



Universidad
Internacional
de Valencia

Cuadro de mandos con OpenData Euskadi

Titulación:

Máster Universitario en Desarrollo de
Aplicaciones y Servicios Web

Curso académico: 2022-2023

Alumno/a: Gómez Olivencia, Rubén

D.N.I.: 78910013-A

Asignatura: Análisis de datos Web

Índice general

1. Introducción	3
2. Contexto de uso	3
3. Datos obtenidos y análisis realizado	4
3.1. Fuente de datos	4
3.2. Análisis y transformación de datos	4
3.3. Modelo de datos resultante	7
4. Creación del cuadro de mandos	8
4.1. Carga de datos	8
4.2. Secciones	9
4.2.1. Conjunto completo de datos	9
4.2.2. Eventos esta semana	10
4.2.3. Futuros eventos	11
5. Conclusiones	12

1. Introducción

A la hora de realizar un análisis de datos (ya sean datos web, científicos, logs de servidores, meteorológicos, ...) no sólo es importante el propio análisis realizado y la información obtenida, ya que también es importante cómo se muestra.

El aspecto que le damos a esa información, junto con la posibilidad de interactuar con ella, puede hacer que esa información llegue a más gente y que se entienda de mejor manera que si sólo es plasmada sin ningún tipo de cuidado.

En este documento se va a explicar cómo se ha realizado la obtención de datos de eventos culturales obtenidos de la web [OpenData](#) del Gobierno Vasco, el análisis y procesado realizado, y cómo se ha creado un cuadro de mandos con la información obtenida.

2. Contexto de uso

Hoy en día se suelen realizar muchos eventos culturales pero que quizá no conozcamos porque no se les ha dado la propaganda suficiente al ser un evento pequeño o por tratarse de un evento en una localidad que no es la nuestra.

Para evitar eso, desde el departamento de datos abiertos [Open Data Euskadi](#) han creado una API asociada a eventos culturales a la que se puede solicitar datos desde la siguiente [web](#).

Aunque existe una web para realizar consultas de los próximos eventos, no se pueden consultar datos anteriores a la fecha actual, a modo de histórico, o para tener un agregado de datos.

Es por eso que a través del [cuadro de mandos](#) que se ha realizado, se puede obtener dicha información, ya sea pasada, presente o futura (teniendo en cuenta los datos existentes).

3. Datos obtenidos y análisis realizado

A continuación se va a exponer todo el proceso de obtención de datos, el análisis realizado y las modificaciones que ha habido que efectuar a los datos.

3.1. Fuente de datos

Tal como se ha dicho previamente, la fuente de datos ha sido el portal [Open Data Euskadi](#), concretamente la sección creada para la [API de eventos](#).

Desde esta web se explica cómo realizar consultas a la API y para ello se ha hecho uso del comando `>_ curl` para realizar la siguiente consulta y la obtención bruta de datos.

</> Obtención de los datos

```
ruben@vega: ~$ curl -X GET \
  "https://api.euskadi.eus/culture/events/v1.0/events?_elements=6845" \
  -H "accept: application/json" > eventos.json
```

De esta manera hemos obtenido un fichero `eventos.json` con los 6845 registros que la API nos devuelve, a los que hay que realizar un análisis de lo obtenido.

3.2. Análisis y transformación de datos

Tras obtener los datos, es momento de realizar un análisis del fichero obtenido. Dado que es un fichero **json**, es fácilmente parseable mediante el lenguaje de programación [Python](#).

En primer lugar es obtener los items de los eventos, ya que el fichero devuelto nos añade información extra, con la siguiente estructura:

</> Estructura de datos original

```
{
  "totalItems": 6845,
  "totalPages": 1,
```

```
"currentPage": 1,
"items": [
  ...
]
}
```

Por ello, al abrir el fichero, tenemos que acceder al campo “items” y recorrerlos, para quedarnos con los índices que nos interesan y comenzar a realizar ciertas modificaciones necesarias. Para ello, se ha tenido que recorrer todos los items de la siguiente manera:

</> Parte del script de parseo de datos

```
#!/usr/bin/env python3
import pandas as pd
import json
import re
# ...
out = json.loads('[]')
# abrimos el fichero
with open('eventos.json', 'r', encoding='utf8') as fichero:
    # ...
    for item in items:
        o = json.loads('{}')
        o.update({"Nombre": re.sub('\\"', '', item['nameEs']).title())})
        o.update({"Tipo": item['typeEs']})
```

Dado que no nos interesan todos los índices de los datos obtenidos, se ha creado una variable “o” donde se irán añadiendo los índices, y renombrando los que sean necesarios.

