

Capítulo 5

Analíticas basadas en Elasticsearch y Kibana

En este capítulo se describen las nuevas tecnologías utilizadas en la versión final del módulo de analíticas que sustituyen a MongoDB y Matplotlib. Para enriquecer los datos que se estaban almacenando por las sondas del servidor, a esta nueva versión de la herramienta, se han añadido nuevas sondas y enriquecido las existentes. También se detallan la creación de recursos de prueba para que los futuros desarrolladores puedan integrar estas tecnologías en local.

5.1. Diseño

Para el desarrollo de las mejoras necesarias de la herramienta de analíticas, el primer paso es realizar un diseño con los cambios a implementar así como las nuevas tecnologías involucradas.

El primer prototipo estaba basado en MongoDB como base de datos de logs, y con Matplotlib como generador de visualizaciones. Pese a ser bastante completo, carece de cierta información útil como datos de visitantes a la aplicación o acerca del *software* y *hardware* utilizado por los usuarios. Además, debido a la necesidad de renderizar cada una de las imágenes en Python, el subsistema de analíticas era lento.

Es por esto, que se decide cambiar la tecnología de visualización de datos a Kibana. Al ser Kibana parte del stack ELK, la base de datos cambia a Elasticsearch para poder utilizar todo el

potencial que este ecosistema ofrece.

Para enriquecer los datos almacenados y evitar tener que cruzar datos como se detalla en la subsección 4.3.2, se van a añadir más sondas de almacenamiento de datos, esta vez con datos relativos a la actividad de los visitantes de la plataforma educativa. Además, se van a unificar las tablas en las que se almacenarán registros similares, evitando tener así que cruzar datos. Estos cambios reducirán el coste computacional de la herramienta haciéndola así más rápida.

El nuevo diseño y los cambios de tecnología involucrados en este prototipo final se pueden ver representados de forma esquemática en la Figura 5.1.

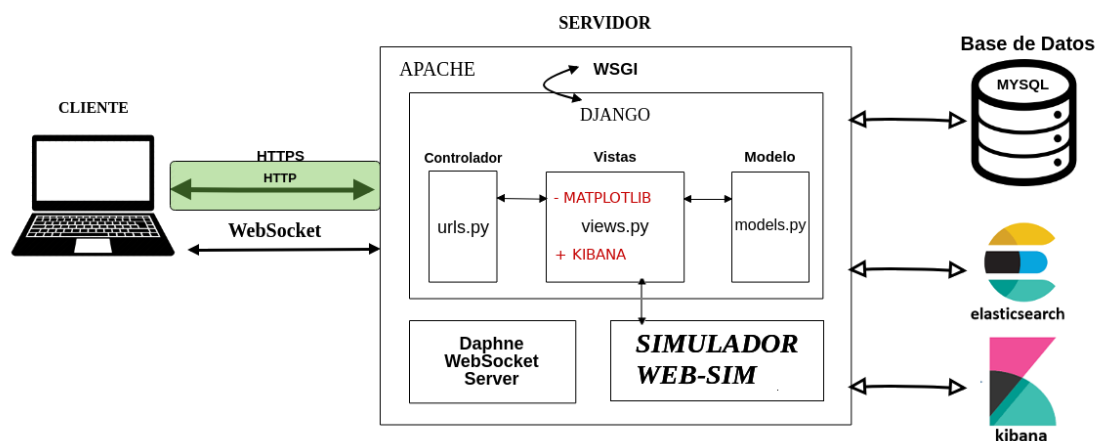


Figura 5.1: Arquitectura Kibotics segundo prototipo.

5.2. Implementación

En esta sección se detallan los cambios involucrados en el desarrollo de la herramienta de analíticas para la integración de Elasticsearch como base de datos y Kibana como herramienta de visualización. Además, de esta versión final de la herramienta, se explica la generación de recursos de prueba para futuros desarrolladores de Kibotics.

5.2.1. Elasticsearch en Webserver

El primer paso en el proceso de mejora de la herramienta es el cambio de base de datos. Para hacer uso del stack ELK, es necesario sustituir MongoDB por Elasticsearch. Para esto lo primero será instalar e iniciar un servicio de Elasticsearch en local ¹.

Una vez iniciado el servicio Elasticsearch, ya dispondremos de la base de datos sobre la que guardar los registros de log o documentos, como se llaman en Elasticsearch. Similar a las tablas de las que hacía uso MongoDB, estos documentos se almacenan en índices, cada uno de los cuales está definido por un esquema de campos y tipologías que definen la estructura de los documentos que almacenan.

Se podrán crear tantos índices como sea necesario, para consultarlos se puede lanzar una sentencia por terminal, o acceder con la IP y el puerto desde un navegador a la API REST configurada en instalación. Un ejemplo de llamada para un índice llamado `index_name_test` es la siguiente:

```
http://127.0.0.1:9200/index_name_test/_search/?size=1000&pretty
```

El siguiente paso es la integración de Elasticsearch en Django. Se realizará haciendo uso de la librería Python `django_elasticsearch_dsl`. Para migrar de MongoDB a Elasticsearch se han realizado dos pasos: creación de los índices y migración de las sondas que almacenan los logs de MongoDB a Elasticsearch.

El primer paso consistió en crear los índices de Elasticsearch. Estos índices son muy similares a los modelos que se utilizan en Django para representar la información de la base de datos. Para esta versión de la herramienta se crean los siguientes índices:

- `kibotics_session_log`: índice en el que se almacenan los eventos referentes a las sesiones.
- `kibotics_simulation_log`: índice en el que se almacena la información relativa a las simulaciones.

