Cloud

Práctica Introducción a AWS

Olga Penedo Suárez

Datahack

Índice general

Introducción

El objetivo de la práctica es diseñar e implementar una web de anuncios explotando los conocimientos adquiridos.
El código fuente generado se puede encontrar en el repositorio `https://github.com/olgapenedo/web-anuncios.git`

Capítulo 1

Arquitectura

Para la realización de este proyecto se han decidido emplear los siguientes servicios:

- **DynamoDB**, para almacenar todo la información relativa a los anuncios.
- **AWS Lambda**, para crear todas las funciones de escritura y consulta de la tabla.
- **API Gateway**, para enrutar las solicitudes.
- **Amazon S3**, para el alojamiento de sitio web estático. Se generará un bucket que almacenará todo el código html y js necesario para la web.

A continuación se muestra el diagrama de arquitectura correspondiente.

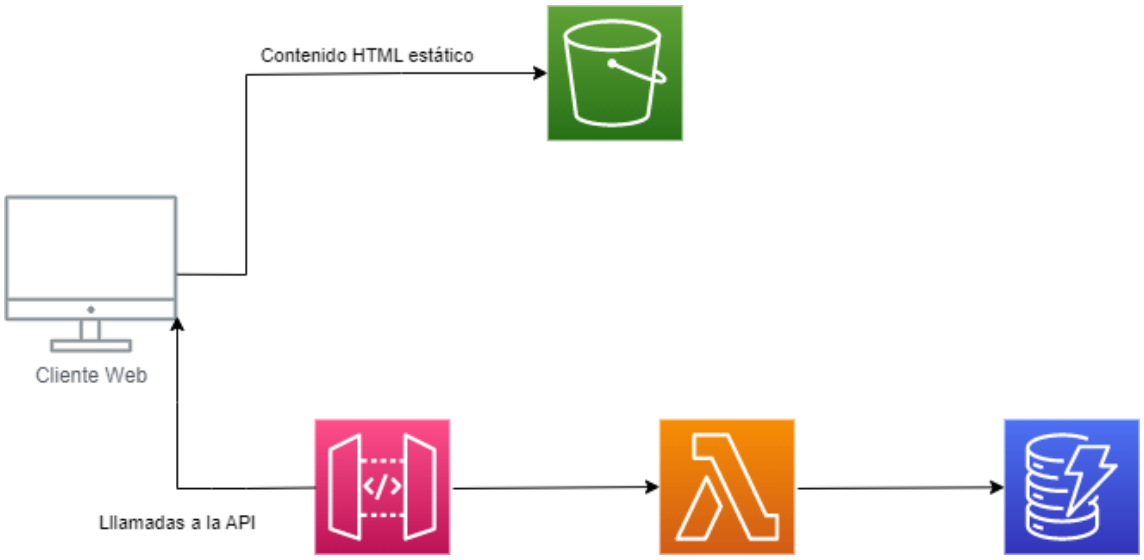

Figura 1.1: Diagrama de arquitectura.

Veamos ahora las razones que han llevado al uso de estas tecnologías:

Escalabilidad

DynamoDB es un servicio de base de datos NoSQL completamente administrado y escalable, lo que permite manejar grandes cantidades de datos y una carga de trabajo alta y variada.
Lambda y API Gateway son completamente autoescalables, pudiendo escalar para manejar el aumento del tráfico.
Por tanto, los tres servicios permitirían soportar un incremento del número de usuarios de la aplicación o de los anuncios incorporados.

Rendimiento

DynamoDB es muy rápido y proporciona tiempos de respuesta bajos, lo que lo hace ideal para aplicaciones web.

Flexibilidad

DynamoDB permite agregar campos a un registro sin tener que hacer cambios en todo el esquema de la base de datos. Esto nos resultará especialmente útil, ya que en primera instancia el anuncio se creará sin comentarios, y este campo solo se añadirá a aquellos anuncios que reciban algún comentario.

Precio

Todos los servicios empleados se incluyen en la capa gratuita de AWS, presentando los siguientes límites:

- **DynamoDB**: 25 GB de almacenamiento, 25 unidades de capacidad de escritura aprovisionada y 25 unidades de capacidad de lectura aprovisionada. Suficiente para manejar hasta 200 millones de solicitudes por mes.
- **AWS Lambda**: 1 millon de solicitudes gratuitas por mes. Hasta 3,2 millones de segundos de tiempo de informática por mes.
- **API Gateway**: 1 millón de llamadas de API recibidas por mes.
- **S3**: 5 GB de almacenamiento estándar, 20000 solicitudes Get y 2000 solicitudes Put.

Por tanto, las pruebas realizadas no supondrán ningún coste.

