Capítulo 5

INTEGRACIÓN DEL STACK ELK EN KIBOTICS

En este capítulo se describen las tecnologías del Stack ELK utilizadas en la versión final desarrollada del módulo de analíticas. Tanto Elasticsearch como base de datos, como Kibana de procesador de datos y generador de gráficas. Además de unos recursos de prueba generados para que futuros desarrolladores dispongan de una instalación sencilla de bases de datos y servicios necesarios.

5.1. Desarrollo local

En está sección se describe la evolución que han sufrido los logs de la aplicación. Así como la versión final de la herramienta de analíticas que hará uso del Stack ELK.

5.1.1. ElasticSearch en Kibotics Webserver

El primer paso el proceso de migración es el cambio de bases de datos. Para hacer uso del Stack ELK, es necesario sustituir MongoDB por Elastic-Search. Para esto lo primero será instalar e inciar un servicio de Elasticsearch en local.

Una vez iniciado el servicio Elasticsearch podremos acceder a los datos indexados en las tablas mediante la interfaz a la que se podrá acceder por la terminal de comandos o de manera más amigable, desde la URL y puerto local que hayamos configurado en la instalación de ElasticSearch.

Un ejemplo de llamada para un índice llamado index_name_test será la siguiente:

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

El siguiente paso en la migración es la integración de Elasticsearch en Django, se realizará haciendo uso de la librería django_elasticsearch_dsl, la cual simplificará esta tarea. La cual, se divide en dos partes principales. Creación de los índices y migración de las sondas de MongoDB a Elasticsearch.

Este primer paso de creación de índices es similar a la metodología de modelos que posee Django con las estructuras de las bases de datos. Se creará los siguientes índices:

- kibotics_session_log: En el cual se almacenarán todos los eventos referentes a las sesiones.
- kibotics_simulation_log: En el cual se almacenarán todos los eventos referentes a las simulaciones.
- kibotics_error_log: En el cual se almacenarán todos los eventos referentes a los errores.
- kibotics_visit_log: En el cual se almacenarán todos los eventos referentes a las visitas.

Aprendiendo de lo aplicado en el primer prototipo y para evitar tener que cruzar datos para el calculo de duraciones de los eventos logueados, tanto en el índice de sesiones como de simulaciones se ha eliminado el campo que registraba la fecha. Para sustituirlo, se han añadido dos nuevos campos los cuales registrarán el inicio y fin de cada evento, unificando así los dos registros de log que se tenían previamente en uno.

Por otro lado el campo USER_AGENT ha sido dividido y sustituido con la información del navegador, dispositivo y sistema operativo con el que cada usuario accederá a la web.

Para trabajar en Kibana con mapas es necesario guardar tanto la longitud como la latitud, para lo cual se creará un campo ya existente llamado Geo Point que almacenará en un diccionario ambos campos.

Para complementar las nuevas sondas, de las que se a continuación, es necesario la creación de un nuevo índice de visitas, en el que se registrarán los accesos a la pagina principal de la aplicación, estén o no registrados.

Un ejemplo de la estructura de uno de estos índices en el fichero Python 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
        }
```

Por último, ya solo será necesaria la migración de las sondas de obtención de logs. Para enriquecer las sondas ya existentes, se han añadido nuevas para registrar eventos de visitantes, así como sondas para fin de sesiones/simulaciones (cierre busco de la aplicación, temporizador por inactividad) con el fin de evitar entradas de log sin fecha de fin.

Las sondas de inicio de los eventos son similares a las que se tenían anteriormente en MongoDB, por ejemlo:

```
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, las sondas de fin de sesión/simulación cambian. Tendrán que buscar en ElasticSearch el último registro de log en el índice para el usuario del cual se quiera actualizar el registro de log y sustituir tanto los

campos end_date como duration.
Esto se realiza gracias al identificador que cada registro indexado posee al cual se le realiza una operación update con los nuevos campos:

```
latest_session = Search(index="kibotics_session_log*") \
    .query("match", username=username) \
    .query('match', duration=0) \
    .sort({"start_date": {'order': 'desc'}})[0]

# Update session with leave date and 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()
                    }
                }
            )
```

Para comprobar que todos estos campos están siendo creados y actualizados correctamente se puede realizar haciendo consultas tanto por terminal, así como a la API Rest de ElasticSearch que se levantó previamente, accediendo mediante la URL y puerto configurados durante el proceso de instalción.