¹<https://www.elastic.co/guide/en/elasticsearch/reference/current/targz.html>

- `kibotics_error_log`: índice en el que se almacenan los eventos referentes a los errores.
- `kibotics_visit_log`: índice en el que se almacena la información relativa a las visitas.

Para evitar la problemática que surgió durante el desarrollo del primer prototipo, relativo al cálculo de la duración de los eventos, se ha eliminado el campo que almacenaba la fecha. Para sustituirlo, se han añadido dos nuevos campos a los índices de Elasticsearch los cuales establecerán el inicio y fin de cada evento, unificando así los dos registros de log que se tenían en la primera versión.

Por otro lado, el campo `USER_AGENT`, que se almacenaba anteriormente y ofrecía información acerca del dispositivo y software que utilizaban los visitantes de la web, ha sido dividido y sustituido con la información del navegador, dispositivo y sistema operativo. Cada uno con su propio campo en los esquemas de los índices de Elasticsearch. Ofreciendo así la información de forma más clara y eliminando ciertos datos no útiles para las visualizaciones que se quieren generar.

Para trabajar en Kibana con mapas geográficos es necesario guardar tanto la longitud como la latitud, para lo cual se hará uso de una tipología de campo ya existente en Elasticsearch llamado *Geo Point* que almacenará en un diccionario ambas variables.

Para complementar las nuevas sondas, que se han mencionado anteriormente y se explicarán con más detalle a continuación, es necesario la creación de un nuevo índice de visitas, en el que se registrarán los accesos a la página principal de la aplicación, estén o no registrados.

Un ejemplo de la definición en Django del índice de sesiones de Kibotcis Webserver en el fichero `documents.py` es:

```
from django_elasticsearch_dsl import Document, Text, Date, Double, GeoPoint, Ip

class SessionDocument(Document):
```

```
username = Text()
start_date = Date()
end_date = Date()
duration = Double()
client_ip = Ip()
browser = Text()
os = Text()
device = Text()
location = GeoPoint()

class Index:
    name = 'kibotics_session_log'
    settings = {
        'number_of_shards': 1,
        'number_of_replicas': 0
    }
```

Con todo esto, se tiene una base sólida sobre la que almacenar la información de logs, solo falta modificar el guardado en Elasticsearch, migrar las sondas que almacenan los registros de log.

Para enriquecer las sondas ya existentes y hacer uso de los nuevos campos e índices creados, se han añadido nuevas sondas para almacenar eventos de visitantes, así como sondas para optimizar la monitorización de salida de sesiones y simulaciones y evitar no tener un registro de inicio sin su correspondiente registro de fin de evento por un cierre busco de la aplicación o inactividad.

Las sondas de inicio de los eventos son similares a las que se tenían anteriormente en MongoDB, un ejemplo para la sonda de inicio de simulación es:

```
SimulationDocument(
    username = "USERNAME_TEST",
    start_date = start_date_object_test,
    end_date = start_date_object_test,
    duration = 0.0,
    client_ip = "CLIENT_IP_TEST",
```

```

simulation_type = "SIMULATION_TYPE_TEST",
exercise_id = "EXERCISE_ID_TEST",
browser = "BROWSER_TEST",
os = "OS_TEST",
device = "DEVICE_TEST",
location = {'lat': latitude_double_test, 'lon': longitude_double_test}
).save()

```

Sin embargo, al unificar los registros de entrada y salida en un único documento las sondas de fin de sesión/simulación cambian. Tendrán que buscar en Elasticsearch el último documento no finalizado en el índice para el usuario del cual se quiera cerrar el evento y sustituir tanto los campos `end_date` como `duration`.

Esto se realiza gracias al identificador que cada documento indexado posee al cual se le realiza una operación *update* con los nuevos campos. Un ejemplo de cierre de sesión en Python es:

```

# Búsqueda del ultimo registro de sesión no cerrado
latest_session = Search(index="kibotics_session_log*") \
    .query("match", username=username) \
    .query('match', duration=0) \
    .sort({"start_date": {'order': 'desc'}})[0]

# Actualización de los campos end_date y duration
for hit in latest_session:
    duration = datetime.now() - datetime.strptime( \
        hit.start_date, \
        "%Y-%m-%dT%H:%M:%S.%f")

    Elasticsearch(settings.ELASTICSEARCH_DSL['default']['hosts']) \
        .update(index = 'kibotics_session_log',
                id = hit.meta.id,
                body = {"doc": {
                    'end_date' : datetime.now(),
                    'duration' : duration.total_seconds()
                }}
        )

```

Durante el desarrollo de esta lógica es útil hacer uso de la interfaz de Elasticsearch para comprobar que todos estos campos están siendo creados y actualizados correctamente. A esta interfaz se accede mediante la URL y puerto configurados durante el proceso de instalación.

