SISTEMA GUATEPASS

PROYECTO FINAL

Sistema de Cobro Automatizado de Peajes para la Ciudad de Guatemala

---

MODALIDAD DE TRABAJO

- **Grupos de 3 estudiantes**
- Cada grupo debe organizarse para distribuir tareas de forma equitativa
- Todos los integrantes deben participar en la presentación final

---

1. CONTEXTO

La ciudad de Guatemala necesita modernizar su sistema de cobro de peajes implementando una solución serverless escalable que permita procesar transacciones de forma automatizada según el perfil de cada usuario.

2. DESCRIPCIÓN DEL SISTEMA

GuatePass es un sistema inteligente de cobro de peajes que opera bajo tres modalidades de pago según el nivel de registro del usuario:

Modalidad 1: Usuario No Registrado (Cobro Tradicional)

- El sistema de cámaras detecta la placa del vehículo
- Se valida si el propietario existe en los registros de tránsito
- Si tiene correo electrónico o teléfono registrado, se le envía invitación para registrarse en GuatePass
- Se genera una factura simulada con cargo premium más multa por pago tardío

Modalidad 2: Usuario Registrado en App (Cobro Digital)

- Al detectar la placa, el sistema identifica automáticamente al usuario registrado
- Realiza el cobro instantáneo a su método de pago asociado

- Envía notificación de cobro por correo electrónico o SMS

Modalidad 3: Usuario con Dispositivo Tag (Cobro Express)

- El usuario posee un dispositivo físico Tag GuatePass instalado en su vehículo
- El sistema detecta el Tag y realiza el cobro según la configuración del usuario en la app
- Es el método más rápido y eficiente de procesamiento

3. ESPECIFICACIONES TÉCNICAS

Input del Sistema

Se recibirá un webhook con la siguiente información de cada transacción:

```json
{
  "placa": "P-123ABC",
  "peaje_id": "PEAJE_ZONA10",
  "tag_id": "TAG-001",
  "timestamp": "2025-10-29T14:30:00Z"
}
```

Nota:

- El campo `tag_id` puede ser `null` o estar vacío si el vehículo no tiene Tag instalado
- El monto del peaje debe ser determinado por el sistema según el `peaje_id` y aplicar recargos según la modalidad del usuario
- La presencia de un `tag_id` válido indica que el vehículo tiene dispositivo Tag activo

Datos Iniciales Proporcionados

- Un archivo CSV con la base de datos de clientes registrados que debe ser cargado al sistema
- El estudiante debe diseñar y proponer la estructura de la base de datos

Comportamiento Esperado del Sistema

El sistema debe:

1. Recibir el evento de paso por peaje
2. Validar el estado del vehículo/usuario
3. Determinar el monto del cobro según el peaje y modalidad del usuario
4. Aplicar la lógica de negocio correspondiente

5. Procesar el cobro o generar la factura simulada
6. Enviar la notificación apropiada (SMS o correo electrónico simulado)
7. Registrar la transacción en el historial

4. SUPUESTOS DEL SISTEMA

1. **Estructura de la Base de Datos:** El estudiante debe diseñar y proponer la estructura de las tablas/entidades necesarias para el sistema

2. **Datos Iniciales:** Se proporcionará un archivo CSV con clientes que debe ser cargado en el sistema al momento del despliegue

3. **Aplicación Móvil:** NO se debe desarrollar una aplicación móvil. Se asume que ya existe una app funcional y que los usuarios pueden registrarse y configurar sus preferencias a través de ella

4. **Facturación:** NO se debe generar factura real del SAT. Se debe simular la generación de facturas con los datos básicos (número de factura, fecha, monto, concepto, datos del contribuyente)

5. **Notificaciones:** Los envíos de SMS y correos electrónicos pueden ser simulados usando servicios en modo sandbox o logs del sistema

6. **Métodos de Pago:** La integración con procesadores de pago reales (tarjetas de crédito/débito) debe ser simulada. Cada usuario tiene un solo método de pago asociado.

