Administración de Sistemas

GeoMi 

2023-2024

...

Titulación:

Grado en Informática de Gestión y Sistemas de Información

...

4º Curso (1º Cuatrimestre)

...

Página de GitHub

Alan García

3 de diciembre de 2023

Índice

1. Descripción de la aplicación

GeoMi es una aplicación web desarrollada para la asignatura Administración de Sistemas del grado Ingeniería Informática de Gestión y Sistemas de Información de la UPV/EHU. El objetivo de este desarrollo es realizar una correcta configuración de servicios y contenedores Docker partiendo de una aplicación web compuesta por, al menos, tres contenedores:

- Un servidor web
- Un servidor de BBDD
- Un tercer contenedor de libre elección

Por ello, se ha desarrollado una aplicación en la que los usuarios son capaces de registrar sus posiciones por medio del servicio de localización del navegador para elaborar un histórico de geolocalizaciones. Los datos de las localizaciones se presentan de dos formas posibles:

- **Formato de tabla**. Es el formato predefinido de los datos. Es un volcado de los datos del usuario en una tabla. Sin embargo, la sección de *Dirección* no está guardada en la base de datos si no que se hace una llamada al servicio Geocoding de la **API de Google** para obtener una descripción con texto de las coordenadas.

ID	Latitud	Longitud	Fecha	Dirección
1	.....	.....	17/11/2023	Calle....
2	.....	.....	18/11/2023	Calle....
3	.....	.....	19/11/2023	Calle....

Cuadro 1: Formato tabla

- **Formato de mapa**. En este formato se muestra la información de las localizaciones del usuario mediante pines en un mapa. Cada pin tiene una pequeña descripción que indica la *fecha* de la localización y su *id*. Sin embargo, la opción de obtener la dirección en texto de la ubicación no está disponible en este modo.

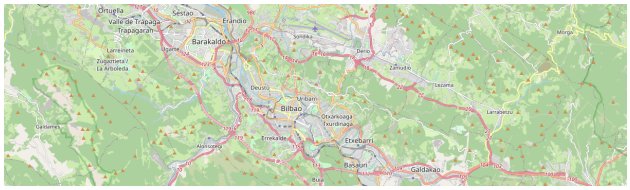

Figura 1: Mapa de Bilbao

Sin embargo, para acceder a todos los datos y a las opciones de añadir nuevas localizaciones, es necesario registrarse e iniciar sesión en la aplicación. Por ello, se ha implementado un sistema de gestión de usuarios con las funciones básicas de Login y Logout. Actualmente, solo está implementado el inicio de sesión por medio de **Google OAuth2** por lo que, para poder tener acceso a la aplicación, es necesario disponer de una cuenta de **Google** (ver sección 4).

Además, esta aplicación tiene integrado un sistema de monitorización mediante **Prometheus**. De esta manera se registran datos del número total de Logins desde la puesta en marcha de la app y los tiempos de procesado de las peticiones por parte del servidor. Para poder visualizar estos datos de forma gráfica, se ha optado por incluir un cliente **Grafana** con un *Datasource* y *Dashboard* configurados para la aplicación.