En esta API, podremos realizar filtrados por campos e índices haciendo uso de expresiones regulares *Regex*, por ejemplo para acceder al índice en el que se almacenan los documentos relativos a los logs de sesiones, podremos acceder mediante la siguiente URL:

```
http://127.0.0.1:9200/kibotics_session_log/_search/?size=1000&pretty
```

En la Figura 5.2 se puede observar la respuesta que se obtiene de la API de Elasticsearch en el navegador a la llamada anterior.

```
▼ 1:
  _index:      "kibotics_session_log"
  _type:       "_doc"
  _id:         "ccCT63EBdVY2vb3Ic8f8"
  _score:      1
  ▼ _source:
    username:   "student_dummy"
    start_date: "2020-01-29T06:00:00"
    end_date:   "2020-01-29T06:31:00"
    duration:   1860
    client_ip:  "161.185.160.93"
    browser:    "dummy_browser_5"
    os:         "dummy_os_5"
    device:     "dummy_device_5"
    ▼ location:
      lat:      40.7654
      lon:      -73.8174
▼ 2:
  _index:      "kibotics_session_log"
  _type:       "_doc"
  _id:         "a8CT63EBdVY2vb3Ic8f0"
  _score:      1
  ▼ _source:
    username:   "teacher_dummy"
    start_date: "2020-01-28T05:00:00"
    end_date:   "2020-01-28T05:32:00"
    duration:   1920
    client_ip:  "195.235.232.206"
    browser:    "dummy_browser_4"
    os:         "dummy_os_4"
    device:     "dummy_device_4"
    ▼ location:
      lat:      40.4
      lon:      -3.6833
```

Figura 5.2: API Rest Elasticsearch.

5.2.2. Kibana en Webserver

Para acceder a Kibana, deberemos primero instalar y ejecutar el servicio². Una vez Kibana está ejecutando podremos acceder a su interfaz gráfica mediante la URL y puerto configurado en la instalación:

`http://127.0.0.1:5601/app/kibana#/home`

Ya con datos en Elasticsearch almacenados y como se muestra en la Figura 5.3, Kibana nos pedirá que añadamos los índices sobre los que queremos obtener soporte y seleccionemos el campo sobre el que se filtrarán temporalmente estos logs. En este caso se utilizará el campo `start_date`.

Step 1 of 2: Define index pattern

Index pattern

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

> Next step

✓ **Success!** Your index pattern matches **6 indices**.

kibotics_classification_log
kibotics_error_log
kibotics_ranking_log
kibotics_session_log
kibotics_simulation_log
kibotics_visit_log

Figura 5.3: Creación de índices en Kibana.

Configurados todos los índices en Kibana se tiene una primera visualización de los documentos indexados en Elasticsearch en la pestaña *Discover* de Kibana. En esta pestaña podremos ver todos los logs que estén almacenados en los índices, así como un histograma en el que se podrá ver gráficamente la evolución de estos índices en el tiempo. Además, esta sección *Discover* proporciona la posibilidad de filtrar por rangos de fechas y campos así como se tenía en el primer prototipo de este proyecto.

²<https://www.elastic.co/guide/en/kibana/current/targz.html>

Esta sección ya nos da una primera pincelada de la potencia de procesamiento de Kibana, así como la sencillez de implementación y despliegue. Las consultas son instantáneas, notablemente más rápidas que las realizadas en el primer prototipo desarrollado.



Figura 5.4: Sección *Discover* en Kibana.

Kibana ofrece la posibilidad de creación de *scripted fields*, campos cuyo valor deriva de otros campos o datos ya indexados. Generados en un lenguaje muy similar a C llamado *painless*.

Para el módulo de analíticas que se quiere desarrollar, y en especial para las visualizaciones que filtran por día de la semana y por hora del día, se han creado dos *scripted fields* para cada uno de los índices utilizados. Estos son *day_of_week* y *hour_of_day*, el código *painless* utilizado para la creación de estos campos es el siguiente:

```
// day_of_week
["", "1 Lunes", "2 Martes", "3 Miercoles", "4 Jueves", "5 Viernes",
"6 Sabado", "7 Domingo"] [doc['start_date'].value.dayOfWeek]

// hour_of_day
doc['start_date'].value.hourOfDay
```

Con estos *scripted fields* además de los campos ya almacenados en los documentos, Kibana dispone de todos los datos necesarios para la creación de las visualizaciones. Para ello, en la sección *Visualize*, Kibana tiene una colección de distintas plantillas gráficas. Estas plantillas

serán las que se configuren con los índices, documentos y campos a usar para la generación de los distintos tipos de visualizaciones. En la Figura 5.5 se muestran algunas de las visualizaciones que Kibana ofrece.

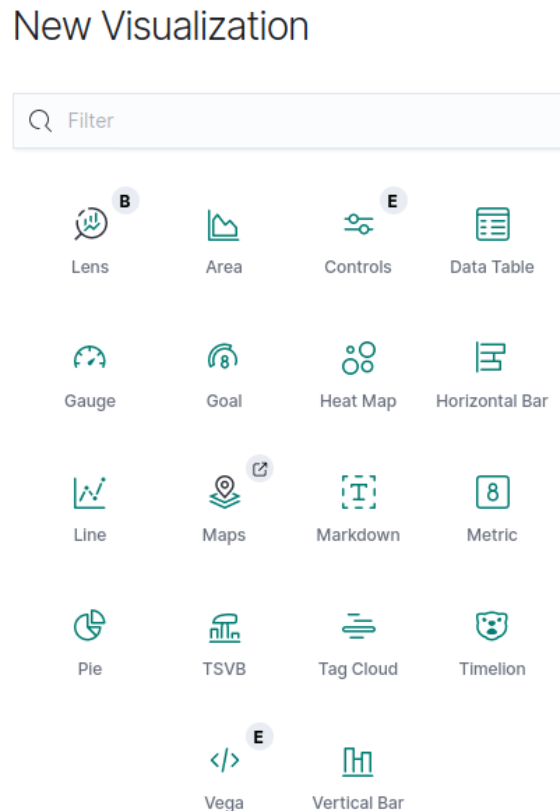


Figura 5.5: Menú creación de visualizaciones.

Ya creadas las visualizaciones, solo quedará integrarlas en Kibotics sustituyendo las generadas en Matplotlib. Para ello se creará una vista `menú` con la que se seleccionará a que tipo de analíticas se quiere acceder. Este menú selector de analíticas es el mostrado a continuación en la Figura 5.6.

