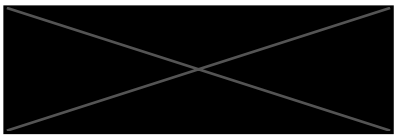


Backend-AWS

DESCRIPCIÓN

Help Center es una empresa con varias sedes en diferentes ciudades donde residen sus colaboradores. La mayoría de las sedes están bajo la modalidad de arrendamiento, y la mayoría de artículos, que de ahora en adelante catalogamos como activos, pertenecen al arrendador.

Sin embargo, algunos de estos activos sí pertenecen a Help Center, y cada administrador de sede realiza un informe mensual con los activos de la compañía detallando su estado. Así mismo, cada vez que se compra un activo nuevo, o se daña o cambia alguno, se debe reportar a la empresa. Help Center registra en un Excel todos los activos, así como su estado.

Los activos no solo están en las sedes: cada trabajador recibe una dotación de equipo al ingresar a la compañía. Estos consisten en un computador portátil, un monitor y algunos otros artículos que no tienen carácter devolutivo, como mouse, teclado, audífonos, etc.

Algunos artículos son muy valiosos para la empresa, y estos tienen incorporados un chip que reporta la localización cada hora. Esto permite lanzar alertas en caso de robo o poder tener una trazabilidad de la localización del objeto e informar a las autoridades correspondientes.

Para Help Center es importante llevar el registro de cada activo y hacer un inventario general cada cierto tiempo.

ACTIVO

Los activos los podemos catalogar según características en común. De ahora en adelante en tres:

- Tecnológico: referente a un activo necesario para prestar labores de desarrollo o relacionados con este. Ejemplo: computador, monitor, video beam, etc.

  Dentro de los activos tecnológicos, tenemos categorías comunes:
  - Computador
  - Monitor

- Mueblería: referente a activos que se usan para el confort y ambiente de los empleados. Normalmente, estos se encuentran en las sedes. Ejemplo: muebles, sillas, mesas, máquinas de café, artículos de aseo, libros físicos, etc.

Dentro de mueblería tenemos categorías comunes:

- Mantenimiento: activos que necesitan abastecimiento o mantenimiento continuo. Ejemplo: la máquina de café necesita alguien que la abastezca, etc.
- Fijo: activo que no necesita mantenimiento alguno.

- Abstracto: referente a los activos no tangibles. Ejemplo: licencias, suscripciones, libros electrónicos, cuentas, etc.

  Dentro de los activos abstractos tenemos la categoría común:
  - Licencia: licencias especiales para el uso de una herramienta o tecnología específica: Por ejemplo, la versión paga de un IDE.

Se debe tener en cuenta que hay artículos que pueden pertenecer a una categoría, pero no a una subcategoría. Ejemplo, un televisor: forma parte de activo tecnológico, pero no es un computador o un monitor.

Todos los activos tienen los siguientes atributos en común:

- Responsable: es la persona o entidad responsable del activo. Un responsable puede ser una persona o una sede.
- Fecha de compra: fecha en la que se hizo la compra o adquisición del activo.
- Número de factura: número de la factura o recibo de pago de la compra del activo.
- Descripción: la descripción del activo. Ejemplo: computador portátil, escoba, licencia de IntelliJ Ultimate Edition.
- Estado: estado del artículo. Ejemplo: operativo, vencido, etc.

Todos los activos tecnológicos tienen los siguientes atributos en común:

- Marca: la marca del artículo como HP, Lenovo, etc.
- Ubicación: dirección de la persona responsable o de la sede.
- Detalle: más información con respecto al activo. Ejemplo: “Equipo robado al empleado. En proceso de investigación de lo sucedido”.
- Fecha de expiración de garantía: la fecha de expiración de la garantía.
- Detalle de la garantía: explicación de lo que cubre la garantía.

Los estados de los archivos tecnológicos únicamente pueden ser: operativo, dañado, reparación, transporte, conflicto y perdido.

La subcategoría “Computador” tiene además los siguientes atributos:

- Procesador: procesador del computador
- RAM: cantidad de memoria RAM
- Disco duro: almacenamiento del computador

La subcategoría “Monitor” tiene el siguiente atributo:

- Pulgadas: cantidad de pulgadas del monitor.

Todos los archivos de “Mueblería” tienen los siguientes atributos en común:

- Fabricante: fabricante del activo: Rimax, Artesanal, etc.
- Ubicación: ubicación del activo.
- Cantidad: cantidad del activo disponible. Aplica por ejemplo cuando son varias sillas, varias escobas, etc.
- Detalle: más información respecto al activo. Ejemplo: “4 de las sillas están partidas”, “Llave extraviada del armario de aseo”, etc.

Los estados de un activo de tipo “Mueblería” son: Operativo, Dañado, Transporte, Perdido, Conflicto. Nótese que son los mismos de los activos tecnológicos, excepto Reparación.

De la subcategoria “Mantenimiento”, tenemos:
- Responsable mantenimiento: Encargado de abastecer o hacer el mantenimiento al activo.

La subcategoría Fijo no tiene atributos extra.

De la categoría Abstracto tenemos los siguientes atributos en común:
- URL: URL del proveedor, recurso o sitio del que se compró el activo.

Los estados de un activo abstracto son: “ACTIVO” e “INACTIVO”

De la subcategoría Licencia tenemos los siguientes atributos en común:

- Proveedor: Nombre de la empresa al que se le compra la licencia.
- Fecha Vencimiento: Fecha en la que se vence la licencia.

  Además, Licencia tiene un estado extra: “VENCIDO”.

RESPONSABLE

Para ejemplo práctico de este reto, digamos que para Help Center existen dos tipos de responsables: Empleado y Sede.

A su vez, los empleados pueden ser dos tipos:

- Empleado: Persona que es empleado de la empresa.
- Colaborador: Persona que contrata Help Center por medio de tercerización o por un tiempo corto o para prestar un servicio. Ejemplo instalar cámaras en una de las sedes.

