
Documento Específico de la Célula 8

Recepción y Agregación de Órdenes de Compra de Gadgets – Acme Inc.

Proyecto: Arquitecturas Serverless en AWS
Célula: 8
Integrantes: Marcos Fabián Flores, Waldemar Sosa
Versión: 1.0
Organización: Universidad La Salle – Facultad de Ingeniería
Arquitectura Asignada: Triggers para agregación de datos
Caso de uso: Sistema de ingestión y agregación de órdenes de compra de gadgets de Acme Inc.

Índice

---

1. Introducción

La Célula 8 es responsable del diseño e implementación del sistema serverless que recibe, procesa y agrega todas las **órdenes de compra de gadgets** realizadas en los sistemas de Acme Inc.

Este módulo se integra directamente con el **e-commerce** diseñado por la Célula 4 y con procesos de facturación, inventarios y analítica.

El sistema debe operar exclusivamente mediante eventos serverless, garantizando consistencia, seguridad y una capacidad de agregación eficiente.

---

2. Objetivo del Módulo

Crear una solución completamente serverless que permita:

- Recibir órdenes de compra provenientes del e-commerce
- Procesar cada orden en forma automática
- Generar **agregados** útiles para:
    - analítica
    - dashboards
    - inventarios
    - proyecciones de ventas

- Publicar datos consistentes y cifrados
- Exponer APIs seguras para consultas

---

3. Alcance

Incluye:

- Recepción de órdenes (input event-driven)
- Procesamiento y validación
- Agregación por:
    - gadget
    - categoría
    - comprador
    - periodo (día / semana / mes)
- Pipeline CI/CD
- Pruebas con curl
- Integración con e-commerce (Célula 4)

No incluye:

- Procesamiento avanzado de pagos
- Integración con ERP real
- UI para dashboards (solo APIs o datos almacenados)

---

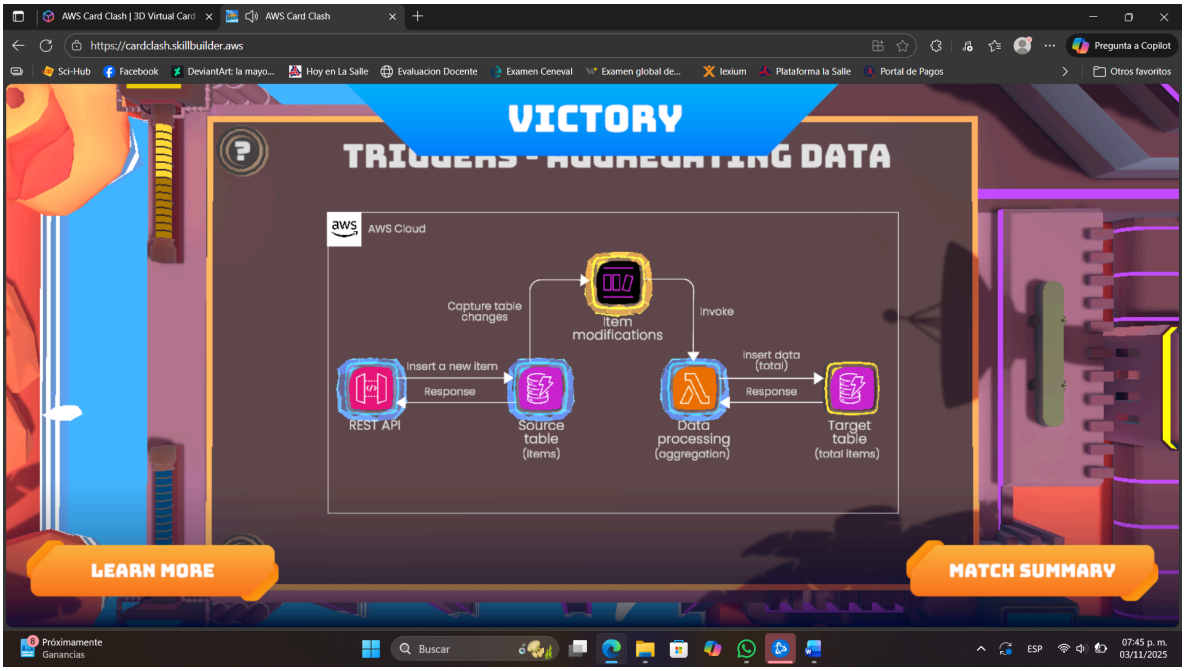
4. Descripción del Proceso de Órdenes de Compra

Cada vez que un usuario finaliza la compra de uno o varios gadgets:

1. El e-commerce genera una **orden**.
2. La orden se publica como evento en Acme Event Bus (EventBridge) o se almacena en DynamoDB.
3. Este evento dispara el sistema de la Célula 8.
4. El sistema:
    - valida la orden
    - registra la compra
    - agrega datos por gadget
    - agrega datos por rango temporal
    - actualiza métricas y totales
5. Los datos agregados se guardan en DynamoDB o S3.
6. APIs permiten recuperar el estado de las ventas en tiempo real.