SELECTOR DE FECHAS

20200512/20200610 ▼

ANALÍTICAS DE USUARIOS

NOMBRE DE USUARIO GRUPO TODOS LOS USUARIOS

admin_dummy Buscar Admin Buscar Buscar

ANALÍTICAS VISITANTES

TODOS LOS VISITANTES

Buscar

Figura 5.6: Menú de selección de analíticas en Kibotics

En esta vista se filtrará tanto por usuarios y grupos, como por fechas de las cuales se quieren analíticas. Una vez filtrado, Django generará automáticamente una URL con los datos seleccionados en el Menú de Kibotics. Esta URL dinámica apuntará a las visualizaciones de nuestro servicio de Kibana y será devuelta por el contexto de Django hasta las plantillas Django que lo insertarán en un elemento HTML `iFrame`.

5.3. Validación experimental

Para esta versión final del módulo de analíticas, se han creado una amplia selección de visualizaciones de las que obtener información de actividad de los usuarios. Divididas en dos *Dashboards* o tableros, el primero reservado a los visitantes no registrados y el segundo para analíticas de sesiones y simulaciones de usuarios registrados. A continuación, se mostrarán las visualizaciones creadas así como una explicación de lo que representan.

Para tener un control del número de accesos a lo largo del tiempo se creó el histograma representado en la Figura 5.7. Este histograma muestra los eventos almacenados en el índice `kibotics_session_log` dividido día a día en el eje de abscisas.

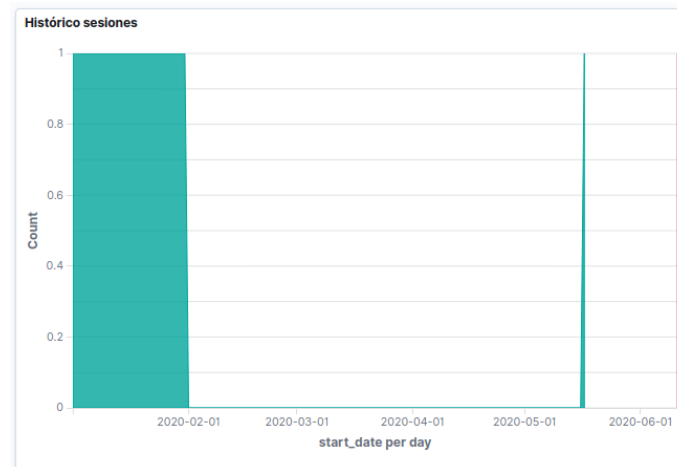


Figura 5.7: Histograma

Se ha creado un mapa de calor, que al igual que el histograma antes mencionado, representa la actividad de inicio de eventos. Dividido en columnas, cada una representa una semana con sus 7 días correspondientes. Un color más oscuro en la celda indica mayor actividad para ese día. Como se puede observar en la Figura 5.8, hay una visualización para las sesiones y otro para las simulaciones, división recurrente que se observa en distintas secciones del módulo Kibana desarrollado.

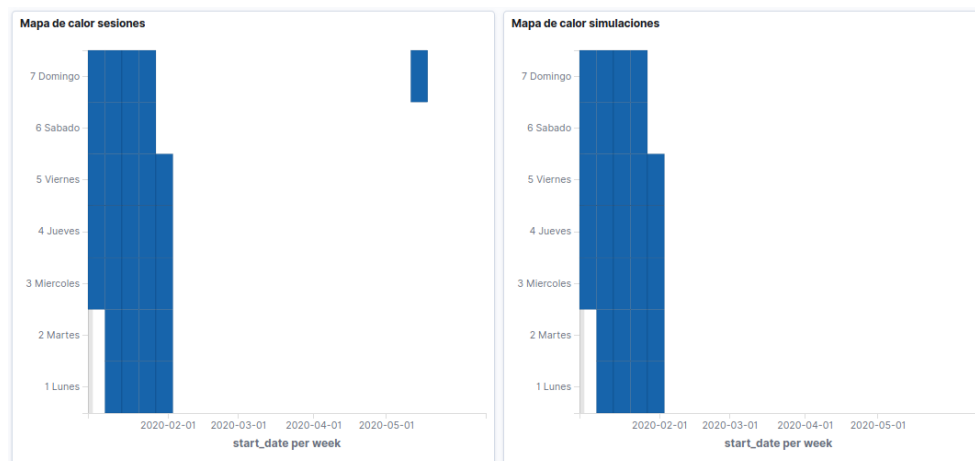


Figura 5.8: Mapa de calor sesiones y simulaciones

La Figura 5.9, dividida en dos visualizaciones, representa la actividad tanto de sesiones como de simulaciones almacenada en los índices de Elasticsearch `kibotics_session_log` y `kibotics_simulation_log` respectivamente. En formato gráfico de barras vertical, los datos, filtrados por el día de la semana en que se registraron en el campo inicio de evento

start_date.

