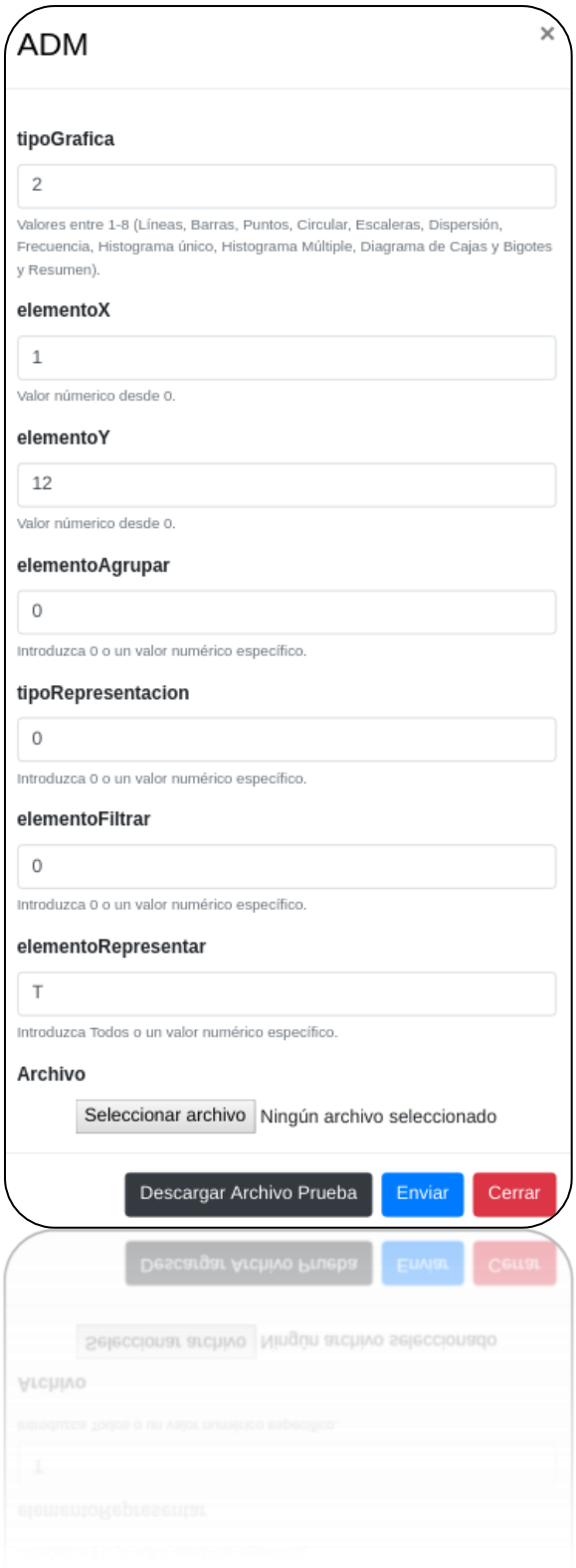
6.2 Mejoras.

6.2.1 Ejecución de ejemplo para los algoritmos.

Para este apartado, se ha incorporado la posibilidad de crear un ejemplo para el algoritmo dado en el proyecto de Computación en la Nube. Para ello, se ha modificado el fichero JSON añadiendo el nombre del fichero de ejemplo y los valores para cada campo. Esto lo podremos ver a continuación:


```
{
  "Name":"ADM",
  "file":"TomeCano.csv",
  "Description":"Método que permite recoger una gráfica a través de un fichero '.csv' dado con anteriornidad.",
  "Elements":[
    {
      "Name":"tipoGrafica",
      "value":"2",
      "Description":"Valores entre 1-8 (Líneas, Barras, Puntos, Circular, Escaleras, Dispersión, Frecuencia, H
    },
    {
      "Name":"elementoX",
      "value":"1",
      "Description":"Valor número desde 0."
    },
    {
      "Name":"elementoY",
      "value":"12",
      "Description":"Valor número desde 0."
    },
    {
      "Name":"elementoAgrupar",
      "value":"0",
      "Description":"Introduzca 0 o un valor numérico específico."
    },
    {
      "Name":"tipoRepresentacion",
      "value":"0",
      "Description":"Introduzca 0 o un valor numérico específico."
    },
    {
      "Name":"elementoFiltrar",
      "value":"0",
      "Description":"Introduzca 0 o un valor numérico específico."
    },
    {
      "Name":"elementoRepresentar",
      "value":"T",
      "Description":"Introduzca Todos o un valor numérico específico."
    }
  ]
}
```