Todos los empleados tienen los siguientes atributos:

- Nombre completo
- Correo Personal
- Tipo de documento

- Número de documento
- Empresa

Un empleado además tiene los siguientes atributos:

- Tipo de contrato
- Cargo
- Correo empresarial
- Salario

Un Colaborador además de los atributos de empleado tiene también:

- Descripción del Servicio
- Fecha fin del servicio
- Pago total del servicio
- Moneda

Además, una Empresa tiene los siguientes atributos:

- Nombre
- Nit
- Ubicación

Las Sedes, por otra parte, tienen los siguientes atributos:

- Nombre
- Ciudad
- Dirección

RETO

Actualmente, Help Center lleva el inventario y registro de sus activos en un archivo Excel y requiere que se cree una aplicación web que permita la gestión de estos. Para esto ha asignado un equipo para llevar a cabo el reto conformado por:

- Scrum Master
- Product Owner
- Desarrollador Backend
- Desarrollador Frontend
- Líder Técnico

Para este reto asumirás el rol de Desarrollador Backend y deberás implementar la solución diseñada por el líder técnico en conjunto con un arquitecto de Software: Diagrama de Arquitectura

Deberás utilizar LocalStack e IaC (Infraestructura como código) usando CloudFormation o Terraform para crear la infraestructura y simular un ambiente de nube en tu ordenador.

En una instancia de EC2 en AWS deberás utilizar Docker y crear dos microservicios con Java usando spring: Activo y Responsable; así como sus respectivas bases de datos.

- Activo: Encargado de la lógica detrás del registro y obtención de los activos de la empresa. Para este microservicio es mandatorio usar programación reactiva con WebFlux e integrarse a un servicio de mensajería de SQS para recibir las constantes notificaciones de las localizaciones de activos.
- Empleado: Encargado de la información de los empleados (empleados y colaboradores) asociados a Help Center.

Para ambos microservicios se deben crear pruebas unitarias que cubran al menos el 80% de la capa de servicio o donde tengas tu lógica de negocio. Se recomienda usar JaCoCo para la generación del reporte de cobertura.
¿Qué debes probar?

- Cualquier capa donde apliques lógica de negocio o hagas orquestación de operaciones tales como capa de servicio, utilidades, mapeadores, etc. En caso de que repartas lógica entre el controlador, repositorio, manejadores, servicios, etc., también deberás probar estos.

¿Qué puedes excluir de la cobertura de pruebas?

- POJOs tales como modelos, entidades, DTOs, etc.
- Clases para constantes.

Los microservicios deberán exponer una API REST para que el Front y otros servicios puedan interactuar con la lógica del sistema.

Para resolver estas tareas es deseable que tengas experiencia o estés familiarizado con:

- Creación de una API REST usando Java con Spring Boot y JPA.
- Creación de pruebas unitarias con Junit y Mockito.
- Entender los fundamentos de POO y haber trabajado con herencia.
- CRUD con alguna BD NoSQL.
- Entender los fundamentos de la computación en la nube.
- Entender los fundamentos de los contenedores y máquinas virtuales.
- Entender los fundamentos de la programación reactiva.
- Estar familiarizado con las colas de mensajería.
- Uso de variables de entorno

En este reto conocerás o profundizarás en:

- Docker y Docker Compose
- Uso de POO y polimorfismo
- Prácticas de Caché
- Programación Reactiva con Spring WebFlux
- Despliegue de aplicaciones en AWS EC2
- AWS API Gateway
- AWS SQS
- AWS SNS
- AWS Lambda
- AWS Step Functions
- AWS DynamoDB

- AWS S3
- AWS SES
- Infraestructura como código (IaC)
- LocalStack

TAREAS

Nota del autor del reto.

Sí, el software parece inmenso, dado el contexto del problema suena como un proyecto para largo tiempo y con varios profesionales. Sin embargo, recuerda que este es un ejercicio práctico, ten en cuenta lo que nos dice YAGNI. No gastes tiempo de más pensando en todos los requerimientos mencionados en el contexto, ni futuras posibles integraciones.

Céntrate en las tareas, a cumplir los requerimientos sin darle tantas vueltas. Para hacer el reto interesante, traté de ajustar un posible escenario real. Lo que se pide en las tareas es, para mí, lo necesario para que puedas explorar herramientas y paradigmas que se usan normalmente en la industria.

Por último, ten en cuenta que los ejemplos mostrados son solo eso, ejemplos. En tu implementación pueden variar los atributos o la organización siempre y cuando se cumpla lo solicitado por el criterio.

TAREAS SEMANA 1 (Creación de microservicios)

La aplicación deberá permitir realizar las siguientes operaciones:

1. Listar todos los activos de la compañía. Cada activo debe mostrar únicamente la información correspondiente a este. Del responsable únicamente se debe mostrar el correo personal, el nombre completo y la empresa. En caso de ser una sede el responsable solo debe mostrar el nombre y la ciudad. Ejemplo:


```
{
  "activos": [
    {
      "responsable": {
        "correoPersonal": "jhon-doe@example.com",
        "nombreCompleto": "Jhon Doe",
        "empresa": "Help Center"
      },
      "fechaCompra": "22-11-2019",
      "numeroFactura": "1234567-89",
      "descripcion": "Computador portátil",
      "estado": "OPERATIVO",
      "marca": "Lenovo",
```

```
      "ubicacion": "calle falsa 123, Bogotá D.C.",
      "detalle": "Funcional",
      "fechaExpiracionGarantia": "13-04-2027",
      "detalleGarantia": "Daño sobrecalentamiento. No cubre caídas.",
      "procesador": "intel i5",
      "ram": "16GB",
      "discoDuro": "512GB"
    },
    {
      "responsable": {
        "nombre": "Cooworking Caobos",
        "ciudad": "Cúcuta"
      },
      "fechaCompra": "22-11-2016",
      "numeroFactura": "1111111-11",
      "descripcion": "Cafetera",
      "estado": "OPERATIVO",
      "fabricante": "Café Colombia Industries",
      "ubicacion": "calle falsa 123, Cúcuta, Norte de Santander",
      "cantidad": "2",
      "detalle": "Funcionales",
      "responsableMantenimiento":{
        "correoPersonal": "juan-doe@example.com",
        "nombreCompleto": "Juan Doe",
        "empresa": "Help Center"
      }
    },
    {
      "responsable": {
        "correoPersonal": "jane-doe@example.com",
        "nombreCompleto": "Jane Doe",
        "empresa": "Compumundo Hiper Mega Red"
      },
      "fechaCompra": "22-11-2018",
      "numeroFactura": "9876543-21",
      "descripcion": "Licencia IntelliJ IDEA Ultimate edition",
      "estado": "ACTIVO",
      "url": "myexample.com",
      "proveedor": "JetBrains",
      "fechaVencimineto": "13-03-2024"
    }
  ]
}
```