5. ENTREGABLES

5.1 API Endpoints Funcionales

Deben implementarse los siguientes endpoints:

Endpoints de Transacciones:

- `POST /webhook/toll` - Recepción del evento de paso por peaje
- `GET /history/payments/{placa}` - Consulta de historial de pagos por vehículo
- `GET /history/invoices/{placa}` - Consulta de historial de facturas

Endpoints de Tags:

- `POST /users/{placa}/tag` - Asociar Tag al vehículo
- `GET /users/{placa}/tag` - Consultar información del Tag

- `PUT /users/{placa}/tag` - Actualizar configuración del Tag
- `DELETE /users/{placa}/tag` - Desasociar Tag

Todos los endpoints deben incluir documentación clara del formato de request y response esperado.

5.2 Infraestructura como Código (IaC)

Obligatorio: Todo el sistema debe estar definido como código usando una de las siguientes opciones:

- AWS SAM (Serverless Application Model)
- AWS CloudFormation
- Scripts en Python usando Boto3

La infraestructura debe incluir:

- Definición de todos los servicios serverless utilizados
- Configuración de permisos IAM
- Definición de bases de datos
- Configuración de API Gateway
- Recursos de monitoreo y logging

5.3 Repositorio Git

Estructura mínima requerida del repositorio:

```
None
/
├── README.md                    # Instrucciones completas
├── infrastructure/              # Templates de IaC
│   ├── template.yaml           # SAM/CloudFormation
│   └── deploy.py               # Script de despliegue (opcional)
├── src/                        # Código fuente
│   ├── functions/              # Lambdas
│   └── layers/                 # Lambda Layers (opcional)
├── data/                       # Datos iniciales
│   └── clientes.csv
├── docs/                       # Documentación
│   ├── arquitectura.png        # Diagrama
│   └── api-docs.md             # Documentación de API
└── tests/                      # Pruebas (opcional)
```

README.md debe incluir:

- Descripción general del proyecto
- Prerrequisitos (AWS CLI, SAM CLI, credenciales, etc.)
- Instrucciones paso a paso para desplegar la arquitectura
- Instrucciones de uso del sistema (cómo probar los endpoints)
- Ejemplos de requests con curl o Postman
- Guía para la carga inicial de datos del CSV
- Información sobre el monitoreo y logs

5.4 Dashboard de Monitoreo con CloudWatch

Debe implementarse un dashboard básico que incluya:

Métricas Principales:

- Lambda Functions: invocaciones, errores, duración
- API Gateway: número de requests, latencia, errores 4xx/5xx
- DynamoDB: operaciones de lectura/escritura, throttles (si aplica)

Logs Centralizados:

- CloudWatch Logs para todas las funciones Lambda
- Log groups organizados por componente

5.5 Diagrama de Arquitectura

- Diagrama técnico detallando todos los componentes AWS serverless utilizados
- Flujo de datos entre componentes con indicaciones claras
- Justificación escrita de decisiones arquitectónicas (mínimo 1 página)
- Puede usar herramientas como: Draw.io, Lucidchart, CloudCraft, o diagrams.py

5.6 Presentación Final

- Slides explicando la solución
- Demo en vivo del sistema funcionando
- Pruebas de los tres escenarios de cobro
- Demostración de endpoints de transacciones y tags
- Visualización del dashboard de monitoreo
- Tomar en cuenta sean 25min de presentacion y 5 min para Q&A

6. LINEAMIENTOS Y RESTRICCIONES

Obligatorio:

- Utilizar exclusivamente componentes serverless de AWS
- Los componentes deben ser seleccionados del stack tecnológico visto en clase y workshops
- Implementar mejores prácticas de arquitectura cloud
- Todo debe estar en IaC (Infrastructure as Code)
- Repositorio Git con documentación completa
- Dashboard de monitoreo funcional

Decisión Arquitectónica Clave:

El estudiante debe determinar y **justificar técnicamente** si implementará:

- **Arquitectura Síncrona**: Respuesta inmediata al webhook
- **Arquitectura Asíncrona**: Procesamiento diferido mediante eventos
- **Arquitectura Híbrida**: Combinación de ambos enfoques

La justificación debe considerar:

- Tiempo de respuesta requerido
- Escalabilidad del sistema
- Resiliencia ante fallas
- Costo operativo
- Complejidad de implementación
- Experiencia del usuario

La calidad de esta justificación es un componente crítico de la evaluación.