En esta API, podremos realizar filtrados por campos e índices haciendo uso de expresiones regulares Regex, un ejemplo genérico para uno de los índices de kibotics será:

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

En la siguiente figura se puede observar la respuesta que se obtiene a la llamada anterior.

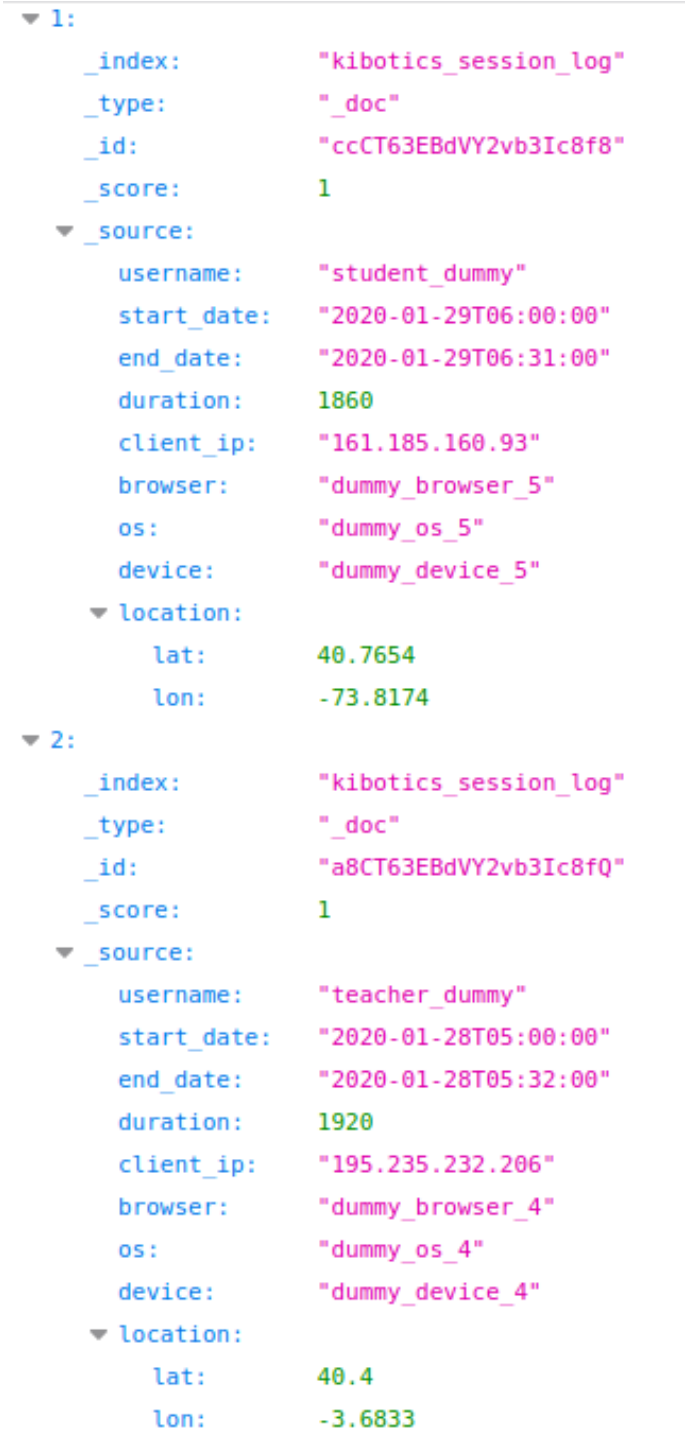


Figura 5.1: API Rest ElasticSeach.

5.1.2. Kibana en Kibotics Webserver

Para acceder a kibana, deberemos primero instalar y ejecutar el servicio. 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, Kibana nos pedirá que registremos los índices sobre los que queremos obtener soporte y seleccionemos el campo sobre el que se filtrarán temporalmente estos logs. En este caso, el campo start_date.