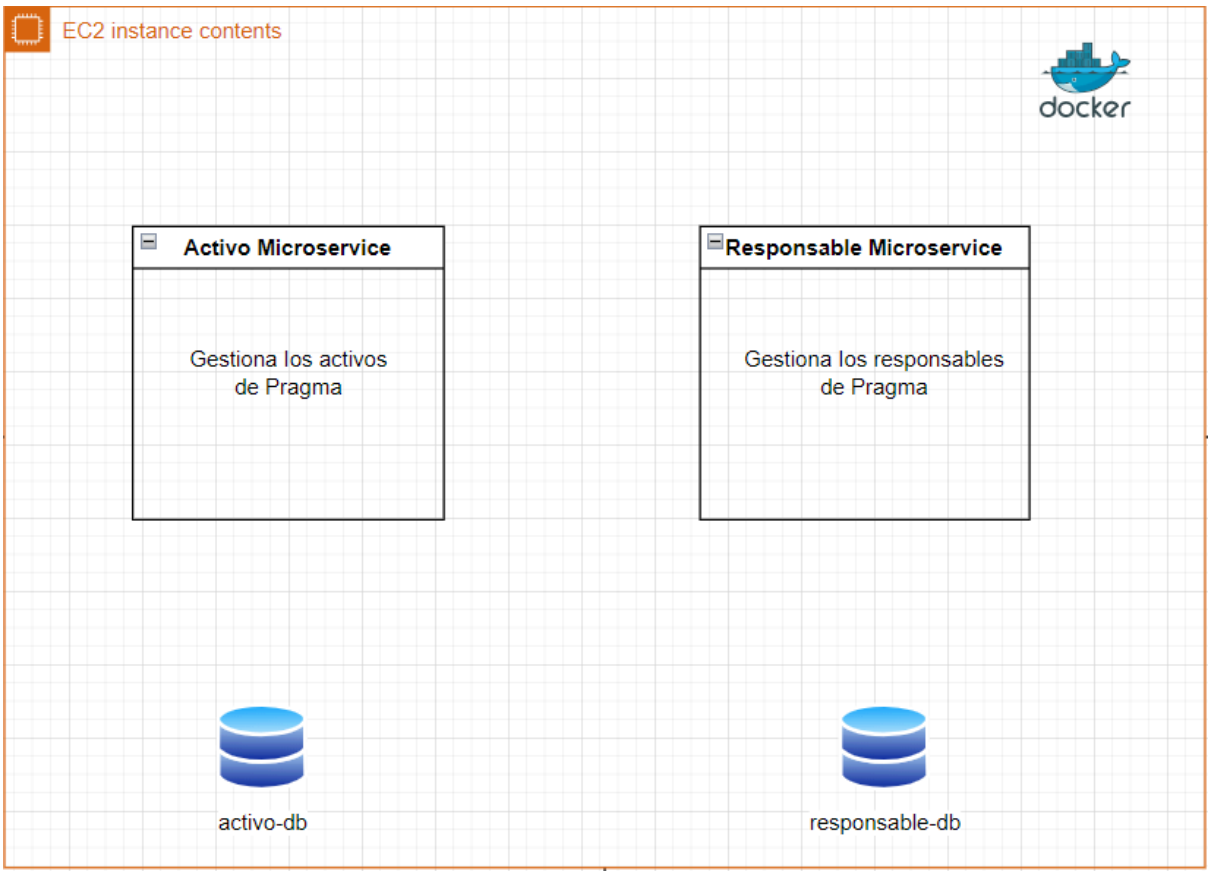
En esta tarea se incentiva el uso de polimorfismo y herencia en general, así como buenas prácticas de POO. Adicionalmente, se considerará el cómo se almacenan los activos en la base de datos dada la herencia. Haz uso de variables de entorno para la integración con el microservicio de Responsables y para la conexión a base de datos.

2. La aplicación deberá permitir el registro de nuevos Responsables, así como la edición de sus datos, obtención por correo y eliminación. También se debe permitir la obtención de responsables por tipo (Sedes y Empleados), así como permitir el filtro por empleados de Help Center y externos.