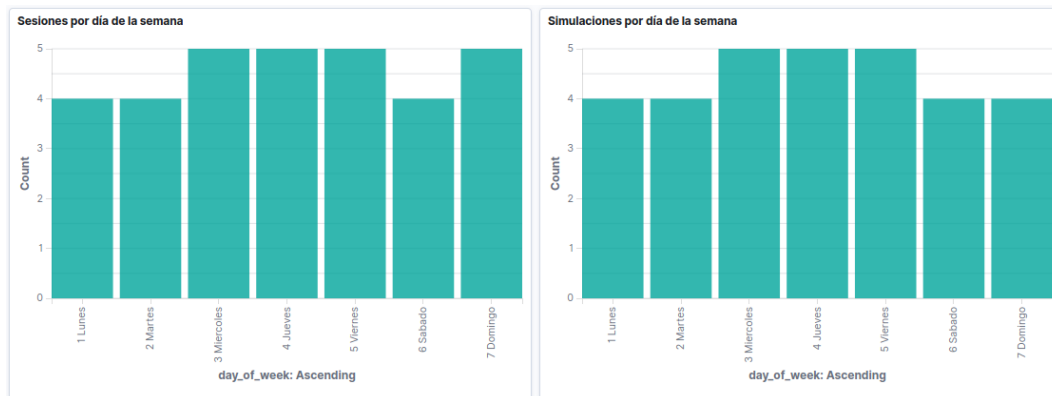


Figura 5.9: Gráfico de barras por día de la semana

La Figura 5.10 muestra, en dos visualizaciones con gráficas de barras verticales, la actividad en el servicio web Kibotics. Filtrados, esta vez, por la hora del día en que se registraron los eventos. Como se puede observar, cada una de estas visualizaciones corresponde a uno de los eventos que han sido logueados en Elasticsearch y almacenados en los índices `kibotics_session_log` y `kibotics_simulation_log`.

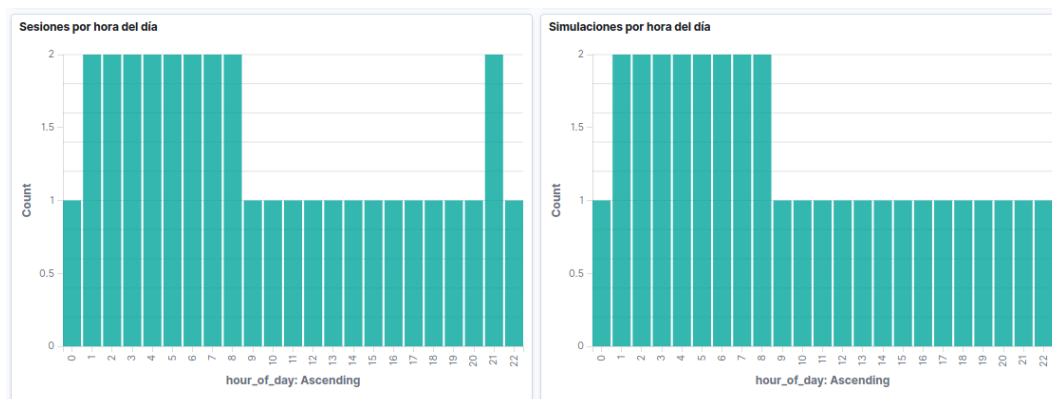


Figura 5.10: Gráfico de barras por hora del día

Para mostrar la actividad en las simulaciones que Kibotics ofrece, se han desarrollado las visualizaciones representadas en la Figura 5.11. Dividido en dos gráficos de barras que representan el tiempo invertido por los usuarios. Esta información está almacenada en el campo `duration` del índice `kibotics_simulation_log`. La primera visualización muestra el tiempo total invertido y la última representa el tiempo medio invertido.

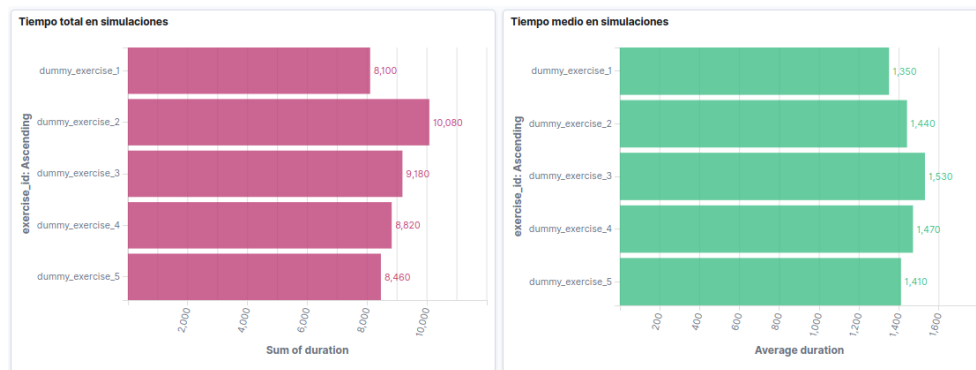


Figura 5.11: Gráfico de barras para tiempo total y medio en simulaciones

En la Figura 5.12, se representa gráficamente la superficie terrestre, en ella se muestran eventos de inicio de sesión ocurridos para el rango de fechas seleccionado. Hace uso de los datos de latitud y longitud almacenados en formato `Geo Point` del índice `kibotics_session_log`.

