Guatepass

Justificación de Decisiones Arquitectónicas – Sistema GuatePass

1. Introducción

La arquitectura diseñada para GuatePass responde a la necesidad de procesar transacciones de peaje en tiempo real con baja latencia, alta disponibilidad y costos mínimos, utilizando servicios serverless de AWS.

El sistema debe validar eventos, identificar usuarios, calcular montos, registrar transacciones y enviar notificaciones sin afectar el flujo vehicular en el peaje.

Para lograrlo, se eligió una arquitectura híbrida (síncrona + asíncrona), alineada con los lineamientos del proyecto y con las mejores prácticas de arquitecturas event-driven en AWS.

2. Justificación de la Arquitectura Híbrida

La decisión principal fue separar el flujo en dos fases independientes para optimizar desempeño, resiliencia y confiabilidad.

A. Procesamiento Síncrono (tiempo real, crítico para el peaje)

El peaje necesita una respuesta inmediata (menos de 1–2 segundos) para abrir la barrera.

Este flujo incluye:

- API Gateway recibe el webhook.
- Lambda Validator valida datos, estructura, placa, timestamp y peaje.
- Consulta a DynamoDB para identificar al usuario.
- Determina la modalidad de cobro (tag, registrado, no registrado).
- Publica un mensaje garantizado en SQS.

- Retorna un 200 OK inmediatamente al peaje.

Razones técnicas para hacerlo síncrono:

- La barrera no puede esperar procesos largos como cobros o generación de facturas.
- Se debe rechazar inmediatamente una placa inválida para evitar fraude o inconsistencias.
- El tipo de usuario se debe conocer antes de encolar la transacción para evitar cobros erróneos.
- Garantiza que el peaje reciba confirmación inmediata de recepción.

B. Procesamiento Asíncrono (seguro, escalable, no bloqueante)

El procesamiento completo del cobro no debe retrasar al vehículo.

Por eso ocurre en segundo plano, mediante:

- SQS como cola de mensajes.
- Lambda Processor para ejecutar la lógica de negocio:
  - cálculo de montos,
  - descuentos por tag,
  - recargos por no registrado,
  - simulación de cobro,
  - generación de factura.
- DynamoDB para registrar pagos y facturas.
- SNS para publicar notificaciones.
- Lambda Notifier para simular envíos de email y SMS.

Razones técnicas para hacerlo asíncrono:

- El cobro puede fallar temporalmente; la cola garantiza reintentos sin perder transacciones.
- Las notificaciones no deben afectar el flujo vehicular.

- Aísla el webhook del procesamiento pesado, evitando timeouts.
- Permite escalar horizontalmente sin afectar la experiencia del usuario.

3. Servicios Elegidos y Motivación

API Gateway

- Recibe el tráfico HTTP de forma administrada.
- Integra directamente con Lambda.
- Permite monitoreo, validación y control de requests.

AWS Lambda (Validator, Processor, Notifier)

- Modelo serverless con escalabilidad automática.
- Pago únicamente por ejecución.
- Favorece la separación de responsabilidades entre funciones.

Amazon SQS

- Proporciona un buffer confiable entre validación y procesamiento.
- Reintentos automáticos y garantía de entrega.
- Desacopla los componentes principales del sistema.

DynamoDB

- Base de datos NoSQL con latencias muy bajas.
- Escalabilidad automática.
- Ideal para almacenar usuarios, tags y transacciones.

Amazon SNS

- Permite publicar eventos de notificación.
- Deja abierto el camino para futuros consumidores o canales de comunicación.

CloudWatch

- Centraliza logs y métricas del sistema.
- Permite dashboards personalizados para monitoreo operativo.

4. Alternativas Evaluadas

Alternativa 1: Arquitectura totalmente síncrona

Descartada porque:

- Un cobro o generación de factura puede tardar de 1 a 5 segundos.
- Un timeout bloquearía la barrera y generaría colas en los peajes.
- La generación de notificaciones retrasaría el flujo vehicular.
- No se cumple el requisito académico de separar validación y procesamiento.

Alternativa 2: Arquitectura totalmente asíncrona

Descartada porque:

- El peaje no recibiría validación inmediata del request.
- Una placa inválida se procesaría como válida.
- No se identificaría al usuario a tiempo, generando cobros incorrectos.
- No se cumplirían los requisitos de seguridad operativa del peaje.

5. Beneficios de la Arquitectura Seleccionada

Criterio	Cómo se cumple
Tiempo de respuesta	Validación síncrona menor a 500 ms
Escalabilidad	Lambda, SQS y DynamoDB escalan automáticamente
Resiliencia	SQS reintenta fallos; Lambda puede integrarse a DLQ
Costos	Pay-per-use sin servidores permanentes
Experiencia de usuario	Barrera responde inmediatamente
Mantenibilidad	Componentes desacoplados y especializados
Cumplimiento del proyecto	Satisface todos los lineamientos académicos

6. Conclusión

La arquitectura híbrida seleccionada permite que GuatePass sea un sistema rápido, seguro, económico y altamente escalable.

La separación entre validación síncrona y procesamiento asíncrono es fundamental para cumplir los tiempos operativos del peaje y garantizar confiabilidad en cobros, facturas y notificaciones.

Esta arquitectura cumple plenamente con los requisitos funcionales y técnicos definidos para el proyecto, garantizando un desempeño óptimo bajo carga real y una implementación eficiente dentro del ecosistema serverless de AWS.