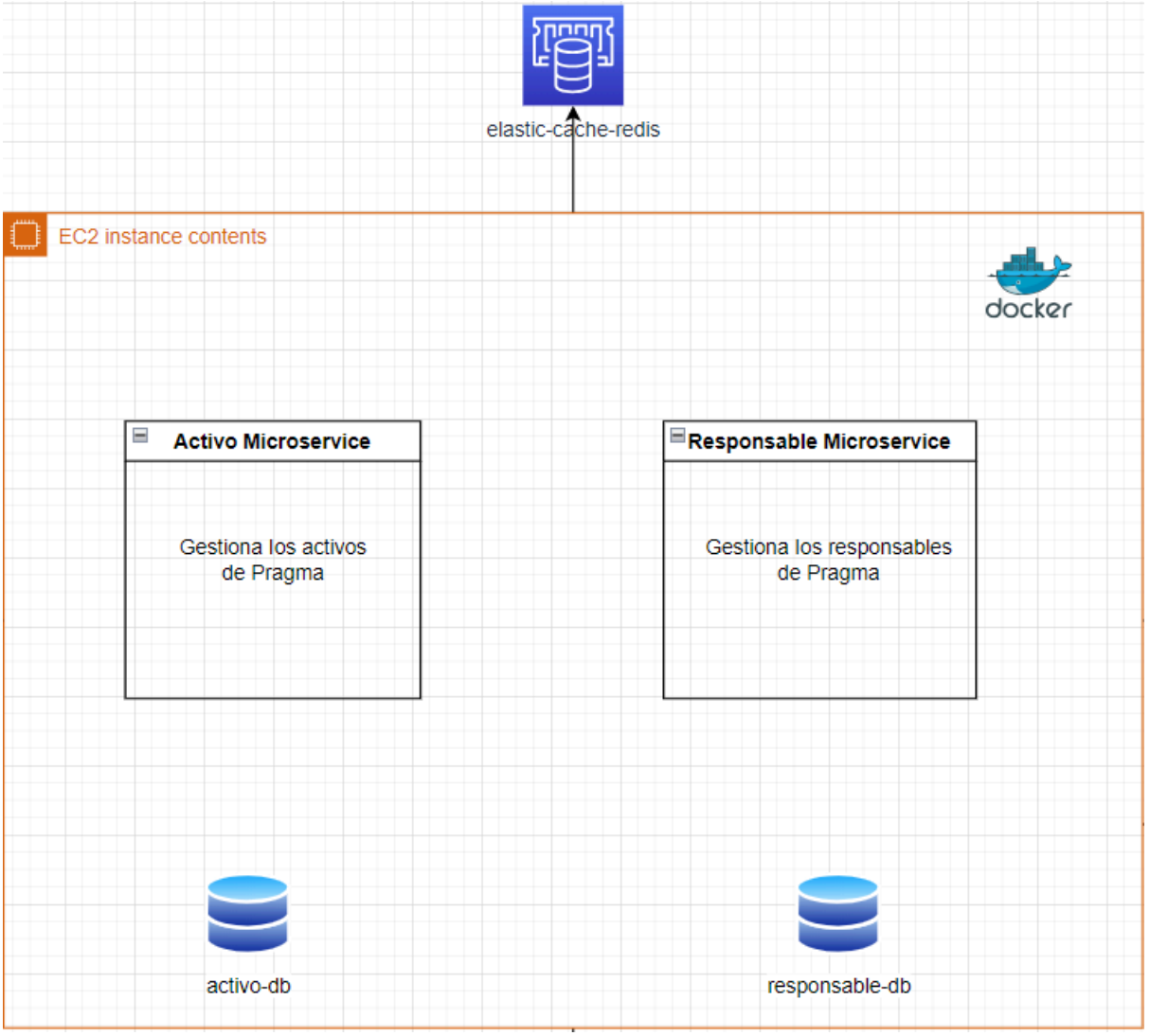
   Nuevamente se incentiva el uso de polimorfismo. Ten en cuenta que la información de los responsables únicamente debe estar en la base de datos de Responsables. El microservicio de Activo solo debe almacenar un identificador único con el que pueda consultar el resto de información de los responsables. Deberás utilizar alguna práctica de caché en Activo para la consulta de responsables. Recuerda que la asignación de responsables no es algo que cambie constantemente. Nuevamente, no olvides usar variables de entorno.

TAREAS SEMANA 2 (Despliegue contenedores EC2)

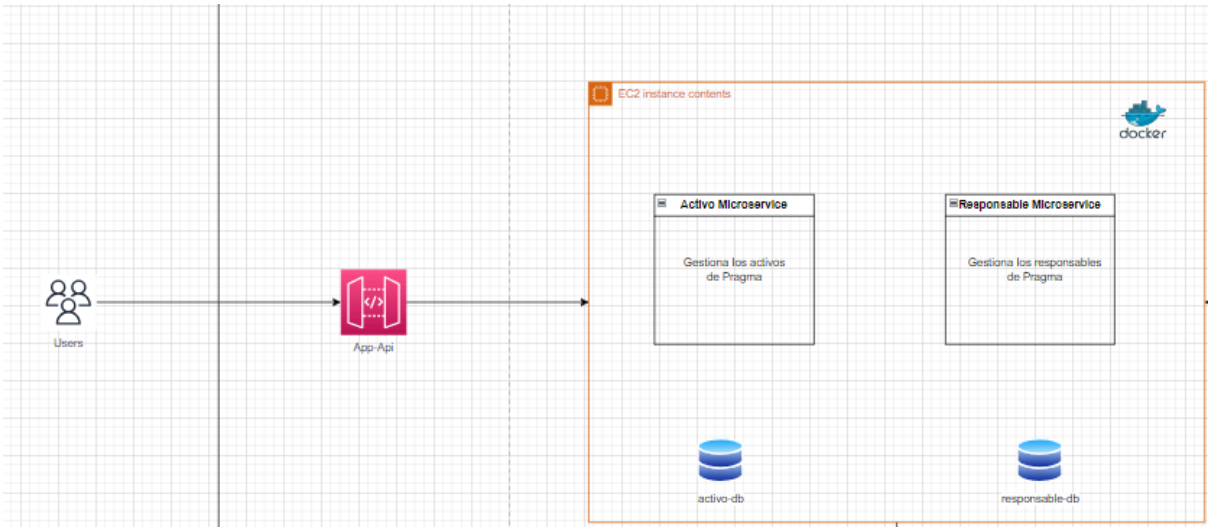
3. Crea una instancia (o más de una si lo consideras necesario) de EC2. Deberás instalar Docker y Docker Compose. Sube tu archivo docker-compose.yml que contenga tus microservicios y bases de datos. Por último corre tu archivo para ejecutar tus micros (asegurate que sea en background).