Posteriormente, se ha modificado el Frontend, recogiendo los valores impuesto por el propietario del algoritmo de forma predeterminada. A continuación, se ha añadido un botón que nos permite descargar el archivo de ejemplo para añadirlo a la ejecución. Un ejemplo de esto lo podremos ver seguidamente:

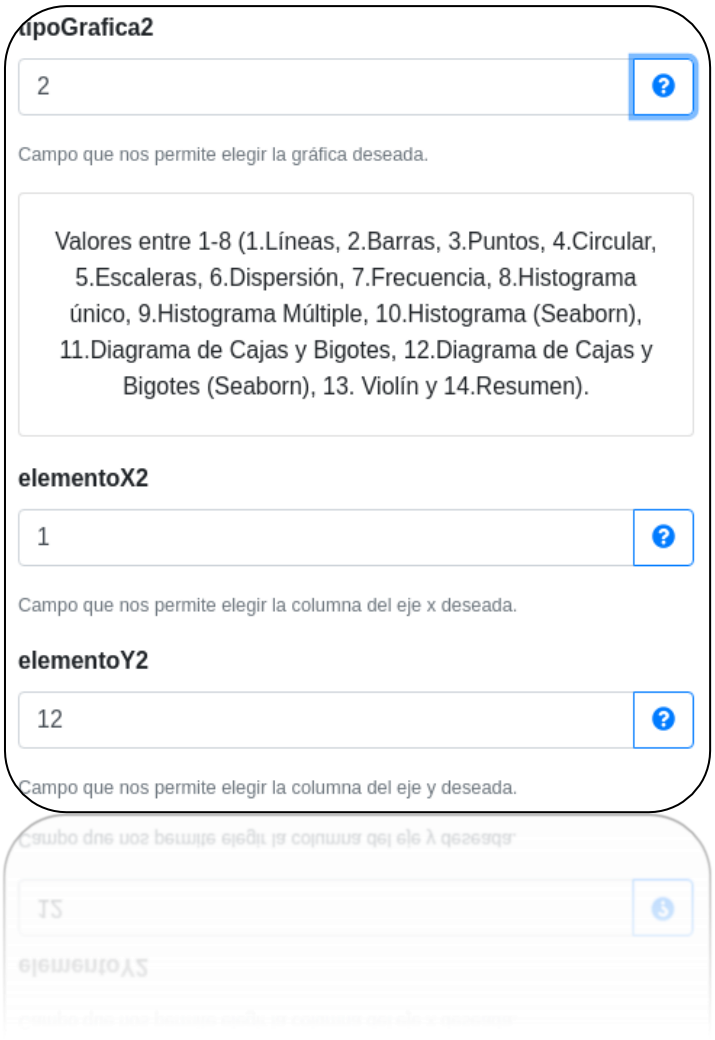


6.2.2 Mejora explicación de cada campo.

Este punto consiste en la posibilidad de añadir más información para un input de nuestro algoritmo. Para ello, tendremos que modificar nuestro fichero json añadiendo dos definiciones una corta (DescriptionShort) y otra larga (DescriptionLong). Esto lo podremos ver a continuación:


```
{
  "Name":"ADM",
  "file":"TomeCano.csv",
  "Description":"Método que permite recoger una gráfica a través de un fichero '.csv' dado
  con anteriornidad.",
  "Elements":[
    {
      "Name":"tipoGrafica",
      "value":"2",
      "DescriptionShort":"Campo que nos permite elegir la gráfica deseada.",
      "DescriptionLong":"Valores entre 1-8 (1.Líneas, 2.Barras, 3.Puntos, 4.Circular,
      5.Escaleras, 6.Dispersión, 7.Frecuencia, 8.Histograma único, 9.Histograma
      Múltiple, 10.Histograma (Seaborn), 11.Diagrama de Cajas y Bigotes, 12.Diagrama de
      Cajas y Bigotes (Seaborn), 13. Violín y 14.Resumen)."
    },
    {
      "Name":"elementoX",
      "value":"1",
      "DescriptionShort":"Campo que nos permite elegir la columna del eje x deseada.",
      "DescriptionLong":"Valor número empezando por 0 de la columna desea para nuestra
      representación gráfica."
    },
    {
      "Name":"elementoY",
      "value":"12",
      "DescriptionShort":"Campo que nos permite elegir la columna del eje y deseada.",
      "DescriptionLong":"Valor número empezando por 0 de la columna desea para nuestra
      representación gráfica."
    },
    {
      "Name":"elementoAgrupar",
      "value":"0",
      "DescriptionShort":"Campo que nos permite elegir la columna de agrupación.",
      "DescriptionLong":"Valor número emepzando por 0 de la columna desea para nuestra
      agrupación. Esto nos permitirá agrupar datos si están repetidos."
    },
    {
      "Name":"tipoRepresentacion",
      "value":"0",
      "DescriptionShort":"Campo que nos permite elegir el tipo de agrupación.",
      "DescriptionLong":"Valor número para nuestra agrupación. Las opciones son: 1.
      Suma, 2.Máximo, 3. Mínimo."
    },
    {
      "Name":"elementoFiltrar",
      "value":"0",