1.1. Arquitectura de la aplicación

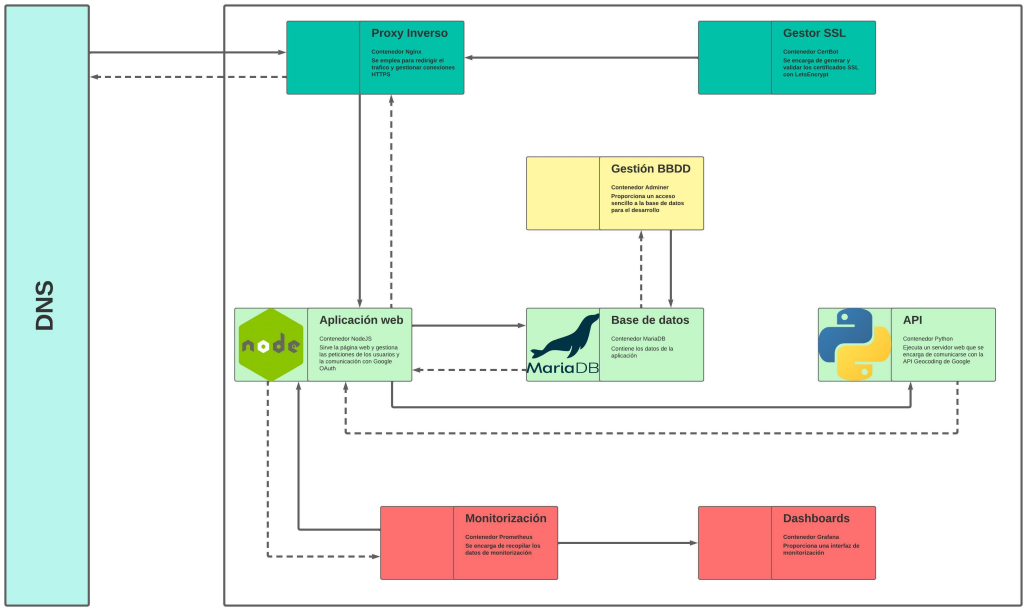

Figura 2: Arquitectura de la aplicación

La base de la aplicación son los tres contenedores destacados en verde. La aplicación web se aloja en un contenedor NodeJS y es el centro de la app. Este contenedor se comunica con la base de datos y con la API de la aplicación. Dicha API es un servidor web en python que procesa peticiones del contenedor NodeJS para transformar coordenadas geográficas en direcciones

de texto haciendo uso de la **API Geocoding de Google**. Las peticiones al contenedor API tienen la siguiente forma:

```
http://geomiapi:3001/api/geocoding/?lat=43.388109&lon=-3.224371
```

Para facilitar el desarrollo, se ha añadido un contenedor *Adminer*. Se trata de un cliente web para la base de datos (contenedor amarillo).

Además, en rojo se destacan los contenedores de monitorización. En primer lugar, tenemos el servidor de *Prometheus* que se encarga de solicitar los datos a la aplicación web y almacenarlos en una base de datos interna. Luego, es el contenedor *Grafana* el que provee de una aplicación web para consumir los datos almacenados en *Prometheus* mediante un *Dashboard*.

En este punto, la aplicación es completamente funcional en un entorno local. Sin embargo, a la hora de realizar el despliegue en un servidor de producción surgen algunos problemas:

- **Google OAuth** sólo permite *URI redirects* a un dominio registrado.
- **La API de geolocalización** del navegador (el acceso a tu ubicación) sólo está disponible desde conexiones HTTPS.

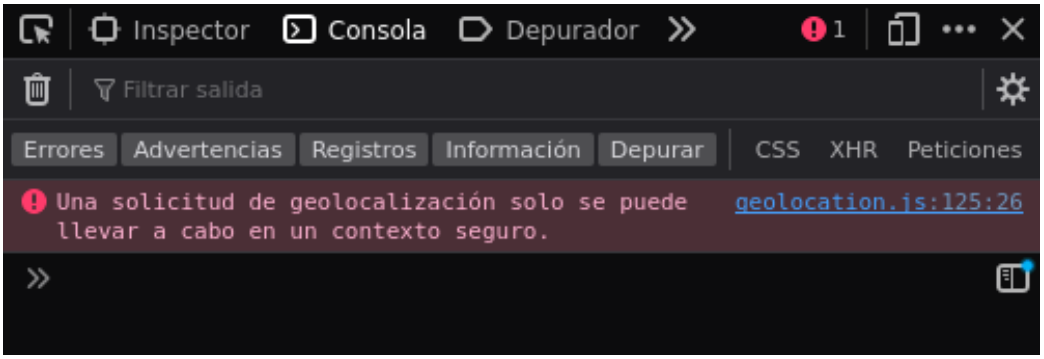

Figura 3: Error en la API de localización

Es por ello que se han introducido dos contenedores extras con el fin de conseguir una conexión HTTPS: un *proxy inverso* y un gestor/proveedor de certificados SSL (contenedores verde oscuro). Además, se ha registrado un dominio DNS para que apunte a la instancia de Google Cloud Computing en la que vaya a realizar el despliegue.

2. Instalación

Se ha desglosado la puesta en marcha de la aplicación en función de la plataforma o medio de instalación.

2.1. Docker Compose

Es el despliegue de los contenedores por medio del sistema *Docker Compose*. Podemos distinguir dos casos de uso: el despliegue de desarrollo *localhost* y el despliegue en una instancia de Google Cloud Computing.

2.1.1. Localhost

Para la puesta en marcha en un entorno local de desarrollo, no es necesario lanzar los contenedores *Nginx* y *CertBot* (de hecho, es recomendable no lanzar *CertBot* para no generar certificados SSL y no saturar la cuenta).

Desde la raiz del repositorio ejecuta el siguiente comando:

```
sudo docker compose up
```

Una vez se hayan descargado las imágenes y dependencias necesarias, se iniciarán los siguientes servicios:

1. localhost:3000/ – Entrada principal de la aplicación web.
2. localhost:3001/ – Servicio de la API de GeoMi. Es el conector a la API de Geocoding de Google.
3. `localhost:3002/` – Servicio de base de datos de MariaDB. Para establecer una conexión con la base de datos se puede utilizar el cliente de mariadb:

   ```
   mariadb --host=localhost --port=3002 -u admin -ptest database
   ```

   Si es la primera vez que se inicia la base de datos y no hay datos previos, se ejecutará el archivo `database.sql` inicializando la base de datos.
4. localhost:3003/ – Servicio web de Adminer. Es un portal web para la gestión de la base de datos de una forma gráfica.
5. localhost:3004/ – Servicio de Grafana. Es un portal que muestra los dashboards de rendimiento de la aplicación.
6. localhost:3006/ – Servicio de Prometheus. Servicio encargado de tomar y registrar todas las métricas que luego se grafican con Grafana.