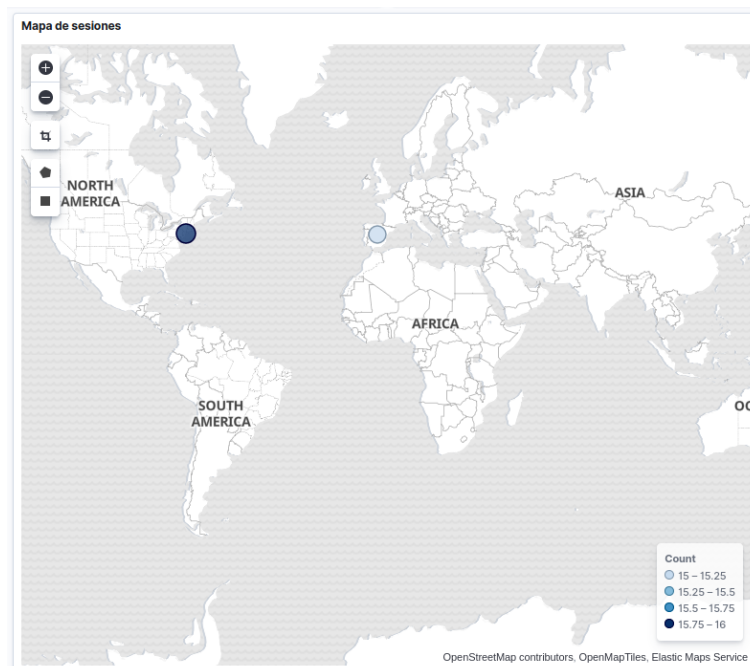


Figura 5.12: Mapa geográfico de sesiones

Para monitorizar el *hardware* y *software* utilizado por los usuarios de la web, se han desarrollado las tres gráficas circulares mostradas a continuación en la figura 5.13. Cada una de ellas representa porcentualmente campos almacenados en el índice de sesiones `kibotics_session_log` que proporcionan información acerca del sistema operativo, dispositivo y navegador utilizados.

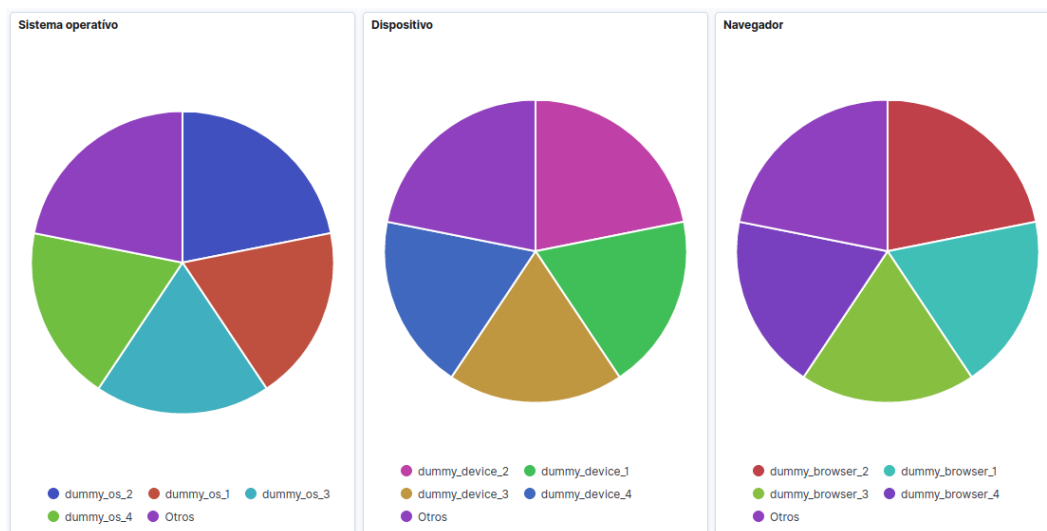


Figura 5.13: Gráficas circulares para SO, Dispositivo y Navegador

En la figura 5.14, se muestra una última sección con dos tablas de datos con los últimos eventos de sesión y simulación almacenados en los índices `kibotics_session_log` y `kibotics_simulation_log` de Elasticsearch para cada uno de los usuarios filtrados.

Últimas sesiones			Últimas simulaciones		
username: Descending	start_date: Descending	Count	username: Descending	start_date: Descending	Count
admin_dummy	May 17, 2020 @ 23:21:37.702	1	betatester_dummy	Jan 27, 2020 @ 05:05:00.000	1
betatester_dummy	Jan 27, 2020 @ 05:00:00.000	1	dummy_user_1	Jan 30, 2020 @ 08:05:00.000	1
dummy_user_1	Jan 30, 2020 @ 08:00:00.000	1	dummy_user_2	Jan 31, 2020 @ 09:05:00.000	1
dummy_user_2	Jan 31, 2020 @ 09:00:00.000	1	student_dummy	Jan 29, 2020 @ 07:05:00.000	1
student_dummy	Jan 29, 2020 @ 07:00:00.000	1	teacher_dummy	Jan 28, 2020 @ 06:05:00.000	1
teacher_dummy	Jan 28, 2020 @ 06:00:00.000	1	admin_dummy	Jan 26, 2020 @ 04:05:00.000	1
dummy_user_3	Jan 19, 2020 @ 20:00:00.000	1	dummy_user_3	Jan 19, 2020 @ 20:05:00.000	1
dummy_user_4	Jan 20, 2020 @ 21:00:00.000	1	dummy_user_4	Jan 20, 2020 @ 21:05:00.000	1
dummy_user_5	Jan 21, 2020 @ 22:00:00.000	1	dummy_user_5	Jan 21, 2020 @ 22:05:00.000	1

Figura 5.14: Últimos eventos logueados

Todas estas visualizaciones son interactivas y se puede filtrar por sus campos simplemente pulsando sobre ellas. Funcionalidad muy útil que no poseían las imágenes renderizadas que se generaban en el primer prototipo. Además, cada una de las visualizaciones del módulo desarrollado tiene una opción para ver los datos representados en texto plano e incluso descargarlos en formato CSV para su posterior tratamiento.

Se han mostrado las visualizaciones referentes a la sección de sesiones y simulaciones de usuarios registrados ya que es la que más información proporciona, pero hay otra sección de analíticas de visitantes con unas visualizaciones de monitorización similares a las mostradas anteriormente.

A continuación, en las siguientes figuras, se muestra Kibana ya integrado en la nueva vista de analíticas del servicio web Kibotics haciendo uso del `iFrame`.