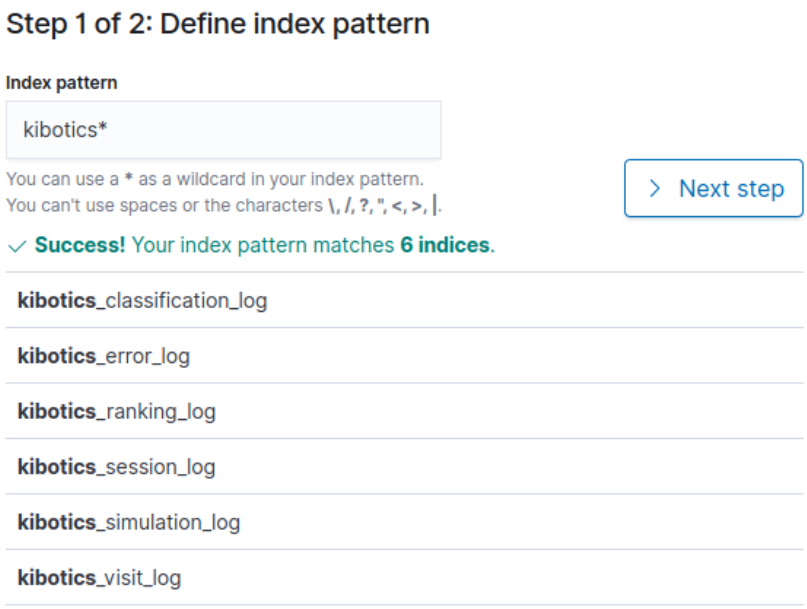


Figura 5.2: Creación de índices en Kibana.

Configurados todos los índices en Kibana, se podrá tener una primerá visualización de los datos indexados en ElasticSearch en la pestaña Discover de Kibana. En esta pestaña podremos ver todos los logs que estén guardados así como un histograma en el que se podrá ver gráficamente la evolución de los índices. 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.

Esta sección ya nos da una primera pincelada de la potencia de procesamiento de Kibana, así como la sencillez de implementación y despliegue.

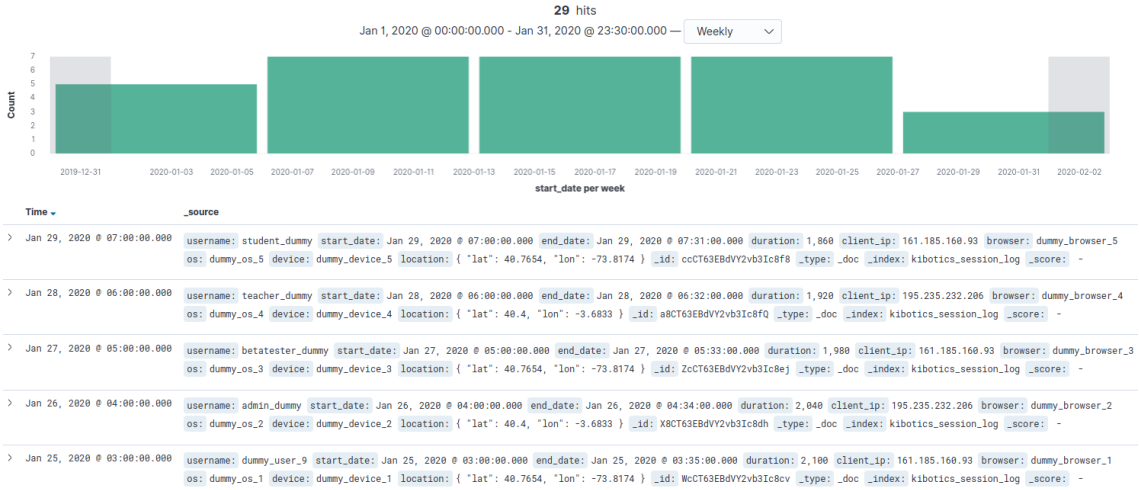


Figura 5.3: Sección Discover en Kibana.

Kibana ofrece la posibilidad de creación de scripted fields, estos son, campos cuyo valol derivará de otros campos o datos ya indexados, generados en un legunaje muy similar a C llamado painless. Para la versión final de la herramienta necesitaremos crear varios de estos campos para distintas gráficas que se explicarán más adelante como por ejemplo el mapa de calor de actividad.