2.1.2. Servidor Google Cloud Computing

La puesta en marcha en una instancia de GCC puede ser un poco más compleja. Lo primero de todo, hay que asegurar que el dominio DNS apunta a la ip de la instancia. Para ello, hay que actualizar el DNS con la nueva ip y comprobar que se ha actualizado con el comando:

```
nslookup web.alangeomi.com
```

Otro aspecto a tener en cuenta es la generación de certificados SSL mediante *CertBot* y *Let's Encript*. Let's Encript sólo permite un máximo de 5 registros a la hora por día. Por ello, hay que minimizar el número de veces que se generan los certificados.

Atención: Para corregir el despliegue con *Docker* se mantendrá el servidor iniciado en web.alangeomi.com.

Finalmente, solo resta ejecutar el siguiente comando:

```
sudo docker compose up
```

Si es la primera vez que se ejecuta, es posible que haga falta esperar a que se inicie la aplicación y reiniciar los contenedores.

```
sudo docker compose down
sudo docker compose up
```

Con todo ello, la aplicación estará disponible en las siguientes direcciones:

1. https://web.alangeomi.com/ – Aplicación web.
2. http://web.alangeomi.com:3001 – API de GeoMi.
3. http://web.alangeomi.com:3004 – Servicio de Grafana.

2.2. Kubernetes

El despliegue en *Kubernetes* es mucho más limitado. Para simplificar la aplicación, se ha añadido un "bypass.ªl inicio de sesión. Este "bypassçonsiste en iniciar sesión como administrador, de forma que la aplicación web toma los datos del usuario administrador de la base de datos. Así, conseguimos no depender del inicio de sesión mediante **Google OAuth**, ya que es un sistema complejo de integrar y que no responde al alcance de este proyecto.

Sin embargo, tampoco es posible añadir nuevas ubicaciones ya que haría falta una conexión HTTPS (véase la figura 3).

A pesar de ello, sí se ha gestionado la persistencia de la base de datos con la creación de un volumen permanente mediante un *PVC* (Persistent Volume Claim).

Para inicializar la aplicación en Kubernetes hay que ejecutar esta lista de comandos:

```
kubectl apply -f kubernetes/geomi-volumes.yml
kubectl apply -f kubernetes/geomi-deployment.yml
kubectl apply -f kubernetes/geomi-services.yml
kubectl apply -f kubernetes/geomi-ingress.yml
```

Es importante recalcar que estamos haciendo uso del namespace *geomi*. Luego, para listar o acceder a los logs de los pods/servicios/ingress hay que añadir el flag *"n geomi"*:

```
kubectl get pods -n geomi
```

3. Repositorios

El repositorio principal de la aplicación es https://github.com/Strawberryai/geoMi y todas las imágenes que conforman la aplicación están en Docker Hub con su correspondiente documentación:

1. alang6154/geomiweb
2. alang6154/geomiapi
3. alang6154/geomidatabase
4. adminer
5. grafana/grafana
6. prom/prometheus
7. nginx
8. certbot/certbot

4. Problemas encontrados

Como se ha descrito anteriormente, los problemas principales que presenta esta aplicación es la complejidad del despliegue en producción debido a la integración con Google OAuth y al uso de la API de geolocalización del navegador. Esto ha motivado la creación de un sistema bastante más complejo de lo necesario utilizando un proxy inverso, el uso de certificados SSL para crear conexiones HTTPS y la gestión de un nombre de dominio DNS.

Además, la integración del sistema de monitorización fue un reto y no se pudo integrarlo con la base de datos (intenté conectar un contenedor exporter al contenedor de la base de datos sin resultados).

5. Declaración de uso de asistentes basados en AI

Por lo general, no he utilizado asistentes basadoes en IA. Sin embargo, a la hora de implementar el sistema de monitorización, sí que consulté sobre cuales eran las posibles opciones, pero para saber cómo realizar las integraciones siempre recurro a las documentaciones. Empleo mucho portales como *Stackoverflow* cuando surgen errores.

ChatGPT sí que ha sido un apoyo a la hora de utilizar la imagen de CertBot, pero en general creo que es más una distracción ya que muchas veces te proporciona soluciones incorrectas y se pierde mucho más tiempo que buscando en la documentación.

Sin embargo, para transcribir textos de markdown a latex funciona sorprendentemente bien.