
Documento Específico de la Célula 7

Sistema de Votación “Gadget del Año” – Acme Inc.

Proyecto: Arquitecturas Serverless en AWS
Célula: 7
Integrantes: Isaí Candelario, Mariano Labrador
Versión: 1.0
Organización: Universidad La Salle – Facultad de Ingeniería
Arquitectura Asignada: Votación en tiempo real
Caso de uso: Plataforma anual de votación para elegir el “Gadget más novedoso” de Acme Inc.

Índice

---

1. Introducción

La Célula 7 es responsable del diseño e implementación del sistema serverless que permite a los empleados, desarrolladores, clientes beta o usuarios seleccionados votar por el **“Gadget más novedoso del año”** entre los productos de Acme Inc.

Este evento se realiza anualmente, y el sistema debe:

- soportar picos de carga intensos
- garantizar integridad del voto
- evitar voto duplicado
- dar resultados en tiempo real o casi real
- exponer APIs seguras
- mantener cumplimiento total con los lineamientos de seguridad de la organización

---

2. Objetivo del Módulo

Construir un servicio de votación serverless capaz de:

- Registrar usuarios y validar identidad mediante Cognito
- Registrar votos por gadget
- Prevenir votos duplicados

- Emitir métricas en tiempo real
- Entregar resultados agregados
- Integrarse con otros módulos (por ejemplo catálogo de gadgets del e-commerce)

---

3. Alcance

Incluye:

- API para votar
- API para obtener el estado de la votación
- Persistencia de votos
- Mecanismo anti-duplicado
- Stream en tiempo real de agregados
- Pipeline CI/CD
- Despliegue en sandbox, pre-producción y producción

No incluye:

- Interfaz gráfica del portal de votación (solo APIs)
- Integraciones externas fuera del ecosistema Acme

---

4. Descripción del Evento de Votación

Cada año Acme anuncia la lista oficial de gadgets candidatos, proveniente del catálogo administrado por la Célula 4 y la Célula 3 (imágenes de gadgets).

Los usuarios registrados pueden:

- revisar la lista de gadgets

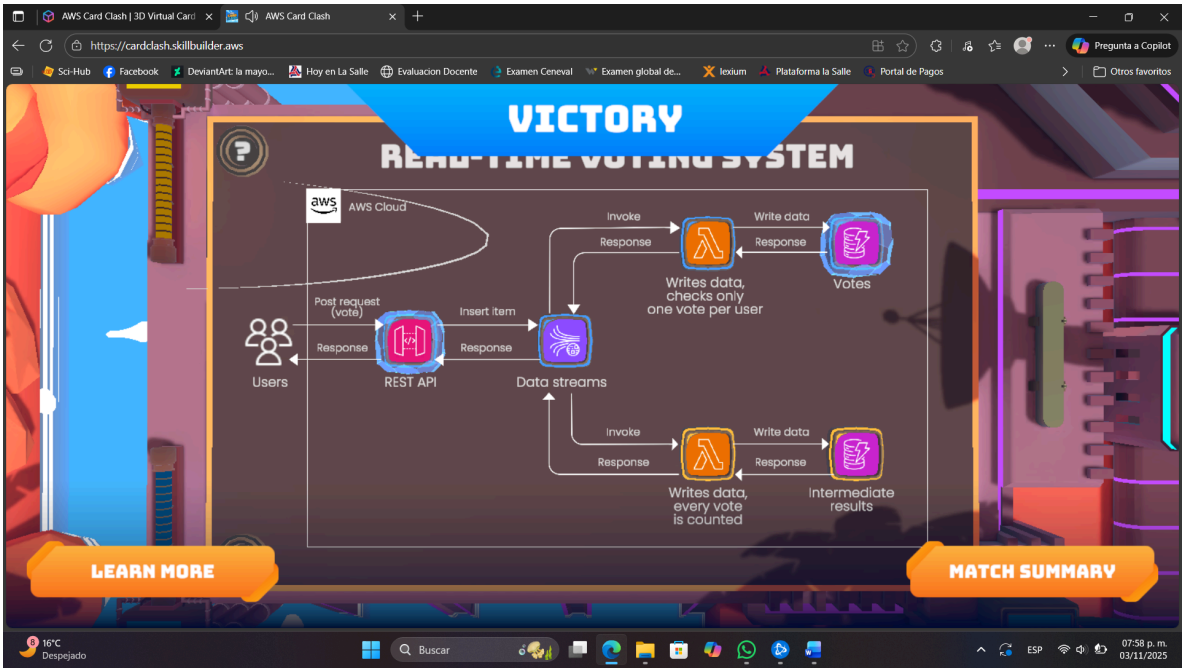
- emitir 1 voto por usuario
- ver los resultados en tiempo real

El sistema debe garantizar:

- **Un solo voto válido por usuario Cognito**
- **Actualización inmediata del total por gadget**
- **Imposibilidad de modificar votos pasados**