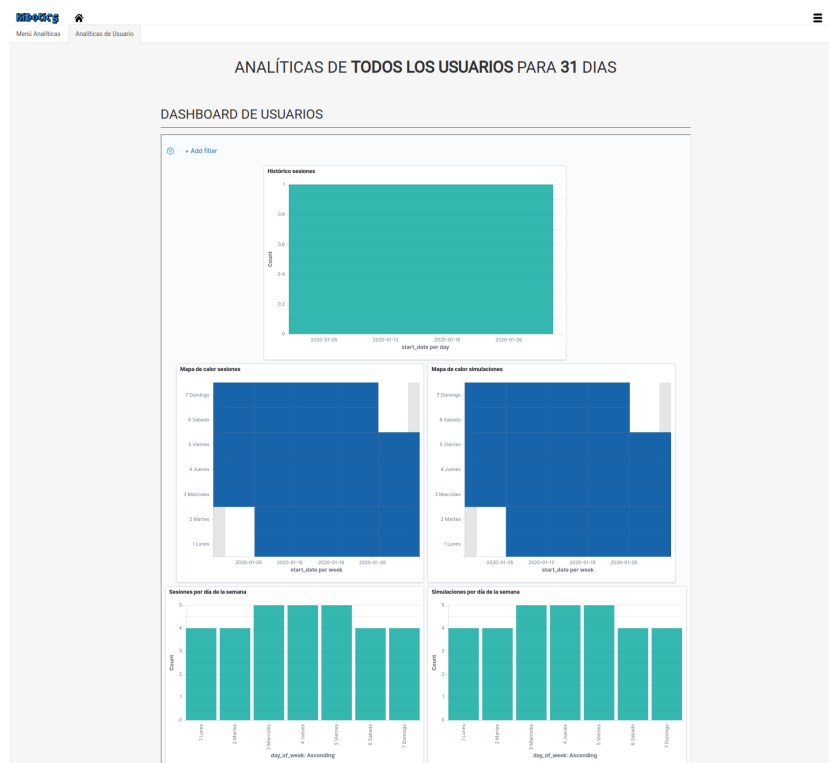


Figura 5.15: Kibana en Kibotics parte 1

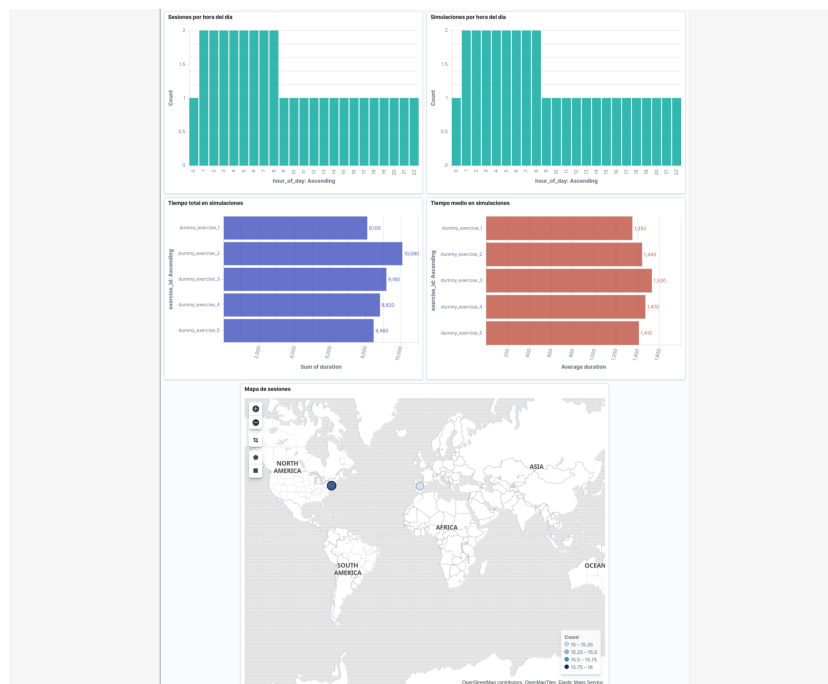


Figura 5.16: Kibana en Kibotics parte 2

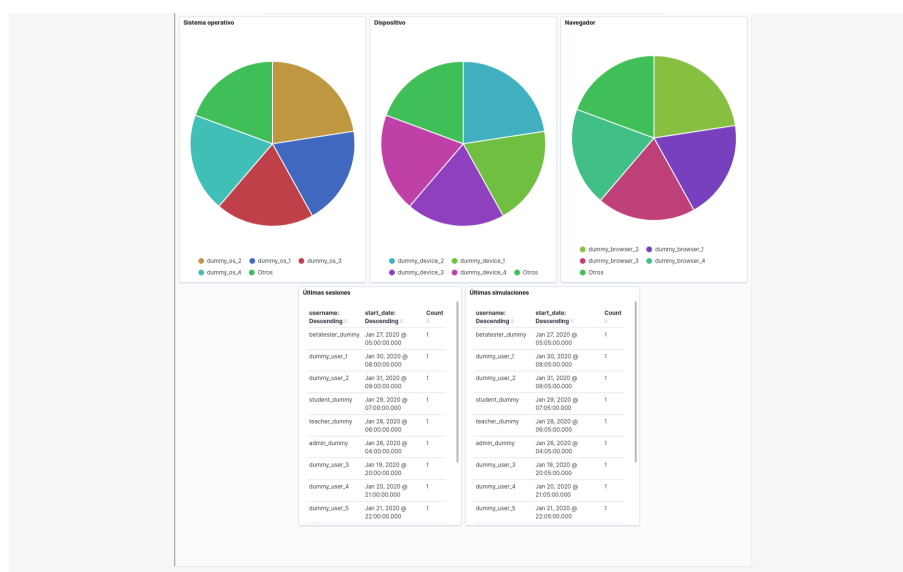


Figura 5.17: Kibana en Kibotics parte 3

Para proporcionar a los futuros desarrolladores los recursos necesarios con los que poder empezar a trabajar sin tener que crearlos de manera manual, se ha creado una base de datos dummy la cual ofrece una variedad de datos para todos los índices utilizados tanto en este proyecto, como en otros desarrollos paralelos.

