

Evaluación del módulo 6

Consigna del proyecto 

Evaluación del módulo

Proyecto: Monolitos Escalables

Situación inicial

Unidad solicitante: Área de Tecnología de una empresa de servicios digitales. La aplicación actual corre en un único servidor on-premise. Presenta caídas ante picos de tráfico y resulta costosa de mantener. La organización necesita migrarla a la nube manteniendo el enfoque monolítico, pero dotándola de **escalabilidad, alta disponibilidad y monitoreo**, utilizando únicamente recursos gratuitos de **AWS Academy** y herramientas visuales como **Cloudcraft**.

Nuestro objetivo

Diseñar e implementar una arquitectura monolítica escalable en un entorno cloud, aplicando principios de disponibilidad, balanceo de carga, contenerización y mensajería, con foco en buenas prácticas de infraestructura, monitoreo y diseño visual de soluciones.

Producto esperado

Implementación funcional de una **aplicación monolítica** desplegada en AWS con:

- **Escalabilidad automática** (Auto Scaling Group) y balanceo de carga (ELB).
- **Contenerización** mediante Docker (ECR + ECS opcional).
- **Servicios de mensajería** integrados (SNS/SQS) para procesos asíncronos.
- **Diagrama arquitectónico** en Cloudcraft con estimación de costos mensuales.

Requerimientos


- Desplegar la aplicación en **EC2** con **Auto Scaling** y **Elastic Load Balancer**.
- Crear una **base de datos** gestionada (RDS o DynamoDB).
- Empaquetar la aplicación en **Docker** y, de forma opcional, correrla en **ECS Fargate**.
- Configurar **SNS** y **SQS** para notificaciones y colas de trabajo.
- Representar toda la solución en **Cloudcraft**, incluyendo VPC, subredes y costos.
- Documentar cada paso, pruebas y refactorizaciones en un archivo Word.

Métricas Generales

- CRUD implementado: mín. 4 – máx. 5 funcionalidades.
- Tests unitarios: mín. 8 – máx. 16.
- Cobertura JaCoCo: mín. 80 %.
- Ciclos TDD (RED-GREEN-REFACTOR): mín. 12.
- Refactorizaciones: mín. 3 – máx. 5.
- Uso de Mockito: mín. 1 dependencia mockeada.

Paso a paso

Lección 1 – Fundamentos de Escalabilidad y TDD


 **Objetivo:** comprender la diferencia entre escalado vertical/horizontal y preparar el entorno de pruebas.

 **Tareas a desarrollar:**

- Leer el Manual L1.

- Instalar VS Code, JDK 17, JUnit 5 y JaCoCo.
- Crear carpetas src y test; iniciar repo Git.
- Ejecutar al menos un ciclo TDD (commit RED) y documentar la prueba RED.

Lección 2 – Implementación Monolítica en la Nube

 Objetivo: desplegar la aplicación base en AWS.

 Tareas a desarrollar:

- Leer el Manual L2.
- Crear instancia **EC2** y base **RDS/DynamoDB**.
- Conectar la aplicación al motor de datos y verificar funcionamiento.


Lección 3 – Escalabilidad y Alta Disponibilidad

 Objetivo: habilitar Auto Scaling y balanceo de carga.

 Tareas a desarrollar:

- Leer el Manual L3.
- Configurar **Auto Scaling Group** con métricas de CPU.
- Implementar un **Elastic Load Balancer** y probar la distribución de tráfico.

Lección 4 – Contenerización con Docker

 Objetivo: empaquetar la aplicación en contenedores.

 Tareas a desarrollar:

- Leer el Manual L4 (sección Docker).
- Crear **Dockerfile**, construir la imagen y subirla a **ECR**.
- (Opcional) Desplegar en **ECS Fargate** y comparar con EC2.

Lección 5 – Servicios de Mensajería

📌 Objetivo: integrar comunicación asíncrona.

📌 Tareas a desarrollar:

- Configurar un **Topic SNS** para notificaciones.
- Crear una **cola SQS** y simular el envío/recepción de mensajes.
- Documentar el flujo con capturas.

Lección 6 – Representación Cloud

📌 Objetivo: documentar visualmente la arquitectura y costos.

📌 Tareas a desarrollar:

- Crear el diagrama en **Cloudcraft** con todos los componentes.
- Estimar costos mensuales y añadirlos al Word.
- Exportar imagen o link del diagrama.

¿Qué vamos a validar? 🔍

- Uso correcto de servicios AWS (EC2, RDS, ELB, Auto Scaling, Docker, SNS/SQS).
- Cumplimiento de las **Métricas Generales**.
- Claridad y completitud de la documentación.
- Diagrama Cloudcraft coherente con estimación de costos.
- Aplicación de buenas prácticas de seguridad, monitoreo y refactor.

Referencias 🚧

Librerías y Frameworks:

- JUnit 5: <https://junit.org/junit5/docs/current/user-guide/>
- JaCoCo: <https://www.jacoco.org/jacoco/trunk/doc/>
- Mockito: <https://site.mockito.org/>

Documentación oficial:

- AWS Academy: <https://awsacademy.instructure.com>
- AWS Documentation: <https://docs.aws.amazon.com/>
- Docker Docs: <https://docs.docker.com/>
- Cloudcraft: <https://www.cloudcraft.co/>

Herramientas de estimación y cálculo:

- AWS Pricing Calculator: <https://calculator.aws.amazon.com/>

Recursos

- Videos: “Deploy Java App on AWS EC2”, “Auto Scaling Deep Dive”, “Docker Basics”.
- AWS Blog: mejores prácticas de ELB, Auto Scaling y mensajería.
- Medium / Dev.to: casos reales de monolitos escalables.
- Foros de AWS Academy y Stack Overflow para resolver dudas.

Entregables

1. **Documento Word** con:
 - Portada.
 - Desarrollo paso a paso con capturas.
 - Evidencia de pruebas, ciclos TDD y refactorizaciones.
 - Tabla de métricas cumplidas.

- Diagrama Cloudcraft (imagen/link) y costos.
- Conclusiones y lecciones aprendidas.

2. **Presentación breve** (opcional) para exponer resultados.

Portafolio

Añadí este proyecto como **“Monolito Escalable en AWS”** resaltando:

- Implementación de Auto Scaling y ELB.
- Contenerización Docker y (opcional) despliegue en ECS.
- Integración de SNS/SQS para mensajería.
- Diagrama profesional en Cloudcraft con costos.
- Pruebas, TDD y refactorizaciones que demuestran buenas prácticas.

¡Éxitos!

Nos vemos más adelante