Los dos primeros campos que vamos a usar son el nombre y el tipo de evento. Debido a que en los datos originales también hay versión en euskera, existen dos campos para el nombre del evento. Nos quedamos el de castellano y lo renombramos a “Nombre”. Esto ha habido que hacerlo también con otros campos.

En el campo del nombre, se ha aprovechado a quitar las comillas usando una

expresión regular, ya que varios registros hacían uso de ellas.

Por otro lado, también ha habido que realizar modificaciones en las fechas y con la posición de geolocalización:

</> Parte del script de parseo de datos

```
d = re.sub('T.*','',item['startDate'])
d = re.sub('-', '', d)
o.update({"Fecha inicio":d})
d = re.sub('T.*','',item['endDate'])
d = re.sub('-', '', d)
o.update({"Fecha fin":d})
if 'municipalityLatitude' in item and 'municipalityLongitude' in item:
    o.update({"GPS":item['municipalityLatitude']+', '+item['municipalityLongitude']})
```

En el caso de las fechas, dado que [Data Studio](#) no identificaba de manera correcta las fechas, ha habido que convertirlas al formato **20221116**, donde el año, mes y día está todo junto y en ese orden.

En lo que se refiere a la **geolocalización**, los datos originales contenían la latitud y la longitud por separado, por lo que se ha creado un índice juntando ambos datos, llamado **GPS**, para que después pueda ser utilizado.

También ha habido que realizar modificaciones en lo que se refiere a la provincia, ya que en los datos no se obtenía el nombre, sino que aparecía el código de la misma. En la propia API hay un apartado donde se puede obtener una relación del código con el nombre. Con esa información se ha decidido realizar el siguiente código para hacer los cambios necesarios:

</> Parte del script de parseo de datos

```
if item['provinceNoraCode'] == '48':
    o.update({'Provincia': 'Bizkaia'})
if item['provinceNoraCode'] == '1':
    o.update({'Provincia': 'Araba'})
if item['provinceNoraCode'] == '20':
    o.update({'Provincia': 'Gipuzkoa'})
if item['provinceNoraCode'] == '31':
```


```
o.update({'Provincia': 'Navarra'})
if item['provinceNoraCode'] == '-3':
    o.update({'Provincia': 'Iparralde'})
```

Ha habido algún otro cambio que ha habido que realizar, pero debido a que son similares a los casos expuestos previamente, se consideran irrelevantes en el ámbito general.

Para finalizar con los datos, cada evento con los datos elegidos y transformados, se ha añadido a un array (llamado “out”) y que posteriormente se ha convertido a formato **CSV** a través de la librería [Pandas](#).

 Parte del script de parseo de datos

```
df = pd.read_json(json.dumps(out), dtype=True)
df.to_csv('eventos.csv', index=False)
```

De esta manera tenemos un nuevo fichero  **eventos.csv** que podemos cargar como fuente de datos en [Data Studio](#).

3.3. Modelo de datos resultante

Tras la transformación realizada en el paso previo, el modelo resultante de datos (que ha sido exportado a formato CSV) es el siguiente:

- **Nombre:** Nombre del evento en formato texto.
- **Tipo:** Tipo de evento, en formato texto.
- **Fecha inicio:** Fecha de comienzo del evento, en el formato de fecha admitido por Data Studio.
- **Fecha fin:** Fecha de finalización del evento, en el formato de fecha admitido por Data Studio.
- **GPS:** La latitud y la longitud de la geolocalización del evento.
- **Municipio:** El municipio donde se celebra el evento, en formato texto.
- **Idioma:** En caso de que exista el idioma, aparece en formato ISO (EU, ES, ...)

- **Precio:** Dado que en los datos originales es un campo de texto, donde no se sigue una estructura fija para poder ser parseado, se mantiene en formato texto.
- **Provincia:** La provincia donde se realiza el evento.

Para finalizar, el fichero ha sido importado en [Data Studio](#) para poder crear el cuadro de mandos.

4. Creación del cuadro de mandos

Para crear el [cuadro de mandos](#) se ha hecho uso de la aplicación [Data Studio](#) de Google, ya que nos va a permitir realizar gráficas y mostrar la información de manera sencilla, pero con gran versatilidad.

4.1. Carga de datos

Como ya se ha dicho previamente, el último paso ha sido generar un fichero CSV ya que es uno de los formatos aceptados por la aplicación, y que ha sido añadido como fuente de datos.

Tras la carga, y haber indicado que el índice GPS sirve para geolocalización, obtenemos el siguiente resultado:

DIMENSIONES (9)			
Fecha fin	⋮		Fecha
Fecha inicio	⋮		Fecha
GPS	⋮		Latitud, longitud
Idioma	⋮	ABC	Texto
Municipio	⋮	ABC	Texto
Nombre	⋮	ABC	Texto
Precio	⋮	ABC	Texto
Provincia	⋮	ABC	Texto
Tipo	⋮	ABC	Texto

4.2. Secciones

El cuadro de mandos realizado se ha dividido en distintas secciones, con sus correspondientes gráficas que pasaremos a detallar a continuación.