---

5. Arquitectura General



El flujo recomendado es:

1. **Orden creada** en el e-commerce (Célula 4).
2. Se emite un evento hacia:
    - DynamoDB Stream **o**
    - EventBridge custom bus **o**
    - S3 ingestion bucket
3. **Lambda OrderProcessor** recibe el evento.
4. Procesa la orden:
    - Orden única
    - Múltiples gadgets (items)
5. **Lambda Aggregator** actualiza:
    - totales por gadget
    - totales por categoría
    - métricas temporales

6. Los agregados se almacenan en:
    - `OrdersAggregatesByGadget`
    - `OrdersAggregatesDaily`
    - `OrdersAggregatesMonthly`
7. API Gateway expone endpoints para consultar:
    - ventas por gadget
    - ventas por día
    - ventas por mes

---

6. Componentes AWS Utilizados

- **Amazon EventBridge** o **DynamoDB Streams**
- **AWS Lambda (Python)**
    - OrderProcessor
    - Aggregator
    - QueryAPI
- **Amazon DynamoDB**
    - Tabla `OrdersRaw`
    - Tabla `OrdersAggregatesByGadget`
    - Tabla `OrdersAggregatesDaily`
- **Amazon S3**
    - Opcional para almacenamiento histórico

- **Amazon API Gateway**
    - Endpoints REST para consultas
- **Amazon Cognito**
    - Autenticación JWT
- **AWS KMS**
    - Cifrado
- **VPC Endpoints + zona de inspección**
- **CloudWatch Logs**
- **CodePipeline / CodeBuild**

---

7. Lineamientos Técnicos Específicos

- Las órdenes deben procesarse **automática e idempotentemente**.
- El sistema debe registrar **todas** las órdenes recibidas.
- El procesamiento debe manejar órdenes con múltiples gadgets.
- La agregación debe ser:
    - rápida
    - consistente
    - precisa
- Debe permitir consultas en tiempo real con baja latencia.
- La información agregada debe estar cifrada en reposo y tránsito.
- El sistema debe ser escalable para miles de órdenes por minuto.

---

8. IaC: Requerimientos CloudFormation

La plantilla debe definir:

- EventBridge bus o referencia a stream
- Lambda OrderProcessor
- Lambda Aggregator
- Lambda QueryAPI
- DynamoDB (raw + aggregates)
- Roles IAM (vinculados a plantilla de Ciberseguridad)
- API Gateway
- Cognito Authorizer
- VPC config (Lambda en subred privada)
- VPC Endpoints obligatorios
- Outputs para integraciones con otras células
- Parámetros para:
    - ARNs relevantes
    - Tablas
    - KMS keys
    - VPC ID
    - Subnets privadas

---

9. Pipeline CI/CD

El pipeline debe incluir:

Source – GitHub

- Código
- IaC
- buildspec
- documentación

Build – CodeBuild

- Validación del template
- Empaquetado de Lambdas
- Instalación dependencias

Deploy – CodePipeline + CloudFormation

- sandbox
- pre-producción
- producción

---

10. Seguridad, IAM y Cifrado

- Cognito obligatorio
- KMS obligatorio
- Lambdas en subred privada
- Acceso a AWS por endpoints
- Roles provistos por Ciberseguridad
- Validación estricta de payloads

- Auditoría completa en CloudWatch Logs
- Protección de consultas mediante autorización JWT

---

11. Datos de Prueba (GenAI)

La célula debe generar mínimo **50 órdenes sintéticas** donde cada orden incluya:

- orderId
- userId
- lista de gadgets (mínimo 1, máximo 10 por orden)
- precio total
- timestamp
- categoría del gadget

Los datos deben simular patrones de compra realistas:

- gadgets populares
- gadgets con ventas bajas
- compras nocturnas/matinales
- compras por categorías

---

12. Pruebas funcionales (curl)

Obtener resultados por gadget

```
curl -H "Authorization: Bearer <jwt>" \
  https://<api>/prod/aggregates/gadgets/GAD123
```

Obtener ventas por día

```
curl -H "Authorization: Bearer <jwt>" \
  https://<api>/prod/aggregates/daily?date=2025-02-01
```

Obtener ventas por mes

```
curl -H "Authorization: Bearer <jwt>" \
  https://<api>/prod/aggregates/monthly?month=2025-02
```

---

13. Diagrama de Arquitectura

El diagrama debe incluir:

- E-commerce (Célula 4) → evento de orden
- EventBridge o DynamoDB Streams
- Lambda OrderProcessor
- Lambda Aggregator
- DynamoDB (raw + aggregates)
- API Gateway + Cognito
- VPC endpoints
- Zona de inspección
- KMS

---

14. Entregables de la Célula

1. Repositorio GitHub
2. Plantillas CloudFormation
3. Código Python

4. Datos GenAI
5. Pipeline CI/CD
6. Pruebas curl
7. Diagrama draw.io
8. Backlog + historias de usuario
9. Estimación de costos

---

15. Criterios de Evaluación

- Correctitud del procesamiento y agregación
- Calidad del IaC
- Seguridad aplicada correctamente
- Funcionamiento end-to-end
- Escalabilidad del sistema
- Documentación y pruebas curl
- Diagrama claro

---

16. Control de Cambios

Versión	Fecha	Descripción	Autor
1.0	2025	Primera versión del documento de la Célula 8	Equipo de Arquitectura

---