7. CRITERIOS DE EVALUACIÓN

Criterio	Peso	Descripción
Funcionalidad del Webhook	25%	Sistema procesa correctamente los tres escenarios de cobro
Arquitectura Serverless	25%	Diseño apropiado, uso correcto de servicios serverless
Justificación Técnica	20%	Argumentación sólida de decisiones arquitectónicas (síncrono/asíncrono)
Infraestructura como Código	15%	IaC completo, desplegable, bien estructurado
Documentación	10%	README completo, instrucciones claras, código documentado
Presentación y Demo	5%	Claridad en la explicación, demo funcional en vivo

8. COMPONENTES SERVERLESS SUGERIDOS

Servicios AWS Serverless Vistos en Clase:

- **AWS Lambda** - Procesamiento de la lógica de negocio
- **API Gateway** - Exposición de endpoints REST
- **Amazon DynamoDB** - Base de datos NoSQL serverless
- **Amazon SQS** - Colas de mensajes para procesamiento asíncrono
- **Amazon SNS** - Notificaciones y pub-sub
- **Amazon EventBridge** - Bus de eventos para arquitecturas event-driven
- **AWS Step Functions** - Orquestación de workflows (opcional)
- **Amazon CloudWatch** - Monitoreo, métricas y logs
- **AWS SAM / CloudFormation** - Infraestructura como código
- **IAM Roles y Policies** - Gestión de permisos

Nota: El estudiante debe seleccionar los servicios apropiados según su diseño arquitectónico y justificar su elección.

9. RECURSOS Y HERRAMIENTAS

Herramientas Recomendadas:

- AWS CLI
- AWS SAM CLI (si usa SAM)
- Postman o Thunder Client para pruebas de API
- Git para control de versiones
- VS Code con extensiones de AWS
- Draw.io o Lucidchart para diagramas

Referencias de Clase:

- Workshop de Serverless Patterns
- Módulos de Lambda, API Gateway, DynamoDB
- Ejemplos de arquitecturas síncronas y asíncronas
- Patrones event-driven con EventBridge

10. FECHAS IMPORTANTES

- **Fecha de entrega completa:** 17 de noviembre de 2025, 11:59pm
- **Fecha de presentaciones:** 17 de noviembre de 2025

Nota: La entrega del proyecto y las presentaciones se realizarán el mismo día.

11. EJEMPLO DE ESTRUCTURA CSV DE CLIENTES

```
None
placa,nombre,email,telefono,tipo_usuario,tiene_tag,tag_id,saldo_d
isponible
P-123ABC,Juan
Pérez,juan@email.com,50212345678,registrado,false,,100.00
P-456DEF,María
López,maria@email.com,50298765432,registrado,true,TAG-001,250.00
P-789GHI,Carlos Ruiz,,,no_registrado,false,,0.00
P-111JKL,Ana
Torres,ana@email.com,50245678901,registrado,false,,75.00
P-222MNO,Luis García,,50267890123,no_registrado,false,,0.00
```

```
P-333PQR,Sofia
Morales,sofia@email.com,50256781234,registrado,true,TAG-002,150.0
0
```

Campos:

- `placa`: Identificador único del vehículo
- `nombre`: Nombre del propietario
- `email`: Correo electrónico (puede estar vacío para no registrados)
- `telefono`: Número de teléfono (puede estar vacío)
- `tipo_usuario`: "registrado" o "no_registrado"
- `tiene_tag`: true/false indicando si tiene dispositivo Tag
- `tag_id`: Código del Tag (vacío si no tiene)
- `saldo_disponible`: Saldo en quetzales para realizar cobros

¡Éxito en el proyecto!