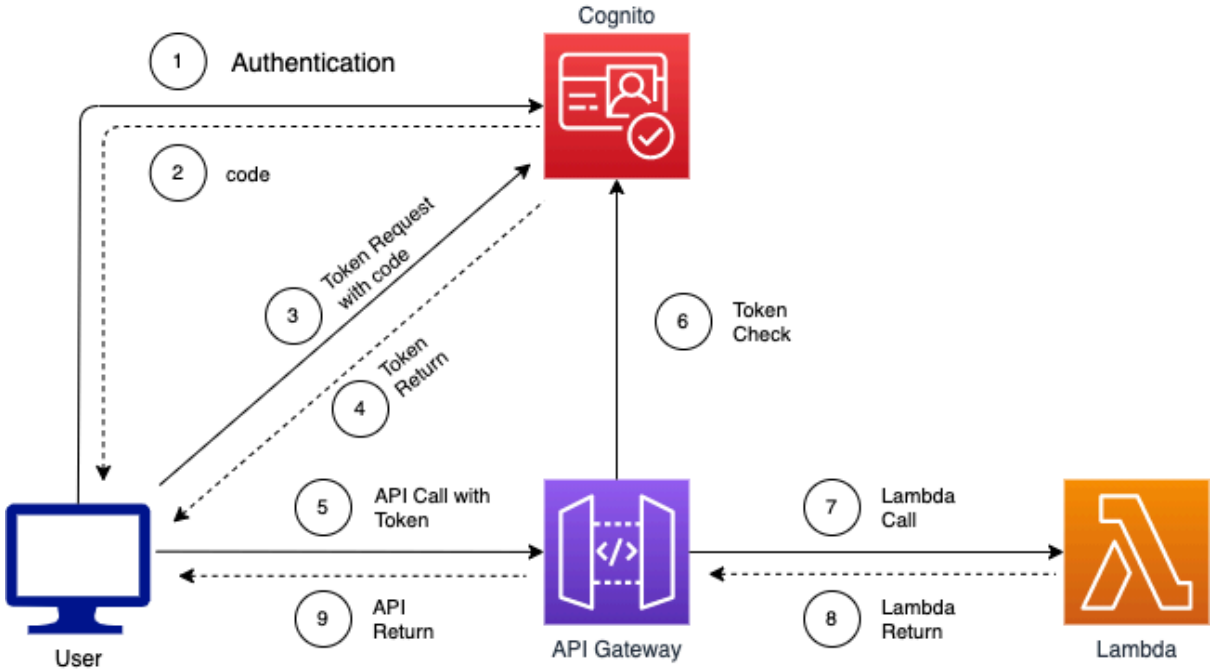
Como estamos usando LocalStack en su versión gratuita, no tenemos acceso a RDS o Aurora para base de datos, ni ECS, ECR, EKS o Fargate para el despliegue de los contenedores, la opción más viable que pensé fue la de EC2.
Sin embargo, si quieres asumir costos de la versión PRO de LocalStack, o directamente trabajando en AWS, responsablemente sabiendo los cobros que conllevan, puedes tener una experiencia más enriquecedora y más basada en contextos reales.
En este caso es más enriquecedor desplegar tus contenedores en ECS o EKS, o si quieres solo un acercamiento sin tanta configuración puedes optar por Fargate. Así podrás también interactuar con VPCs, balanceadores de carga, etc. Y será un ambiente muy parecido al real que encontrarás en las empresas.
Y si no es el caso, no te preocupes, la integración de estos servicios con EC2 es algo también muy común. Además, si nunca has trabajado con EC2, es mejor que hagas el ejercicio normal; EC2 es de los primeros servicios que hay que aprender.
De igual forma, en caso de que decidas usar la versión PRO o directamente en AWS, puedes aprovechar y configurar un clúster de Elastic caché con Redis.



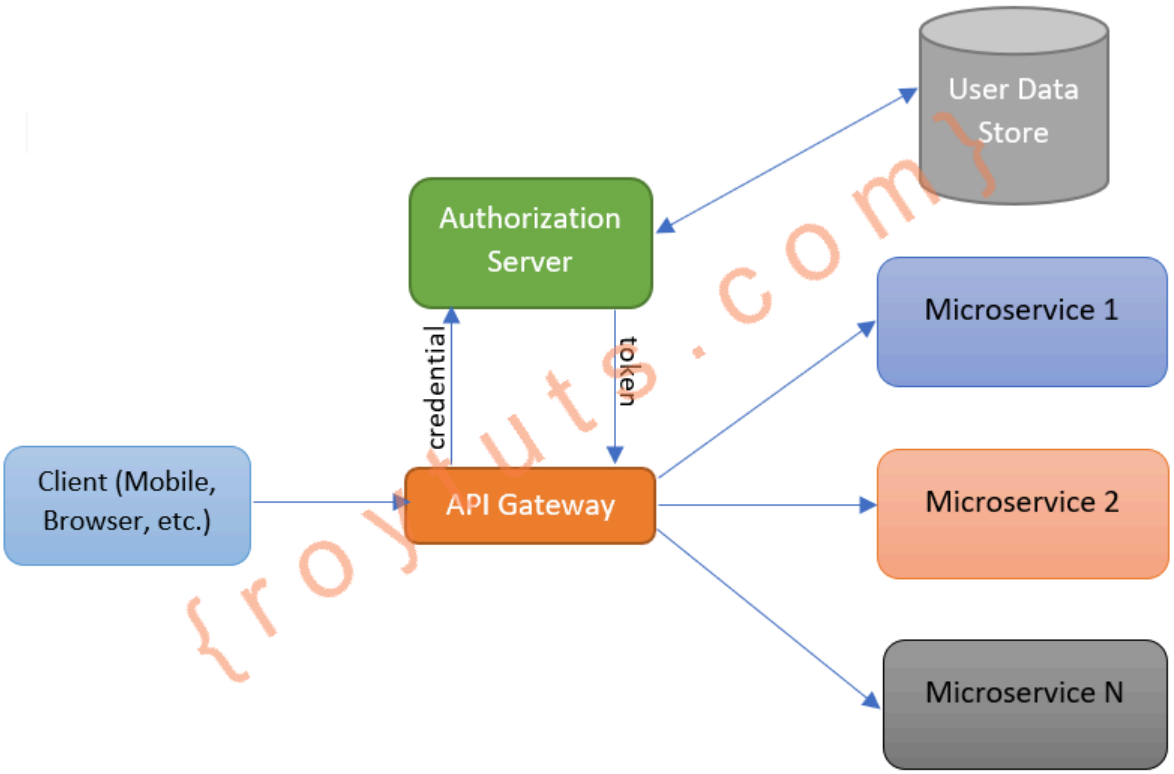
4. Crea un API Gateway de tipo REST y realiza la configuración necesaria para enrutar las peticiones a tus microservicios.



