
Documento Específico de la Célula 4

Sistema de E-Commerce de Gadgets – Acme Corp

Proyecto: Arquitecturas Serverless en AWS
Célula: 4
Integrantes: Isaí Candelario, Mariano Labrador
Versión: 1.0
Organización: Universidad La Salle – Ingeniería
Arquitectura Asignada: Sistema de e-commerce basado en microservicios serverless
Caso de uso: Plataforma principal de venta de gadgets Acme (core system)

Índice

14. Entregables de la Célula
15. Criterios de Evaluación
16. Control de Cambios

---

1. Introducción

La Célula 4 desarrolla uno de los sistemas **centrales (core)** de Acme:

la plataforma de comercio electrónico donde se venden los gadgets de la empresa.

Este módulo es el corazón del ecosistema tecnológico de Acme, y será utilizado por el resto de las células (incluyendo catálogo de imágenes, votación, triggers, etc.).

La arquitectura se implementará con un enfoque serverless, desglosada en microservicios independientes, escalables y seguros, todos alineados al **AWS Well-Architected Framework**.

---

2. Objetivo del Módulo

Crear una **plataforma de e-commerce funcional y segura**, capaz de manejar:

- Productos (gadgets)
- Usuarios
- Órdenes
- Carrito de compras
- Pagos (simulados)
- Integración con imágenes generadas por la Célula 3
- APIs autenticadas y autorizadas

- Persistencia serverless
- Procesamiento asíncrono de eventos
- Auditoría y trazabilidad

---

3. Alcance

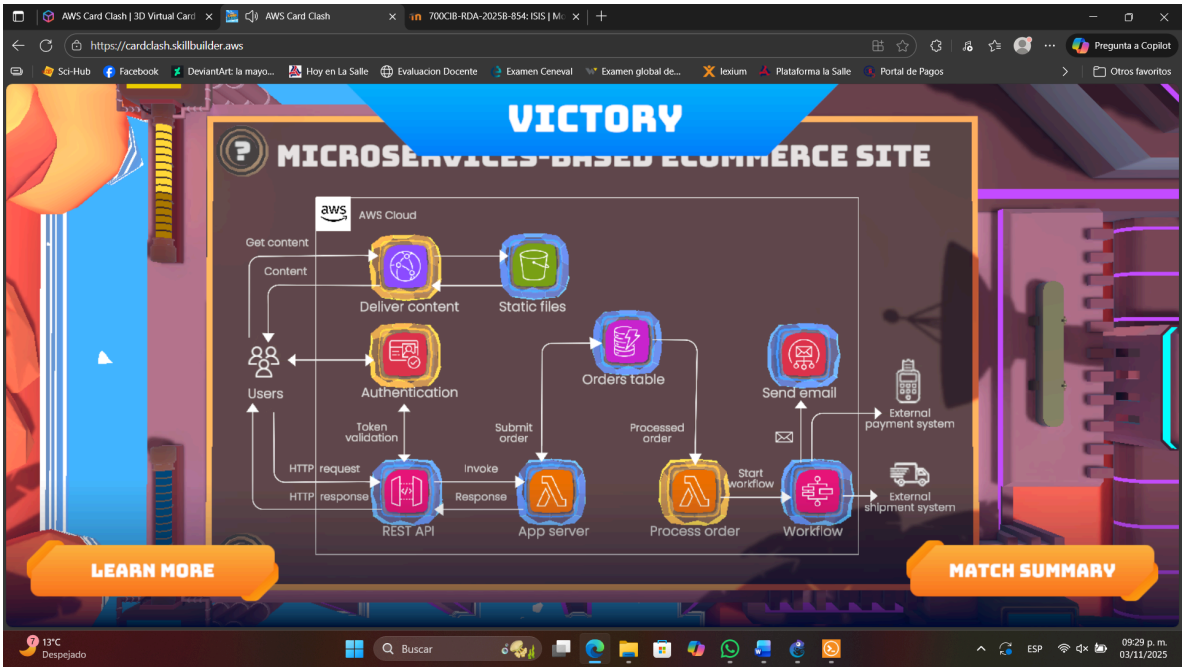
Incluye:

- Diseño completo del **dominio de e-commerce**
- Implementación en microservicios
- APIs REST seguras para interactuar con:
    - Productos
    - Carrito
    - Órdenes
    - Pagos
    - Usuarios
- Procesamiento de eventos internos
- Pipeline CI/CD
- Despliegues en sandbox, pre-producción y producción
- Documentación y pruebas curl

No incluye:

- Interfaz gráfica (frontend)
- Integración financiera real (pagos externos)
- Portal administrativo completo (solo APIs)

4. Arquitectura General



La arquitectura se basa en microservicios serverless, con los siguientes principios:

- Desacoplamiento basado en **API Gateway** y **EventBridge**
- Persistencia por dominio en **Amazon DynamoDB**
- Funciones de negocio en **AWS Lambda (Python)**
- Integración con Célula 3 para imágenes de productos
- Seguridad centralizada con **Cognito** y **KMS**
- Red privada con **VPC + endpoints + inspección**
- Mensajería interna con SQS/SNS/EventBridge

Flujo general del proceso de compra:

1. Usuario autenticado inicia sesión en Cognito
2. Consulta el catálogo de gadgets (productos)
3. Agrega artículos al carrito