Estos son el día de la semana y la hora del día en que los logs se registraron.

```
# 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 campos creados además de los contenidos en los registros de log, Kibana ya dispone de todos los datos necesarios para la creación de las visualizaciones. Para ellos en la sección Visualize, kibana tiene una colección de distintas plantillas gráficas. Las cuales, deberán ser configuradas para representar los datos e índices que sean necesarios.

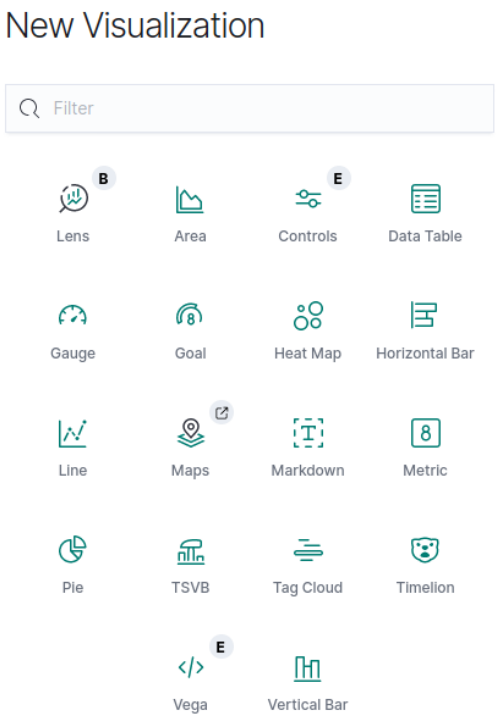


Figura 5.4: Menú creación de visualizaciones.

Se han creado una amplia selección de visualizaciones. Divididas en dos Dashboards, el primero reservado a los visitantes y el segundo para analíticas de sesiones y simulaciones de usuarios registrados. A continuación se mostrarán las gráficas creadas, así como una breve explicación de lo que representan.

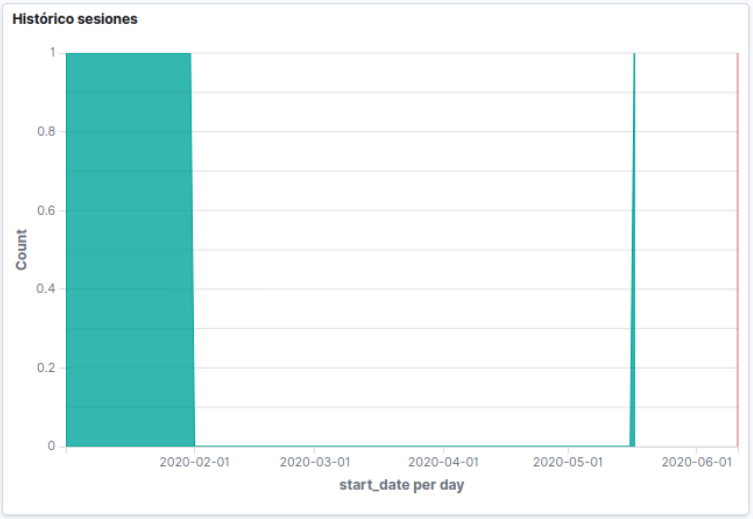


Figura 5.5: Histograma

Histograma que representará el número de logs registrados en el índice de sesiones para el rango de fechas señalado.

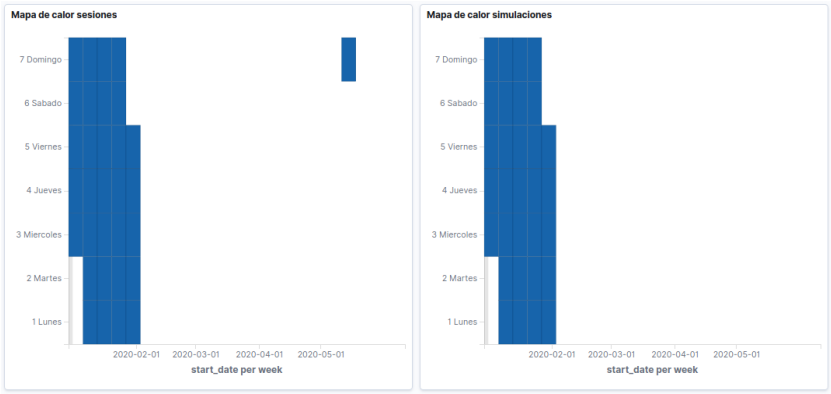


Figura 5.6: Mapa de calor sesiones y simulaciones

Mapas de calor para los índices se sesiones y simulaciones, divididos en columnas por semanas, cada una de ellas en los respectivos días de la semana. Un color más oscuro representará mayor actividad para ese día.

Figura 5.7: Gráfico de barras por día de la semana

Gráfica de barras que dividirá los datos filtrados por el día de la semana en que se registraron. Tanto para sesiones como para simulaciones.

Figura 5.8: Gráfico de barras por hora del día

Gráfica de barras que dividirá los datos filtrados por la hora del día en que se registraron. Tanto para sesiones como para simulaciones.

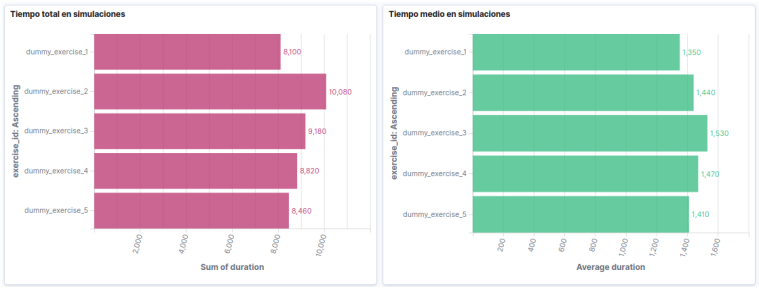


Figura 5.9: Gráfico de barras para tiempo total y medio en simulaciones