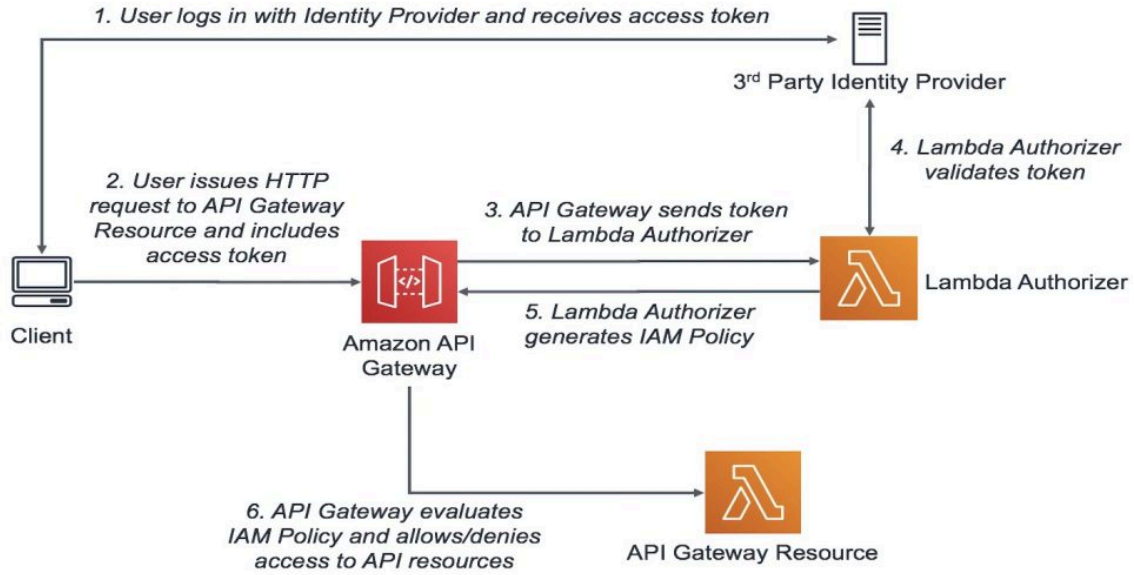
Si vas bien de tiempo, también haz una configuración de CORS y haz algunas pruebas con navegadores web. Y si quieres ir más allá, podrías interactuar con un método de autenticación y/o autorización. (el método o tipo de autenticación lo dejo a tu preferencia, según la tecnología que quieras explorar). Te dejo algunas opciones si nunca has hecho un ejercicio similar:
** Si estás trabajando con la versión PRO de LocalStack o directamente con AWS explora una autenticación con Amazon Cognito.*



** Puedes crear un nuevo microservicio para la autenticación de usuarios con Spring Security y usar JWT para autenticar/autorizar las peticiones de los usuarios.*



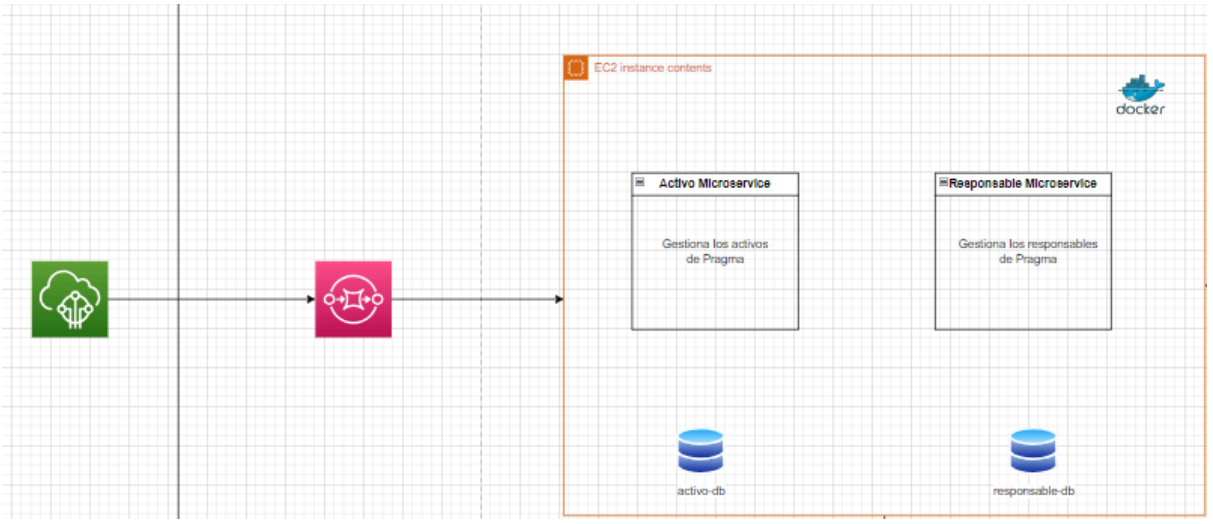
** Utiliza una lambda authorizer para conectarte a un proveedor de identidad y genera una política de acceso.*



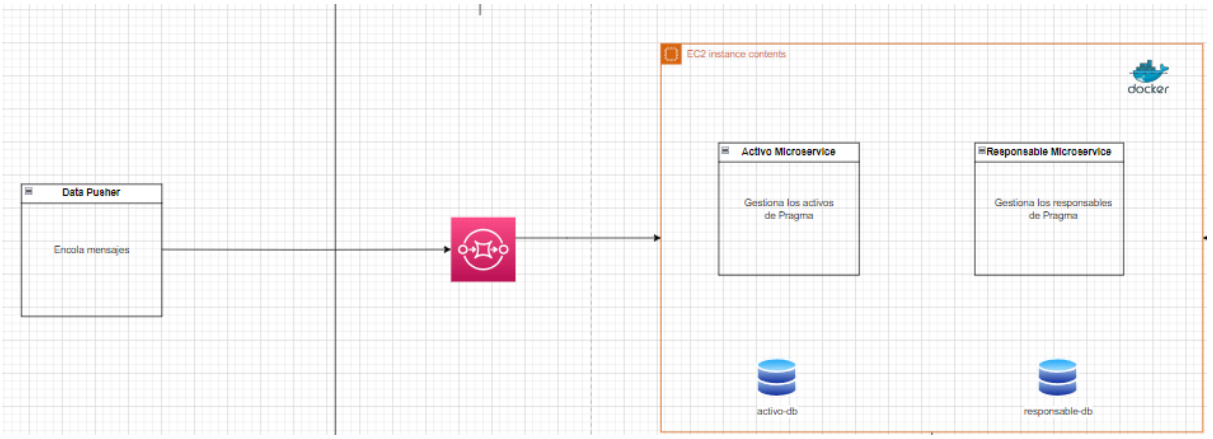
TAREAS SEMANA 3 (Sistemas Reactivos)

5. En esta fase simularás el procesamiento de datos en tiempo real. Para este ejercicio crearás una SQS y harás que el microservicio de Activos se suscriba a este para

