Challenger Dev Python Senior

Instrucciones

Se le encarga crear un sistema de registro de infracciones de tránsito en Python. El sistema debe tener las siguientes características:

1. Debe existir una interfaz administrativa, donde puedan manejarse los siguientes registros:
    a. Persona
        i. Campos Nombre y Correo Electrónico
    b. Vehículo
        i. Campos placa de patente, marca, color. Deben estar relacionados con una persona
    c. Oficial
        i. Nombre y un número único identificatorio

La interfaz administrativa debe permitir crear, ver, modificar y borrar registros (debe garantizarse la integridad referencial). Puede construir la interfaz usted mismo, o usar cualquier generador de interfaces que le parezca conveniente.

2. El sistema debe habilitar una API que permita a una App usada por los oficiales de policía cargar una infracción a un vehículo. Cree para esto una API con un método “cargar_infraccion”. Este debe recibir en el body y por método post, un JSON con la siguiente estructura

{

“placa_patente”: “placa patente del vehículo”,

“timestamp”: “marca de tiempo de la infracción”,

“comentarios”: “texto libre”

}

Para la autenticación, la API debe poder generar un token de acceso asociado a cada oficial de policía. Este token debe adjuntarse en Header de la llamada POST, usando la autenticación mediante Bearer Token. Este token puede generarse mediante una interfaz o mediante línea de comandos (en ese caso, documente cómo debe realizarse la llamada).

El llamado API debe devolver status 200 en caso que todo funcione correctamente, 404 en caso de haber algún parámetro no encontrado (patente que no exista) y 500 si hay un error inesperado. Debe devolver también un mensaje de error apropiado.

3. Debe implementar un método API llamado “generar_informe”, el cual debe recibir como parámetro el correo electrónico de una persona, y devolver un JSON con el listado de infracciones de cualquier vehículo a su nombre. No es necesaria autenticación para este llamado.

4. Empaquete la solución en un Virtual Env y suba el código a un repositorio Github

5. Por último, cree una imagen Docker que permita ejecutar todos los componentes de la solución (servidor de apps, bd, etc) y súbala al repo público de Docker, de forma que esta pueda ser descargada y ejecutada fácilmente.

6. Proponga una arquitectura de servicios AWS que sea compatible con el despliegue de esta aplicación en producción. Haga un listado de los servicios que recomendaría y justifique su elección

Notas

- Intencionalmente varios requerimientos son algo ambiguos. Haga los supuestos que estime conveniente para salvar estas dudas.

- Asegúrese que el código de la app y cualquier librería usada sean compatibles con python 3.8 o superior.
- No se evaluará estética ni usabilidad de interfaces, basta que funcionen.
- Escriba este código haciendo uso de las buenas prácticas que considere razonables para este caso.

Entregables:

- URL del repositorio Github con el código. Asegúrese que el repo sea público y que el archivo requirements.txt esté correctamente cargado
- URL de acceso a la imagen Docker de la aplicación
- Documento que contenga:
  - Supuestos
  - En caso de tener comandos por CLI, un ejemplo de como llamarlos
  - Claves de usuarios administrativos de la BD, en caso de haberlos
  - Instrucciones de instalación y ejecución de Docker
  - Arquitectura AWS propuesta.
  - Cualquier otra cosa que considere relevante para evaluar este challenge