Arquitectura near real time para el análisis de negocio

RODRIGO DE MIGUEL GONZÁLEZ

MASTER EN ARQUITECTURA BIG DATA
KSCHOOL

TRABAJO FIN DE MASTER

Curso 2018-2019

Director: Rubén Casado Tejedor

1. Introducción **3**

2. Casos de uso alto nivel **4**

3. Arquitectura técnica **5**

5. Procesos **7**

5.1. Dashboard animales 7

5.2. Dashboard búsquedas 7

5.3. Sistema de suscripciones 7

6. Módulos técnicos **7**

1. Introducción

Desde hace año y medio trabajo junto con un amigo (y socio) en el desarrollo de una plataforma web para la adopción y gestión de animales procedentes que las protectoras tiene a su cargo a la espera de ser adoptados por sus nuevas familias.

La idea de este TFM es el desarrollo de una arquitectura que pueda proporcionar información útil de forma cercana al tiempo real sobre el uso que se hace de la plataforma tanto de los cliente de la misma (registro de animales nuevos) como de la demanda de los usuarios (búsquedas de los animales) y poder hacer el análisis y toma de decisiones sobre la misma y estrategias de mercado y marketing.

La plataforma web permite que las protectoras aumenten tanto su visibilidad como la de los animales en internet registrándolos en ella para facilitar a sus posibles adoptantes encontrar el compañero ideal que están buscando, con el objetivo de fomentar la adopción frente a la compra.

Por otra parte, de cara a los usuarios, la plataforma expone un potente buscador de los animales registrados, de forma que poder encontrar la mascota ideal sea muy sencillo.

La plataforma está implementada con React+Redux en el front, NodeJS en el Backend , MongoDB como sistema de persistencia principal y AWS S3 como repositorio de objetos.

La arquitectura montada para este TFM permite que el *backend* de la plataforma además de realizar la persistencia en MongoDB pueda enviar los registros de nuevos animales que se realizan por parte de las protectoras y de las búsquedas que se están realizando a un pequeño cluster de analitica hecho con Elasticsearch y Kibana desplegado en Amazon Web Services.

Además permite que los usuarios puedan suscribirse a los filtros de las búsquedas que quieran para poder recibir vía email avisos inmediatos de los animales que entren nuevos mediante el uso de AWS Lambda, AWS Kinesis (junto con Kinesis Firehose) y el módulo de envío de emails implementado en el en el backend.

2. Casos de uso alto nivel

El entorno de analítica lo conforman dos casos de uso:

- Dashboard en Kibana del registro de animales nuevos
- Dashboard en Kibana de las búsquedas realizadas

Esta parte consiste en un dashboard que muestra la información en tiempo real los animales nuevos y búsquedas que se estén realizando en la plataforma. Haciendo especial hincapié en la especie de los mismos (perros o gatos), y sobretodo en la raza, característica que se considera principal a la hora de elegir un animal de compañía.

Se muestra un conteo de registros o búsquedas, la línea de tiempo del conteo para poder ver la evolución histórica, un top 5 de razas tanto en gatos como perros, y lo que a mi parecer aporta mayor valor de negocio, un mapa de puntos calientes en lo que poder visualizar desde donde hay mayor actividad, desde donde realizan las búsquedas los usuarios y poderlo contrastar con dónde tiene mayor presencia la plataforma y poder realizar campañas de marketing dirigido a esas zonas.

Por otro lado se ha implementado la posibilidad de realizar suscripciones a filtros de búsquedas de forma que al registrarse un nuevo animal se chequean las suscripciones activas que cumplan las características de dicho animal y se procede al envío de un email a los usuarios avisando de la nueva entrada con la información del animal y de contacto con la protectora.

3. Arquitectura técnica

La arquitectura realizada, aunque a mi parecer es sencilla, no conteniendo demasiados elementos, pero está diseñada con una finalidad muy clara, por una parte, la obtención de valor para negocio de lo que ocurre al otro lado de la plataforma, en el lado de clientes/usuarios, y por otro, una cota superior en coste y recursos, ajustarla en coste dado que la intención tras este máster es seguir usandola, y, a día de hoy, será pagada con fondos personales.

Como introducción y posicionar la entrada a la arquitectura del TFM, el backend de la aplicación está realizado en **NodeJS**, usando **ExpressJS** como framework web y de procesamiento de la aplicación operacional. La aplicación, a día de hoy, está desplegada en **Heroku** (soportado por Salesforce), es un **PaaS**, una plataforma como servicio para el despliegue de aplicaciones web.

En dicha plataforma, conectada con el repositorio de código **GitHub**, se tiene desplegado los tres entornos más típicos, el entorno de desarrollo se utiliza la máquina local para la aplicación aunque los distintos servicios utilizados si se encuentran desplegados (MongoDB, SendGrid como servicio de emails y AWS S3), el entorno para preproducción con todos los servicios activos y un entorno de producción completo aunque actualmente no productivo.

La base de datos principal es **MongoDB**, alojada en mLab, un **DBaaS**, una base de datos como servicio. El entorno de trabajo y preproducción por ser más que suficiente para el desarrollo actual usa la versión gratuita de entorno, que consta de un cluster de un único nodo compartido con 0,5GB de almacenamiento y RAM variable. El entorno de producción que se empezará a utilizar será el básico, aunque aún no activo por no ser usado, consta de un cluster de tres dos, un primario, un secundario y uno de arbitraje, parte desde 1GB de almacenamiento ampliable hasta 8, RAM variable, aunque en un estado de poco uso ronda los 150MB.

Por otra parte todo el alojamiento de imágenes se realiza en AWS S3 desde el backend para su uso en el *frontend.*

El backend, una vez realizada la búsqueda o la inserción del nuevo animal en la BBDD envía el registro a los **Streams** de **AWS Kinesis**, desde ahí **Kinesis Firehose** los envía al cluster de **ElasticSearch** para su explotación.

Los streams están configurados con un único shard por el bajo volumen de información, eso sí, su reescalado es trivial pudiéndose automatizar el proceso.

Cada shard de kinesis permite la escritura de 1MB/s o 1000 registros/segundo, y la lectura de 2MB/s, siendo el escalado lineal.

Al igual que el Kinesis, el cluster de Elasticsearch está configurado actualmente con el mínimo de máquinas, una instancia de datos con una t2.small.elasticsearch, una instancia EC2 t2.small customizada para ELK. Obviamente esta configuración de cluster no está nada recomendada para un entorno productivo, para ello AWS probé la posibilidad de tener hasta tres zonas de disponibilidad de las instancias de datos, el reescalado del número de instancias, poder añadir instancias maestras dedicadas, y mejorar las máquinas EC2.

En la parte el almacenamiento es una configuración común para cada instancia, tipo de almacenamiento EBS, el estándar de AWS, con SSDs y 10GB por nodo del cluster. Para la parte de seguridad se pueden configurar el cifrado entre máquinas, backups, y la autenticación en Kibana mediante Amazon Cognito, que para el proceso no se tiene activo.

4. Arquitectura de Despliegue

5. Procesos

5.1. Dashboard animales

5.2. Dashboard búsquedas

5.3. Sistema de suscripciones

6. Módulos técnicos