La creación de esta base de datos de pruebas se generó mediante un *script* de Python que crea los índices y su estructura haciendo uso de la librería que Elasticsearch proporciona para Python.

El *script* creará todos los índices con sus respectivas estructuras y tipologías de datos. Un ejemplo para el índice de sesiones es:

```
# Import librería Elasticsearch y conexión
from elasticsearch import Elasticsearch
client = Elasticsearch()

# JSON con la estructura del índice
session_mapping = {
    "settings": {
        "number_of_shards": 1,
        "number_of_replicas": 0 },
    "mappings": {
        "properties": {
            "username": { "type":"keyword" },
            "start_date":{ "type":"date" },
            "end_date":{ "type":"date" },
            "duration": { "type":"double" },
            "client_ip":{ "type":"ip" },
            "browser": { "type":"keyword" },
            "device":{ "type":"keyword" },
            "location": { "type": "geo_point" },
            "os":{ "type":"keyword" }
        }
    }
}

# Creación del índice en Elasticsearch
client.indices.create( index="kibotics_session_log",
                      body=session_mapping,
                      ignore=400 )
```

Una vez creadas las estructuras de los índices, el siguiente paso es guardar todos los objetos con la información de prueba que se desee guardar.

Finalizada la ejecución del *script* ya tendríamos los datos en nuestro servicio local de Elasticsearch. Para simplificar más la instalación de la base de datos, se han generado una serie de documentos JSON con la estructura y datos que otros usuarios importarán a su servicio Elasticsearch.

Estos documentos JSON se han generado haciendo uso de la herramienta `elasticdump`, la cual tiene una instalación muy sencilla:

```
$ sudo npm install elasticdump -g
```

Para la generación de los documentos de datos y mapeo se han ejecutado las siguientes sentencias para cada uno de los índices usados en Elasticsearch:

```
$ elasticdump --input=http://127.0.0.1:9200/"INDEX_NAME"  
--output="./mapping_elasticsearch.json" --type=mapping
```

```
$ elasticdump --input=http://127.0.0.1:9200/"INDEX_NAME"  
--output="./data_elasticsearch.json" --type=data
```

Para la importación de estos ficheros en la base de datos, el desarrollador simplemente tiene que instalar `elasticdump` y ejecutar un *script bash* el cual recorre y almacena todos los ficheros al servicio Elasticsearch local:

```
indexes="session simulation visit error classification ranking"  
directory="./kibotics_dummy_es/"  
  
for index in $indexes; do  
    elasticdump --output=http://127.0.0.1:9200/"kibotics_"$index"_log"  
    --input=$directory"mapping_"$index"_es.json" --type=mapping  
  
    elasticdump --output=http://127.0.0.1:9200/"kibotics_"$index"_log"  
    --input=$directory"data_"$index"_es.json" --type=data  
done
```

El proceso de exportación e importación de datos de prueba para Kibana es sencillo pues la propia interfaz gráfica de Kibana nos proporciona una herramienta para realizarlo como muestra la Figura 5.18.

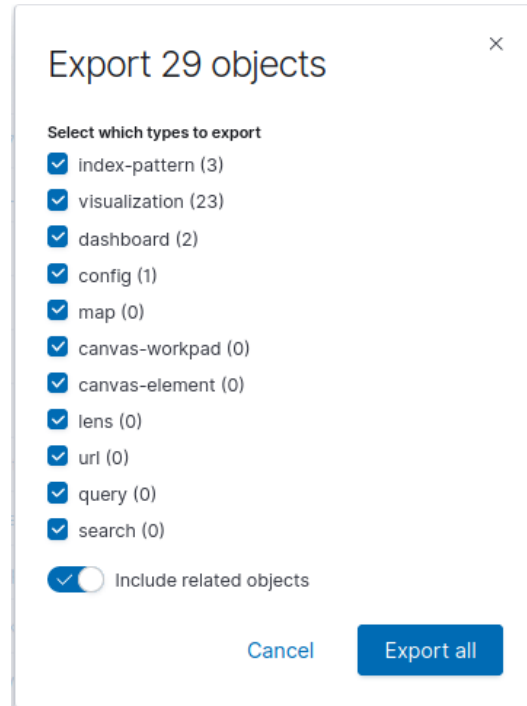


Figura 5.18: Exportación en interfaz Kibana.

Esta herramienta nos generará un fichero NDJSON similar a la estructura JSON con los patrones de índices creados, así como las visualizaciones, tablas o *scripted fields* guardados en Kibana.

Para que un futuro desarrollador importe estos datos simplemente podrá hacerlo por la interfaz gráfica de Kibana. Para unificar la metodología y ya que en el servidor de pre-producción/producción no se dispone de esta interfaz gráfica, esta también se puede realizar mediante la siguiente sentencia:

```
$ curl -X POST "localhost:5601/api/saved_objects/_import" -H "kbn-xsrf: true"
--form file=@kibotics_dummy_kibana.ndjson
```