Otras servicios considerados

- **AWS Amplify**: Se ha valorado el uso de AWS Amplify, ya que incluye alojamiento de aplicaciones web estáticas, pero también ofrece una amplia gama de características, como la autenticación, la autorización, la integración con API y la gestión de bases de datos, entre otros. No obstante, se ha descartado, ya que, aunque se incluye en la capa gratuita, una vez superada la misma suele ser más costoso que S3. Además, la interfaz desarrollada para la aplicación es muy sencilla, por lo que S3 resulta más que suficiente.
- **Cognito**: El uso de este servicio para la administración de usuarios sería interesante para incorporar una mayor funcionalidad a la solución.
- **Route 53**: Se podría emplear Route 53 para crear un nombre de dominio personalizado para una API Gateway en lugar de utilizar el ID de la misma. Esto permitiría el poder conseguir un one-click deployment, ya que no sería necesario conocer dicho ID y tener que explicitarlo para realizar llamadas a las APIs en la web.

Capítulo 2

Diseño

Veamos cómo se ha diseñado y funciona la aplicación.
Se ha decidido crear una tabla de DynamoDB para almacenar los anuncios que contendrá los siguientes campos:

- **adId**: Clave primaria de la tabla, que se genera empleando UUID.
- **nameAd**: Nombre del anuncio.
- **userId**: Identificador del usuario que publica el anuncio.
- **description**: Descripción del anuncio.
- **datePosted**: Fecha de publicación del anuncio.
- **comments**: Array de objetos con el usuario (userId) y el comentario (comment) realizado por el mismo.

Como ya se ha comentado previamente, aunque para esta solución el userId se inserta manualmente, se podría valorar el uso de Cognito para la autenticación de usuarios y, por tanto, emplear ese usuario como userId.
Se han implementado también las siguientes funciones Lambda:

- **createAd**: Inserta un anuncio en la tabla indicando en el event body un nameAd, userId y description.
- **getAdvertisement**: Recupera todos los campos de un anuncio a partir de su adId.
- **getAllAds**: Recupera el adId, nameAd y userId de todos los anuncios.
- **addComments**: Permite añadir un usuario y el comentario que realiza al array comments del anuncio con el adId indicado.

Con el fin de facilitar la interacción con las APIs, se ha creado una interfaz web.
En la primera página, **index.html**, se permite introducir los datos de un nuevo anuncio y crearlo al pulsar *Publicar anuncio*, indicando un mensaje de *Anuncio publicado* y redirigiendo a **ads.html**. Si no se desea crear un nuevo anuncio, se puede pulsar *Ver todos los anuncios disponibles* para poder acceder directamente a **ads.html**.

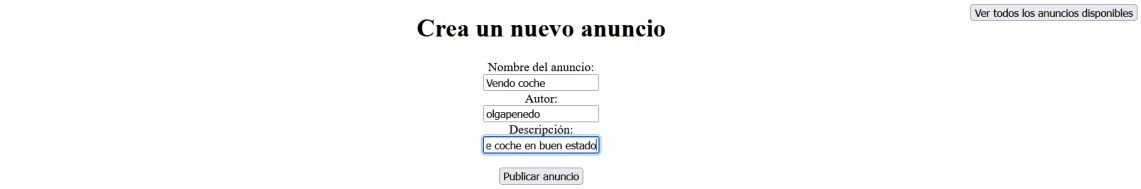

Figura 2.1: Página de inicio.

En la siguiente página se listan todos los anuncios disponibles, en este caso solo el que hemos generado previamente. También se permite buscar por adId o volver a la página anterior para agregar un nuevo anuncio.
En cuanto al buscador, se ha decidio realizar una búsqueda por adId para asegurar el obtener un único anuncio, pero otra posible solución sería la de realizar una búsqueda por nameAd y, en caso de haber varios anuncios con el mismo nombre, poder elegir uno de ellos.

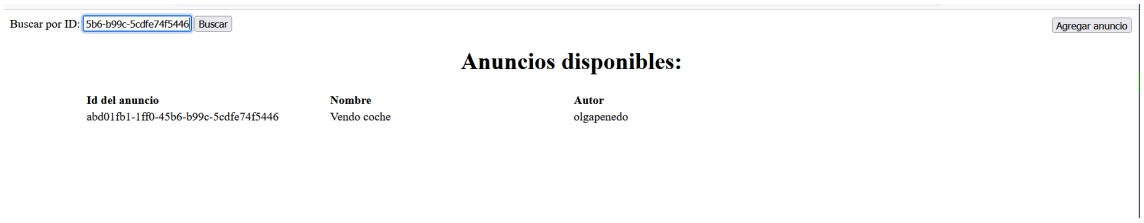

Figura 2.2: Listado de anuncios.

Finalmente, al buscar por adId se nos redirige a **advertisement.html**, en donde se ven todas las características del anuncio en cuestión. Aunque aún no se han añadido comentarios, esta página nos permite hacerlo al pulsar en *Añadir un comentario*. También nos permite volver a la página donde se listan todos los anuncios o publicar un nuevo anuncio. En la siguiente ilustración se muestran los pasos realizados para añadir un comentario y el usuario que lo introduce (cabe volver a recalcar que sería preferible que esto se realizase con un servicio como Cognito), resultantes en la actualización automática de la información mostrada. Como se puede ver, en la url se ha añadido el adId para facilitar dicha

operación.

Figura 2.3: Añadir comentarios.

El siguiente diagrama resume lo detallado en este apartado.