4.2.1. Conjunto completo de datos

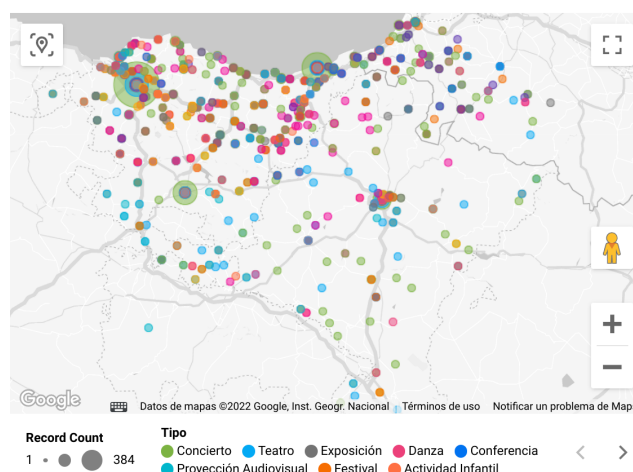
La primera sección se hace uso del conjunto completo de datos, en el que se puede ver el total de los datos, elegir por el tipo de evento, la provincia donde se realiza o elegir entre las fechas en las que se realiza.

En caso de realizar alguna modificación en estos controles, se verá reflejado en la tabla que hay a continuación, así como en las gráficas creadas.

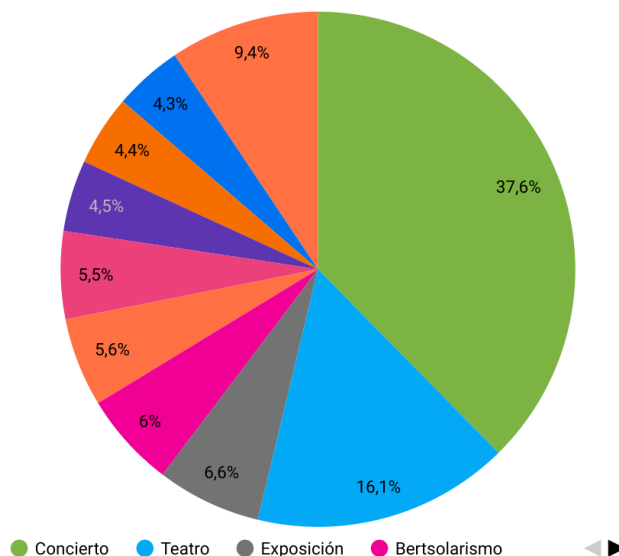


Las gráficas creadas representan:

- **Geoposición de los eventos:** La posición geográfica de los eventos, diferenciando cada tipo de evento con un color distinto.



- **Porcentaje de tipos de eventos** en formato de círculo. En este caso se ve que el mayor porcentaje de eventos son conciertos.



- **Mapa de calor** de los eventos. Similar al mapa comentado previamente.
- **Gráfica de barras** donde aparecen los totales de eventos por territorio, teniendo en cuenta la selección que se haya realizado.

4.2.2. Eventos esta semana

Se ha decidido crear una sección donde los datos cuenten con un periodo predeterminado. Para que las gráficas resultantes sólo muestren la cantidad de eventos que se van a realizar la semana en curso en la que nos encontramos, han sido configuradas indicando el periodo personalizado, para que muestren “esta semana”, y comenzando por lunes.

Periodo predeterminado

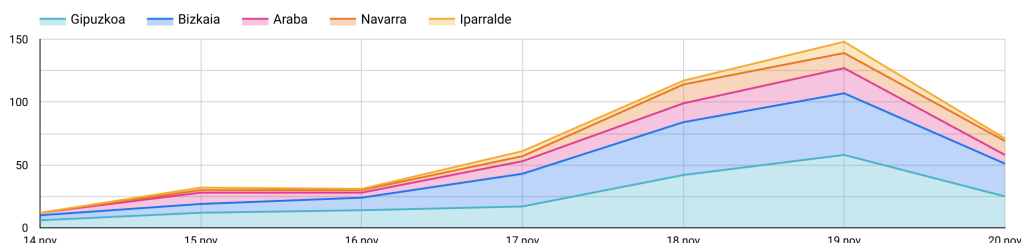
☐ Automático

☒ Personalizado

☐ Esta semana (empieza en lunes) ▼

Hay dos gráficas:

- **Eventos totales** por día de la semana.
- **Eventos diferenciando el territorio.** De esta manera, se puede elegir sólo los eventos del territorio que nos interese.



