ANEXO TÉCNICO
DESARROLLO DE PLATAFORMA MODULAR BASADA EN TECNOLOGÍA CLOUD
"PLATAFORMA GENÉRICA DE COMPRA – FASE 1"

I. ARQUITECTURA

Se espera una arquitectura serverless multi región alojada en la nube de AWS. El frontend debe corresponder a una Single Page Application utilizando de preferencia React.

a. Frontend Web

Se espera que la capa frontend sea multi región. El código de la Single Page Application debe ser alojado en S3 en buckets pertenecientes a regiones AWS distintas y expuesto a Internet mediante CloudFront. Guiarse por la siguiente imagen[1]:

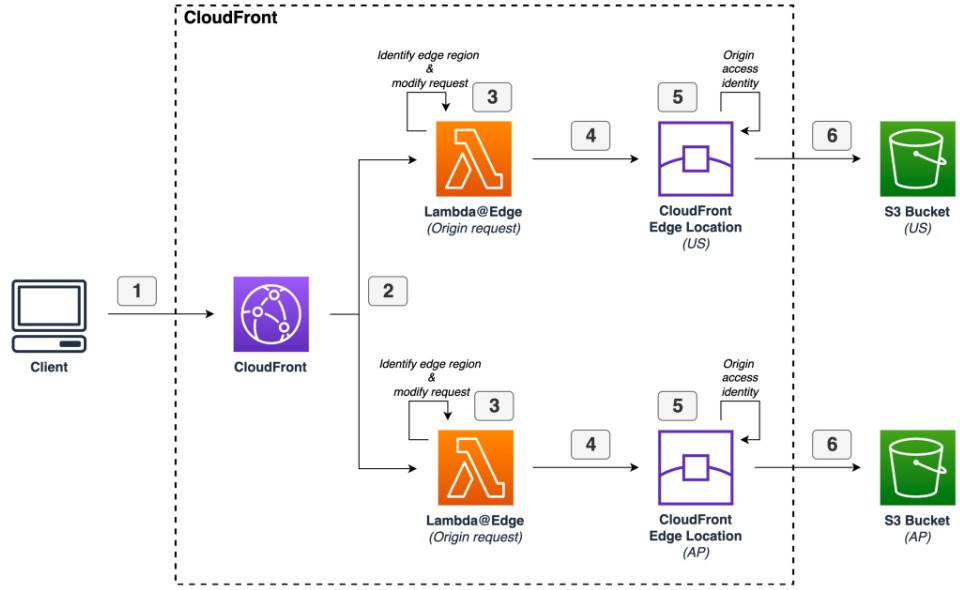

Imagen 1 (Fuente: AWS)

[1] Using Amazon CloudFront and Amazon S3 to build multi-Region active-active geo proximity applications | Networking & Content Delivery // https://aws.amazon.com/es/blogs/networking-and-content-delivery/using-amazon-cloudfront-and-amazon-s3-to-build-multi-region-active-active-geo-proximity-applications/

b. Backend Web

Se espera que el backend de la Single Page Application sea expuesta mediante una API REST a través de Amazon API Gateway e implementada usando tecnología serverless, utilizando servicios como AWS Lambda, Amazon DynamoDB, entre otros. Tomar la siguiente imagen como referencia[2]:

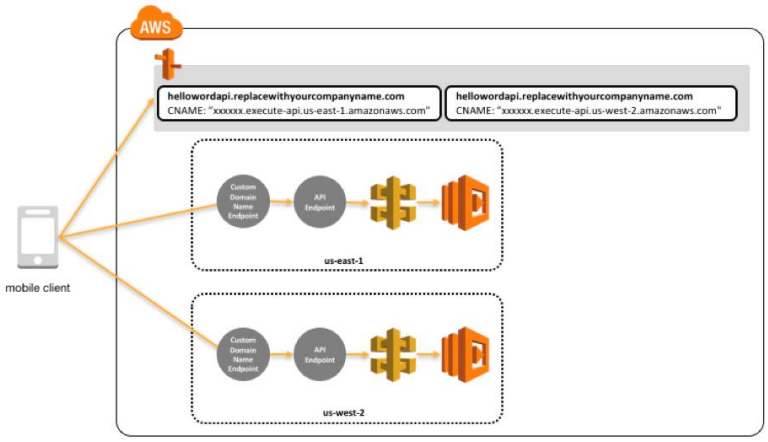

Imagen 2 (Fuente: AWS)

c. Core

Se espera que el Core de la plataforma sea también implementado con tecnología serverless y orientado a eventos utilizando AWS EventBridge como base. Se espera la utilización de AWS Step Functions cuando corresponda. Tomar la siguiente imagen como referencia[3]:

[2] Building a Multi-region Serverless Application with Amazon API Gateway and AWS Lambda | AWS Compute Blog // https://aws.amazon.com/es/blogs/compute/building-a-multi-region-serverless-application-with-amazon-api-gateway-and-aws-lambda/

[3] https://aws.amazon.com/blogs/architecture/lets-architect-designing-event-driven-architectures/