Gráfica de barras que representará el tiempo invertido por los usuarios filtrados en las respectivas simulaciones. El gráfico de la izquierda muestra el tiempo total y el de la derecha el tiempo medio invertido.

Figura 5.10: Mapa geográfico de sesiones

Representación geográfica de los eventos de sesión ocurridos para el rango de fechas filtrado.

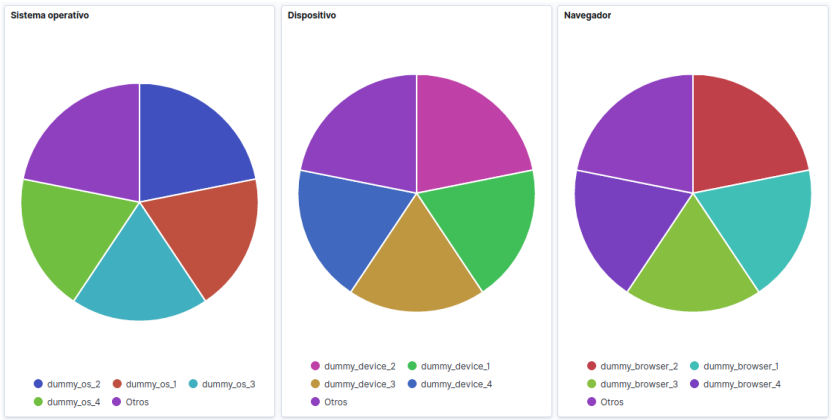
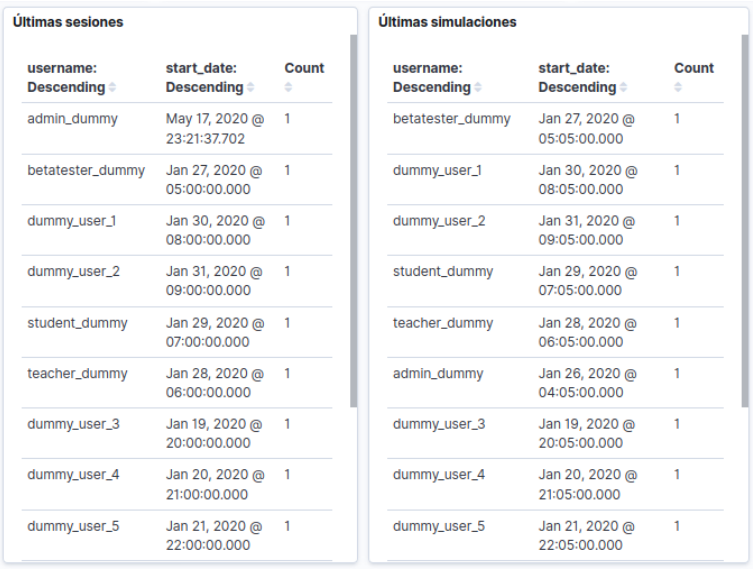


Figura 5.11: Gráficas circulares para SO, Dispositivo y Navegador

Tres gráficas circulares que representarán información acerca de los dispositivos que están siendo utilizados para acceder a la aplicación, filtra por sistema operativo, dispositivo y navegador.



Últimas sesiones

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

Últimas simulaciones

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

Figura 5.12: Últimos eventos logueados

Tabla de datos con los últimos registros de sesión y simulación para cada uno de los usuarios filtrados.

Todas estas visualizaciones son interactivas y se puede filtrar por sus campos simplemente pulsando sobre ellas. Funcionalidad muy útil que no poseían las imagenes renderizadas que se generabane n el primer prototipo.

Ya creadas las visualizaciones, solo quedará integrarlas en Kibotics al igual que se hizo con las generadas en Matplotlib. Para ello se creará una vista simple con la que se seleccionará a que tipo de analíticas se quiere acceder.

Figura 5.13: Menú de selección de analíticas en Kibotics

En esta vista se filtrará tanto por usuarios y grupos, como por fechas de las cuales queremos analíticas. Una vez filtrado, Django generará automaticamente 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 que lo insertarán en un elemento HTML iFrame.

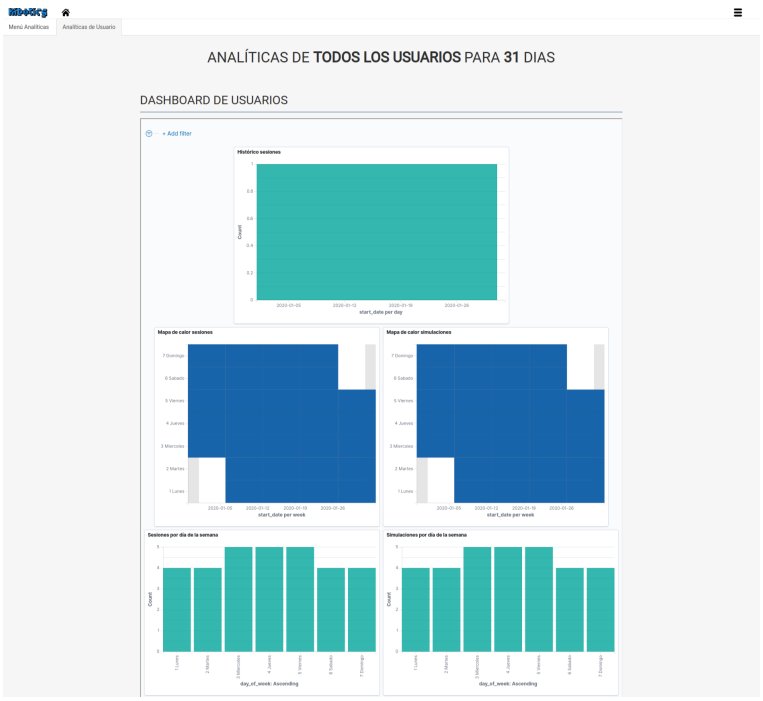


Figura 5.14: Kibana en Kibotics parte 1

Figura 5.15: Kibana en Kibotics parte 2

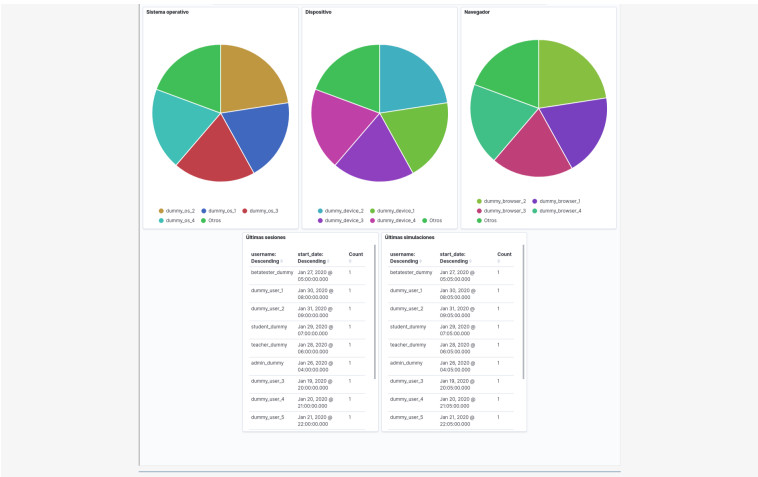


Figura 5.16: Kibana en Kibotics parte 3

5.2. Despliegue en producción