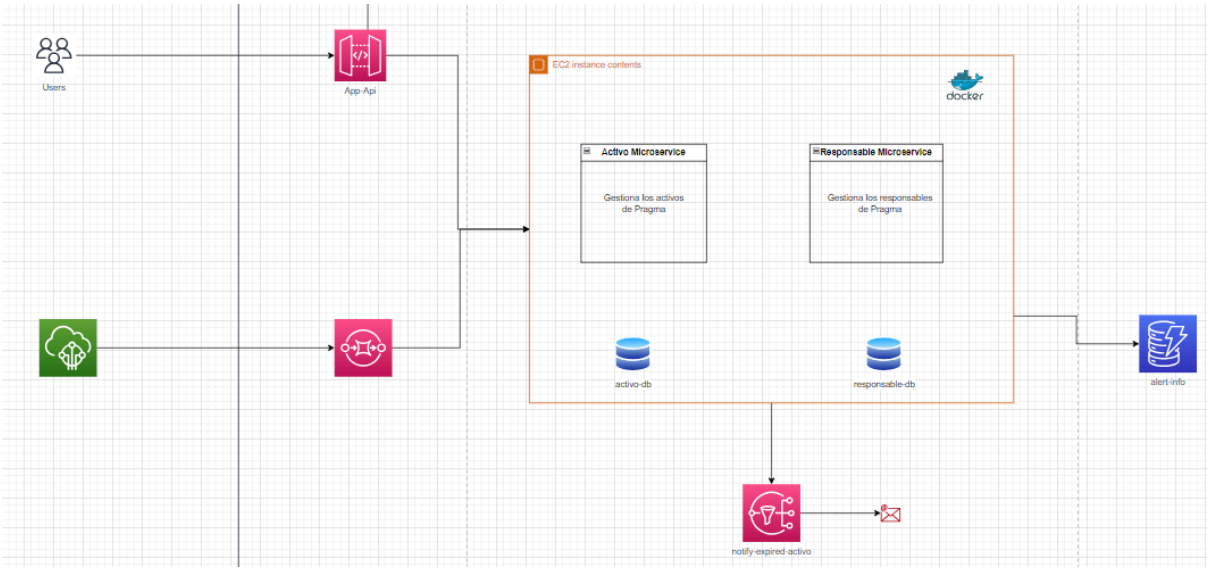
recibir los mensajes. Deberás usar programación reactiva con Spring WebFlux para que el procesamiento de estos datos se haga de una manera concurrente.



En este ejercicio práctico, a falta de chips, crearás una aplicación simple en tu entorno local que simule el envío de señales. Corre simultáneamente algunas instancias de este y encola varios mensajes por segundo durante 5 minutos. Y mientras estás encolando estos mensajes, interactúa con los endpoints del microservicio desde Postman o cualquier otro cliente para verificar que el tratamiento de múltiples solicitudes a la vez no sea bloqueante y el sistema sea elástico y permanezca responsivo.



Una vez recibidos los mensajes deberás mostrarlos en consola por medio de un log. En los mensajes encolados, incluye un atributo booleano “alert”, además del ID del activo. Otros atributos pueden ser la localización, el nombre de la sede, la temperatura, etc.
En caso de que el atributo “alert” sea true, se deberá notificar por medio de SNS al correo del responsable de la sede. Además de esto se deberá almacenar en una base de datos dynamo los mensajes de alerta con la información correspondiente.

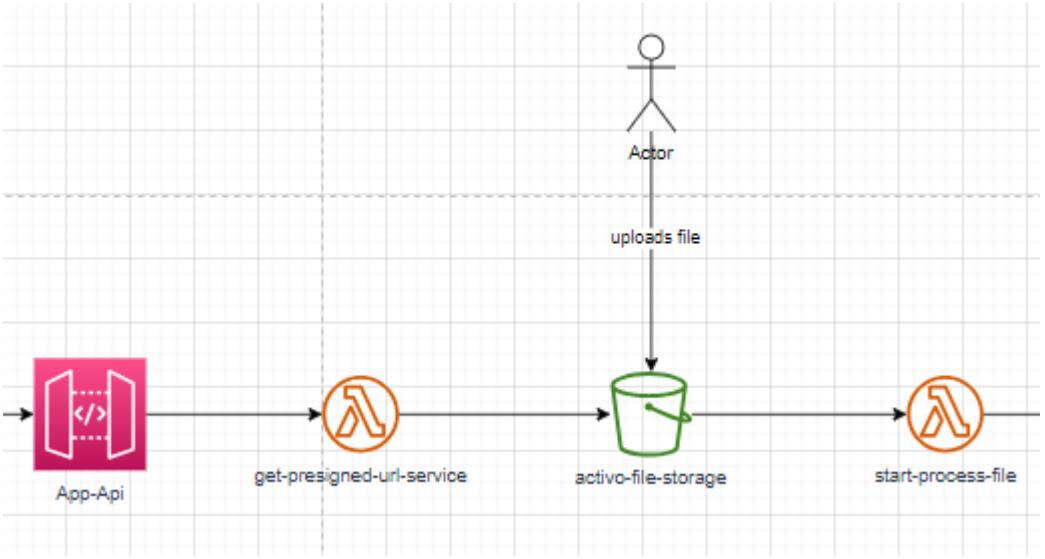


Crea endpoints para que desde el microservicio de Activo se pueda consultar a Dynamo sobre las alertas por el id de un activo, por sede, por correo de responsable o por fecha.

Antes de enviar un correo, recuerda que debes suscribir a los responsables de Sede a las notificaciones de SNS. Explora los índices y las distintas formas de consulta que permite hacer dynamoDb. Si quieres ir más allá, haz que SNS también notifique vía SMS, a otra lambda, o a un canal de Slack o Discord.
Si quieres medir la potencia de usar un enfoque reactivo, te invito a que crees otro microservicio que también se suscriba a la cola de mensajería, pero trabajarlo desde el modo tradicional. Usa CloudWatch o cualquier otro servicio para la medición de rendimiento de los dos micros en cuanto al procesamiento de solicitudes.
Por último, si quieres explorar una herramienta aún más potente para el procesamiento en tiempo real y trabajar con más señales por segundo, reemplaza la SQS por Kinesis Data Streams para que trabajes con flujo de datos.