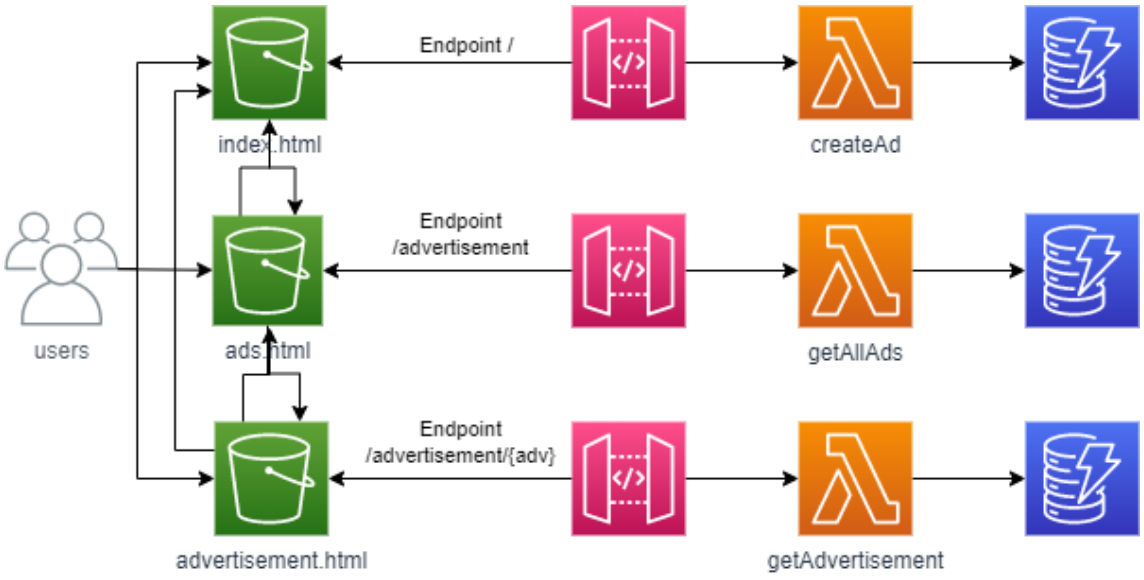

Figura 2.4: Diagrama resumen.

Capítulo 3

Anexo

3.1. Manual de despligue

Con el fin de poder desplegar el proyecto será necesario bajarnos el código de github con un *git clone https://github.com/olgapenedo/web-anuncios.git*.
Como requisitos previos para el despliegue será necesario:

- Cuenta de AWS con permisos de administrador.
- Instalación de AWS CLI.
- Instalación de Node.js.
- Instalación del serverless CLI vía npm: *npm install -g serverless*
- Instalación de uuid y de aws-sdk: *npm install uuid; npm intall aws-sdk*

Posteriormente, ejecutando el comando *aws configure* introduciremos el access key y la contraseña de nuestra cuenta, así como la región que emplearemos, en este caso, eu-west-1.
Una vez realizado esto, podemos desplegar el proyecto mediante el comando *serverless deploy*. Cuando haya terminado, habremos levantado y administrado todos los recursos que se utilizarán.
Si se desea emplear la interfaz, antes de subir los html y los scripts correspondientes, situados en la subcarpeta /html, debemos cambiar el id de la API Gateway en estos últimos. Veamos un ejemplo:
Al realizar el deploy se han obtenido que el id de la API es gs7ddf1m76.

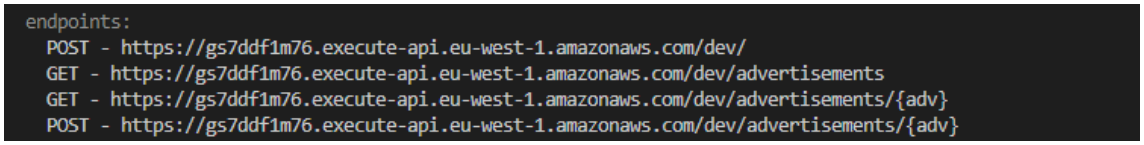

Figura 3.1: Endpoints generados.

Por lo que la variable API_ENDPOINT de los ficheros scripts.js, scripts2.js y scripts3.js deberá tomar el valor *'https://gs7ddf1m76.execute-api.eu-west-1.amazonaws.com/dev/'*. Finalmente, ejecutando el script uploadBucket.sh, alojado también en la subcarpeta /html, se realizará la subida de los ficheros y se podrá acceder a la interfaz en el link `http://advertisements-olga-penedo.s3-website-eu-west-1.amazonaws.com`

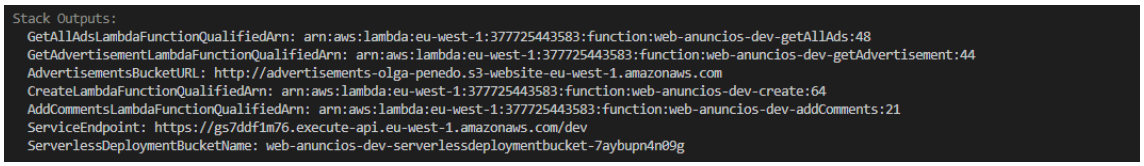

Figura 3.2: URL de la página web.