(INCOMPLETO)

5.3. Generación de recusos de prueba para el desarrollo local

Para facilitar futuros desarrollos y ampliaciones de las mejoras de analíticas implementadas se han generado una serie de recursos para proveer al repositorio Github de herramientas.

5.3.1. Receta de instalación de ElasticSearch

Se ha incluido en el repositorio GitHub la receta necesaria para instalar la versión de ElasticSearch utilizada en producción.

Con esto, se asegura que cualquier futuro desarrollador de la plataforma simplemente tenga que seguir la receta de instalación ahí explicitada para tener un entorno totalmente funcional.

Actualmente la última versión estable de ElasticSearch es la versión 7.7.2, liberada el 3 de Junio de 2020. Por cuestión de fechas, en producción, actualmente se está utilizando la versión 7.6.2 del 31 de Marzo del 2020.

5.3.2. Creación de bases de datos Dummy para Elasticsearch

Para proporcionar a los futuros desarrolladores datos realistas con los que poder empezar a trabajar sin necesidad de 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.

Para la creación de esta base de datos de pruebas, se generó mediante un script de Python el cual crea los índices y su estructura haciendo uso de la librería de 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
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 guar-

dar todos los objetos con la información de prueba que se deseé guardar.

Finalizada la ejecución del script ya tendríamos los datos en nuestro servicio local de Elasticsearch. Un problema que se encontró es que para ejecutar este script es necesaria la instalación de varias librerías. Para simplificar aún 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_elasticseach.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 tendrá que instalar elasticdump y ejecutar un script bash el cual recorrerá y cargará 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
```