---

5. Arquitectura General



El flujo sugerido es:

1. El usuario inicia sesión mediante Cognito
2. Obtiene un JWT válido
3. Usa API Gateway para emitir su voto
4. Lambda valida:

    - identidad
    - que no haya votado antes
    - gadget válido
5. El voto se registra en DynamoDB
6. DynamoDB Streams genera un evento
7. Lambda o EventBridge procesa el stream:
    - recalcula contadores
    - actualiza tabla agregada de resultados
8. Usuarios consultan resultados a través de API Gateway
9. Todas las operaciones están registradas en CloudWatch Logs

---

6. Componentes AWS Utilizados

- **API Gateway**
    - Entrada para votar
    - Entrada para consultar resultados
- **Amazon Cognito**
    - Autenticación y autorización
- **AWS Lambda (Python)**
    - Emit Vote
    - Get Results
    - Stream Processor (DynamoDB)
- **Amazon DynamoDB**

    - Tabla `Votes` (votos individuales)
    - Tabla `VoteResults` (agregados por gadget)
- **DynamoDB Streams**
    - Para procesamiento en tiempo casi real
- **EventBridge**
    - (Opcional) Para eventos del proceso de votación
- **KMS**
    - Cifrado de tablas, logs, S3 (si aplica), parámetros
- **VPC Endpoints y zona de inspección**
    - Requerimientos obligatorios
- **CloudWatch**
    - Logs y métricas
- **CodePipeline / CodeBuild**
    - CI/CD

---

7. Lineamientos Técnicos Específicos

Reglas de negocio

- Cada usuario puede votar solo una vez
- Los votos no pueden modificarse
- La tabla de resultados debe estar siempre actualizada

Reglas técnicas

- Las APIs deben ser seguras, sin endpoints públicos sin autenticación

- El sistema debe manejar cargas elevadas en ventana corta
- Los registros deben ser idempotentes
- Debe garantizarse el orden lógico del proceso (voto → stream → agregación)

Esquema DynamoDB sugerido

Tabla Votes

- PK: `userId`
- SK: `voteId` (UUID)
- Atributos:
    - gadgetId
    - timestamp

Tabla VoteResults

- PK: `gadgetId`
- Atributos:
    - totalVotes
    - lastUpdated

---

8. IaC: Requerimientos CloudFormation

Debe incluir:

- API Gateway
- Cognito Authorizer
- Lambdas (3 funciones mínimo)

- DynamoDB (2 tablas)
- DynamoDB Streams
- VPC configuration (subred privada)
- VPC Endpoints (S3, DynamoDB, Logs, Secrets, etc.)
- Outputs para integraciones
- Parámetros configurables
- Uso de roles definidos por Ciberseguridad

---

9. Pipeline CI/CD

Estructura requerida:

1. Source (GitHub)

- Código Python
- IaC
- Documentación
- Datos de prueba

2. Build (CodeBuild)

- Validación del template
- Instalación de dependencias
- Empaquetado de Lambdas

3. Deploy (CloudFormation)

- Sandbox

- Pre-producción
- Producción

---

10. Seguridad, IAM y Cifrado

- Cognito obligatorio para acceso
- KMS para:
    - DynamoDB
    - Logs
    - Variables Lambda
- Security Groups restrictivos
- Zero-trust entre microservicios
- VPC Endpoints obligatorios
- Loggeo detallado para auditoría electoral interna

---

11. Datos de Prueba (GenAI)

La célula debe generar mínimo **50 votos y 10 gadgets sintéticos**, con:

- nombre del gadget
- ID
- imagen (opcional, vinculada con Célula 3)
- categorías
- descripción

Los votos deben representar escenarios como:

- picos masivos de votos
- usuario intentando votar dos veces
- gadgets con bajo/alto volumen de votos

---

12. Pruebas funcionales (curl)

Obtener token Cognito

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

Emitir voto

```
curl -X POST \
  -H "Authorization: Bearer <jwt>" \
  -d '{"gadgetId":"GAD42"}' \
  https://<api>/prod/vote
```

Consultar resultados

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

---

13. Diagrama de Arquitectura

Debe contener:

- API Gateway
- Cognito
- Lambda (emitVote)

- Lambda (getResults)
- Lambda (streamProcessor)
- DynamoDB (Votes / VoteResults)
- DynamoDB Streams
- VPC, endpoints y zona de inspección
- KMS

---

14. Entregables de la Célula

1. Repositorio GitHub
2. Plantillas CloudFormation
3. Código Python
4. Datos sintéticos
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 flujo de votación
- Prevenir voto duplicado

- Tiempo de actualización de resultados
- Calidad del IaC
- Uso adecuado de Cognito y KMS
- Correcto procesamiento de streams
- Documentación + pruebas curl
- Diagrama claro

---

16. Control de Cambios

Versión	Fecha	Descripción	Autor
1.0	2025	Primera versión del documento de la Célula 7	Equipo de Arquitectura

---