4.2.3. Futuros eventos

Para la última sección se ha decidido crear distintos filtros para que sólo aparezcan los eventos que están registrados, pero que se van a celebrar en el futuro. Aparte, se ha decidido diferenciar por idiomas mediante otro filtro.

Por tanto, se han creado tres filtros:

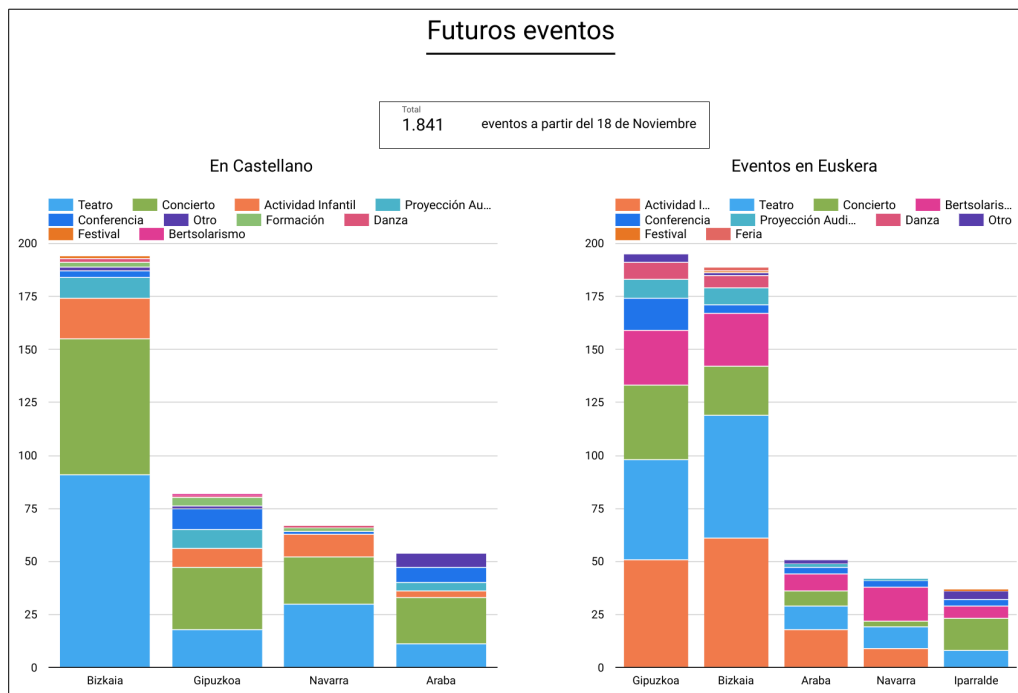
- **Filtro castellano:** Para la gráfica que sólo muestra eventos que se han especificado que van a ser en castellano.
- **Filtro Euskera:** Para la gráfica que sólo muestra eventos que se han especificado que van a ser en euskera.
- **Filtro de fecha futuros:** Que sirve para que las gráficas de esta sección sólo incluyan eventos cuya fecha de inicio sea mayor o igual que el 18 de Noviembre.

Filtros		
Nombre	Se utiliza en el informe	Descripción
Filtro Castellano	1 gráfico	Incluir Idioma Igual que (=) ES
Filtro Euskera	1 gráfico	Incluir Idioma Igual que (=) EU
Fechas futuros	3 gráficos	Incluir Fecha inicio Mayor o igual que (>=)

Poniendo como ejemplo el gráfico que muestra eventos futuros, y que van a ser en castellano, se le han configurado dos de los filtros mencionados previamente:



De esta manera, aplicando los filtros creados, cada uno en su correspondiente gráfica, la sección queda de la siguiente manera:



El tercer gráfico que hace uso del filtro “fechas futuros” es el cuadro de resultados, que muestra el total de eventos, en este caso 1841.

5. Conclusiones

A la hora de realizar un análisis de datos el tiempo se debería invertir en el procesado de los mismos: estudiar cuál es el conjunto de datos obtenidos, ver qué nos interesa de los mismos, realizar posibles transformaciones...

Si una vez dedicado ese esfuerzo, debemos dedicar otro tanto en cómo plasmar los resultados, utilizando sistemas tediosos, librerías javascript quizá no bien documentadas, y gestionando el servidor donde se van a alojar los datos, lo más seguro es que el resultado se resienta.

Es por eso que es importante conocer y hacer uso de aplicaciones, como Google DataStudio, que nos facilitan el crear cuadros de mando interactivos y con un aspecto cuidado, como el que hemos realizado para este documento. De esta manera podremos dedicar el esfuerzo en lo realmente importante: el análisis de los datos.