4. Crea una orden
5. Se ejecuta el flujo de pago simulado
6. Se registra la orden final
7. Se publican eventos de seguimiento (ej. `OrderCreated`)

---

5. Dominios Funcionales del E-Commerce

El sistema debe dividirse en **microservicios**, idealmente uno por dominio:

5.1 Servicio de Usuarios

- Registro, autenticación mediante Cognito
- Sin manejo de contraseñas propio

5.2 Servicio de Productos

- CRUD de gadgets
- Integración con sistema de imágenes de la Célula 3
- Consulta pública autenticada

5.3 Servicio de Carrito (Cart Service)

- Agregar, modificar y eliminar artículos
- Cálculo preliminar del total

5.4 Servicio de Órdenes (Order Service)

- Creación de órdenes
- Validación del carrito

- Publicación de eventos

5.5 Servicio de Pagos (Simulado)

- Procesamiento ficticio
- Flujo aprobado/denegado
- Integración con EventBridge

5.6 Servicio de Notificaciones (Opcional)

- Confirmaciones por email/SNS
- Mensajería interna

---

6. Componentes AWS Utilizados

- **API Gateway**
    - Entrada a todos los servicios
- **Amazon Cognito**
    - Control de acceso, autenticación/autoría
- **AWS Lambda (Python)**
    - Implementación de cada microservicio
- **Amazon DynamoDB**
    - Tablas independientes por dominio:
        - Users
        - Products
        - Cart

        - Orders
        - Payments
- **Amazon S3**
    - Integración con Célula 3 para imágenes
- **Amazon EventBridge**
    - Orquestación y eventos internos
- **Amazon SQS/SNS** (opcional según diseño)
- **AWS KMS**
    - Cifrado total de datos
- **VPC + Endpoints + Zona de inspección**
    - Seguridad obligatoria
- **CodePipeline + CodeBuild**
    - CI/CD

---

7. Lineamientos Técnicos Específicos

- Microservicios totalmente desacoplados
- Ninguna función debe acceder directamente a la base de otra
- Cada API debe ser autenticada con Cognito
- Todas las tablas deben tener particionado eficiente
- Todas las operaciones deben ser idempotentes
- El sistema debe soportar:
    - Carga variable

    - Fallas parciales
    - Reejecución segura
- Todos los errores deben registrarse con CloudWatch Logs

---

8. IaC: Requerimientos CloudFormation

La plantilla debe incluir:

- API Gateway completo (múltiples rutas por dominio)
- Lambdas por microservicio
- DynamoDB por dominio con SSE-KMS
- Roles IAM (referenciados desde plantilla de Ciberseguridad)
- EventBridge (event bus, rules)
- VPC endpoints
- Variables cifradas
- Outputs para integraciones con otras células

---

9. Pipeline CI/CD

Debe contener:

1. **Source (GitHub)**
2. **Build (CodeBuild)**
    - Validación de IaC
    - Empaquetado de Lambdas

    - Instalación de dependencias
3. **Deploy (CloudFormation)**
    - Sandbox
    - Pre-producción
    - Producción

---

10. Seguridad, IAM y Cifrado

- Roles provistos por Ciberseguridad
- Token JWT obligatorio para todas las operaciones
- Tráfico por VPC endpoints
- Cifrado total en:
    - DynamoDB
    - S3
    - Logs
    - Parámetros
- Validación estricta de payloads
- No se permite acceso público sin Cognito

---

11. Datos de Prueba (GenAI)

La célula debe generar **mínimo 50 registros de gadgets** usando GenAI, incluyendo:

- productId

- nombre
- categoría
- características
- precio
- URL de imagen

---

12. Pruebas funcionales (curl)

Ejemplos obligatorios:

Login en Cognito

```
curl -X POST https://<cognito-domain>/oauth2/token \
    -d "grant_type=password" \
    -d "username=<user>" \
    -d "password=<pwd>" \
    -d "client_id=<client-id>"
```

Catálogo de productos

```
curl -H "Authorization: Bearer <jwt>" \
  https://<api-id>.execute-api.amazonaws.com/prod/products
```

Crear orden

```
curl -X POST \
  -H "Authorization: Bearer <jwt>" \
  -d '{"cartId": "CART123"}' \
  https://<api-id>.execute-api.amazonaws.com/prod/orders
```

---

13. Diagrama de Arquitectura

Debe incluir:

- Microservicios

- API Gateway
- Cognito
- DynamoDB
- Lambdas
- EventBridge
- Capa de imágenes (Célula 3)
- KMS
- VPC
- Endpoints
- Zona de inspección

---

14. Entregables de la Célula

1. Repositorio GitHub
2. CloudFormation completo
3. Lambdas Python
4. Datos GenAI
5. Pipeline CI/CD
6. Diagrama draw.io
7. Pruebas curl
8. Backlog completo
9. Estimación de costos

---

15. Criterios de Evaluación

- Diseño completo por dominios
- Desacoplamiento real de microservicios
- Calidad del IaC
- Cumplimiento estricto de seguridad
- Correcto uso de Cognito + API Gateway
- Funcionalidad demostrable
- Documentación y pruebas curl
- Complejidad bien manejada (sistema core)

---

16. Control de Cambios

Versión	Fecha	Descripción	Autor
1.0	2025	Primera versión del documento de la Célula 4	Equipo de Arquitectura

---