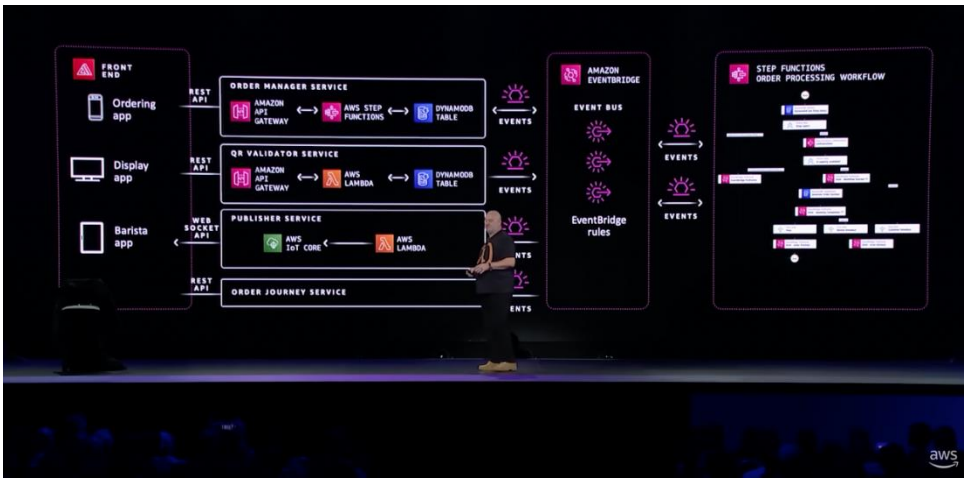

Imagen 3 (Fuente: AWS)

d. Base de Datos

Se espera una separación entre base de datos OLTP (Online Transaction Processing) orientadas a lo transaccional y base de datos OLAP (Online Analytical Processing) orientadas al análisis de los datos y reportería. El core debe hacer uso directo de base de datos OLTP.

Se deben implementar mecanismos de sincronización que permitan copiar los datos transaccionales hacia base de datos OLTP.

Preferentemente, la base de datos OTLP a usar debe ser serverless asegurando de esta forma la escalabilidad del runtime de la aplicación, opciones son AWS DynamoDB, Amazon Aurora Serverless, u otra similar. La base de datos OLTP por su parte debe ser capaz de soportar consultas complejas optimizando costos, opciones son Google Bigquery o Amazon Redshift.

Por otra parte, se espera que la configuración de mecanismos de compra, al igual que las configuraciones de procesos de compra, ofertas y evaluaciones/adjudicaciones tengan una representación textual alojada en S3.

e. Monitoreo y Trazabilidad

Se requiere la configuración de herramientas de monitoreo como CloudWatch y sus Dashboard y Alarmas para conocer en todo momento el estado de toda la infraestructura.

Se espera además contar con mecanismos de trazabilidad de los eventos más importantes del ciclo de vida de mecanismos de compra al igual que del ciclo de vida de procesos de compra, ofertas y evaluaciones/adjudicaciones.

f. Seguridad

La plataforma debe ser diseñada tomando la seguridad como característica fundamental. A nivel WEB se espera la utilización de servicios como Amazon WAF con reglas que mitiguen ataques DDoS, que securiticen formularios de autenticación, entre otros. Se espera recaptcha para formularios públicos. La autenticación se realizará de preferencia mediante el protocolo OpenID Connect.

A nivel backend se espera que las componentes sean construidas con los roles y permisos AWS que aseguren el principio de mínimo privilegio. Todas las APIs expuestas deben asegurar que solo puedan ser consultadas por entes que tengan los permisos para ello.

Se espera que los datos se encripten en tránsito y en reposo utilizando mecanismos correspondientes como uso de HTTPS y bases de datos encriptadas.

II. PROGRAMACIÓN

El framework de preferencia para la programación del frontend es React mientras que el código del backend debe ser implementado de preferencia en Java.

III. API

Se espera que la plataforma exponga una API REST definida usando OpenApi e implementada mediante el uso de HTTP API de Amazon API Gateway integrada con OpenApi (https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/http-api-open-api.html).

La implementación de la API debe ser serverless y multi región utilizando AWS Lambda.

Se espera la implementación de las siguientes APIs:

- **API de Gestión de Compras:**
  - Permite la creación, modificación y eliminación de mecanismos de compra.
  - Endpoints para acceder a los controladores del módulo de gestión de compras.

- **API de Requerimientos de Compra:**
  - Facilita la publicación, consulta y gestión de requerimientos de compra.

  - Permite la integración con otros sistemas para la creación y actualización de requerimientos.

- **API de Recepción de Ofertas:**
  - Permite a los proveedores enviar ofertas en respuesta a los requerimientos de compra publicados.
  - Provee endpoints para la recepción y validación de las ofertas.

- **API de Evaluación/Adjudicación:**
  - Permite la evaluación automática de las ofertas recibidas según los criterios predefinidos.
  - Ofrece endpoints para calcular los puntajes y determinar las ofertas ganadoras.

a. Mensajes y Datos JSON

- **Formato de Mensajes:** Se utilizará JSON para la transmisión de datos entre los diferentes componentes del sistema.
- **Estructura de Datos:** Los datos JSON seguirán una estructura predefinida para cada tipo de mensaje o solicitud, incluyendo información sobre mecanismos de compra, requerimientos, ofertas y resultados de evaluación.

b. Protocolos de Comunicación

- **HTTP/HTTPS:** Se utilizará el protocolo HTTP o su versión segura HTTPS para la comunicación entre los diferentes módulos y con sistemas externos.
- **RESTful:** Las APIs se diseñarán siguiendo los principios de una arquitectura RESTful para garantizar la interoperabilidad y la escalabilidad del sistema.
- **WebSockets (opcional):** Se podría considerar el uso de WebSockets para la comunicación en tiempo real entre los módulos del sistema, especialmente en escenarios donde se requiera una actualización instantánea de datos.