5.3.3. Receta de instalación de Kibana

Para completar la instalación de recursos del Stack ELK utilizados en el proyecto se ha añadido la receta de instalación de Kibana.

Actualmente la última versión estable de Kibana es la versión 7.7.1, liberada el 3 de Junio de 2020. Por cuestión de fechas, en producción, actualmente se está utilizando la versión 7.7.0 del 13 de Mayo del 2020.

5.3.4. Creación de bases de datos Dummy para Kibana

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.

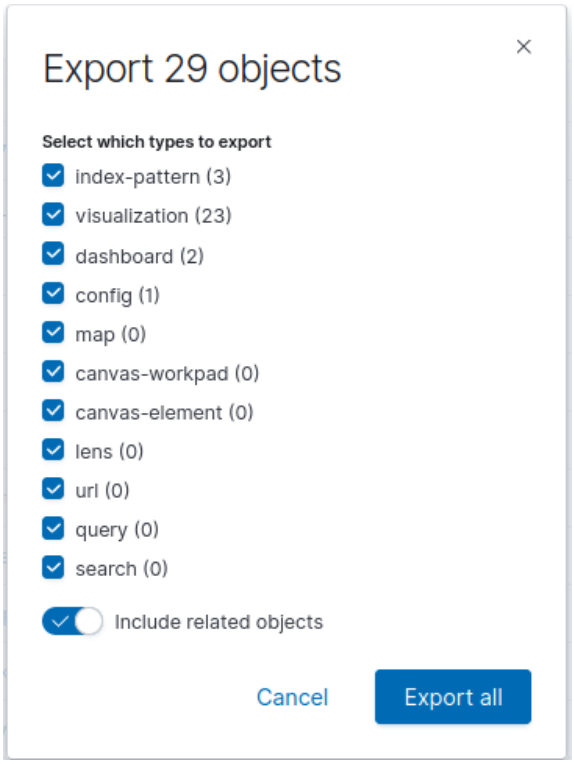


Figura 5.17: Exportación en interfaz Kibana.

Esta herramienta nos generará un fichero NDJSON similar a la estructura JSON con los patrones de índices creados, así como las visualizaciones, tablas o campos scripted 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
```