      "DescriptionShort":"Campo que nos permite filtrar.",
      "DescriptionLong":"Columna (valor numérico empezando por 0) que nos permite saber
      porque columna desea filtrar."
    },
    {
      "Name":"elementoRepresentar",
      "value":"T",
      "DescriptionShort":"Introduzca Todos o un valor numérico específico.",
      "DescriptionLong":"Valores separados por comas. Si no se requiere introduzca un
      0."
    }
  ]
```

Posteriormente, en el Frontend tendremos que tratar esta casuística añadiendo unos estilos apropiados. En este caso, se ha añadido un botón que al pulsarlo nos sale la información extendida. Un ejemplo de esto lo podremos ver a continuación:



7.- Visión de futuro.

En primera instancia, se ha hecho una búsqueda exhaustiva de diferentes aplicaciones relacionadas con este tipo de Framework. Esto se ha realizado para intentar diferenciarnos de lo que ya está hecho y, aportar una solución nueva al mercado.

Tras pensar detenidamente en las posibles soluciones que puede aportar esta herramienta. Considero que la mejor opción es transformarla en un servicio de análisis empresarial, es decir, a partir de una fuente de datos (base de datos, csv,..) poder conseguir aportar información de vital importancia sobre estos. No obstante, existen diversas soluciones ya creadas para este fin (Microsoft Power BI, Tableau,..), sin embargo, la idea principal es no solo enfocarse a los usuarios (como hacen estas) sino, también, a personal cualificado y poder aceptar código (realizado en Python u otro lenguaje) y añadirle de forma manual por cada usuario o empresa funcionalidades extras. De igual forma, estas funcionalidades extras podrían venderse dentro de la página web y, así crear una herramienta que pueda evolucionar añadiendo diferentes plugins y hacerla mucho más competitiva.

Sumado a lo comentado anteriormente, una buena opción es aportar a la herramienta dinamismo en la estética, es decir, una estética que pueda modificarse casi al 100% por el usuario o empresa. Esto provocaría que los usuarios estén más cómodos (con una estética personalizada) y, que parezca una herramienta realizada de forma exclusiva para ellos (aunque realmente no lo sea).

Por todo lo comentado, considero que este código podría derivar en una herramienta muy útil y que aporte mucho al sector del Business Intelligence. De igual forma, es necesario buscarle un nombre llamativo (como Dynamic BI) y, realizar una campaña de marketing para atraer a un mayor público.