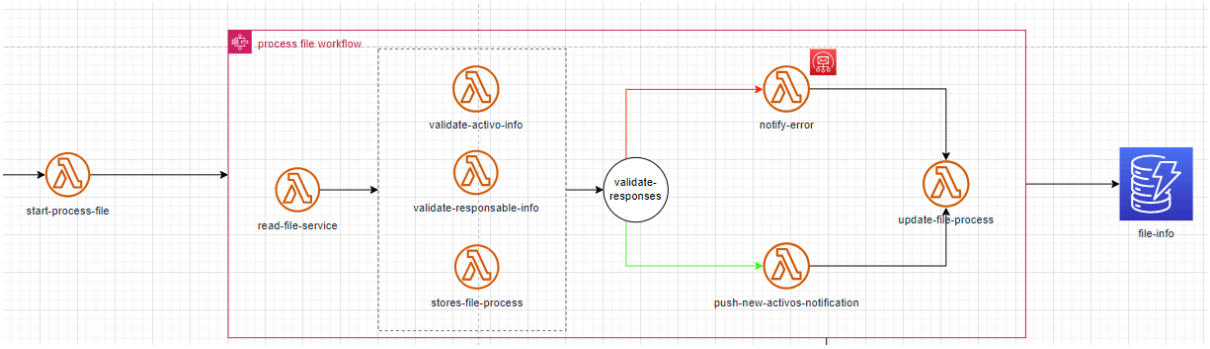
TAREAS SEMANA 4 y 5 (SERVERLESS)

6. Crea un bucket de S3 y dos lambdas. En la primera lambda solicitarás al bucket de S3 una presigned-URL para un archivo .xslx, .xsl o .csv que retornarás como respuesta. Solicita como metadata el correo del usuario que solicita la URL. En el API Gateway que creaste en la tarea 4 crearás un nuevo endpoint para la activación de esta lambda.
   Configura un evento en el bucket de S3 para que cuando un archivo sea subido, una segunda lambda sea ejecutada. En la petición para cargar el archivo, se debe enviar como metadata el correo del usuario que hace la solicitud.



El archivo subido debe tener la información de uno o varios activos nuevos de la compañía, el correo del responsable y los datos del activo.
Antes de crear la plantilla de tu archivo, lee los requerimientos del siguiente paso, para que puedas visualizar los datos claves que deben tener los activos que incluyas en la hoja de cálculo que subas.

7. La segunda lambda del paso anterior (start-process-file) deberás configurarla para que una vez reciba la notificación de que un archivo fue subido, inicie la ejecución de una Step Function de manera asíncrona y envíe la información recibida desde el evento de S3.



Esta step function deberá hacer el procesamiento del archivo: validación de su contenido, guardado de información, notificación de errores y encolamiento de notificación para que sea procesado posteriormente por el microservicio de Activo. A continuación se detallan las lambdas de la step function y sus responsabilidades.

a. read-file-service: A partir de la información recibida, lee el archivo de S3, interpreta su información para convertirla a Json y darla como respuesta. *Te recomiendo usar python para que le saques el jugo a librerías como Pandas o Numpy que te ayudan con la lectura de archivos excel u hojas de cálculo en general.*
b. Proceso paralelo:
    i. validate-activo-info: Consulta el microservicio de Activo para saber si la información consignada en el archivo, con respecto a los activos, es válida. Ejemplo: si es un activo tecnológico de tipo computador tenga la información correspondiente a RAM y Procesador.

ii. validate-responsable-info: Consulta que el correo del responsable consignado en el archivo exista en la base de datos del microservicio de Responsable.
iii. stores-file-process: Almacena en una base de datos dynamo la información de cada activo del archivo y les colocas un estado “inProcess”.

Recuerda el objetivo, no priorices tanto las reglas de negocio, sino más bien sacarle provecho a este ejercicio. Si vas corto de tiempo no gastes de más en la validación de los activos.

c. validate-responses: Es un paso intermedio (No una lambda), para interpretar las respuestas de las lambdas del proceso paralelo. Si todas las respuestas fueron satisfactorias, entonces ejecuta la lambda D. Sino la lambda E.
d. push-new-activos-notification: encola un mensaje a una SQS con los activos nuevos de la compañía. El microservicio de Activos debe estar suscrito a esta cola de mensajería y guardar en base de datos los activos nuevos.
e. notify-error: En caso de que uno o varios activos vengan con errores, se debe notificar por correo usando SES al usuario que cargó el archivo sobre los activos que tienen algún error.
f. update-file-process: Independientemente si los activos fueron encolados satisfactoriamente, o vienen con error, se debe actualizar los registros de estos en dynamo. Cambiando su estado de “inProcess” a “Error” o “Success”. En caso de error se debe colocar la descripción del error. Ejemplo: “El Responsable no existe en la base de datos.”

TAREAS SEMANA 6 (IaC)

8. Si quieres sacarle todo el jugo a este ejercicio, el despliegue de la infraestructura realízalo a través de infraestructura como código. Explora el uso de CloudFormation (preferiblemente para una fácil integración y uso de LocalStack) o Terraform. Idealmente, lo mejor sería que desde el principio uses IaC, esta semana la tienes extra como colchón por esa curva de aprendizaje. O si no quieres correr riesgos, esta semana es para que pases toda la infraestructura que creaste las pasadas semanas a IaC.

Preguntas para reflexionar después de terminar el reto:

1. ¿Qué ventajas tiene usar lambdas en un flujo serverless que se encargue del registro de activos nuevos y no un nuevo microservicio que se encargue de todo el procesamiento? ¿O no sería mejor que el micro de Activo se encargue de todo?
2. ¿Qué ventajas tiene usar servicios como RDS o Aurora a diferencia de tener nuestras bases de datos en contenedores como en este ejercicio? RDS vs Aurora, ¿qué ventajas tiene usar uno u otro?
3. ¿Por qué necesito usar una Step Function para la orquestación de las lambdas y no hacemos el llamado directo entre ellas? ¿Para qué tantas lambdas? ¿No sería mejor agrupar responsabilidades en algunas para reducir la cantidad?
4. SNS vs SES, ¿en qué escenarios es mejor usar uno u otro para la notificación por correo?