

---

## PROYECTO 1

---

**202346773 –** Marco Antonio Martínez Gutiérrez

**202342112 –** Douglas Estuardo Cetino Ciramagua

### Resumen

En el proyecto CloudSync, se profundizó en la gestión de recursos en la nube, un área crítica en la infraestructura tecnológica actual. Para ello, se desarrolló un simulador integral capaz de administrar centros de datos, máquinas virtuales y solicitudes de clientes. La metodología empleada priorizó el uso de programación orientada a objetos (POO) y la creación de estructuras de datos propias, como listas enlazadas y colas de prioridad. Al evitar bibliotecas nativas, se logró replicar con precisión la lógica de un orquestador real, exigiendo un diseño riguroso del código. El sistema integra el procesamiento de archivos XML para la gestión de datos y utiliza Graphviz para la generación de reportes visuales del estado del sistema. Esta implementación permitió demostrar la relevancia de un diseño de software sólido y la correcta aplicación de algoritmos en la construcción de sistemas complejos a futuro.

### Palabras clave

CloudSync, gestión de recursos, nube, POO, XML

### Abstract

In the CloudSync project, we delved into cloud resource management, a critical area in today's technological infrastructure. To achieve this, an integrated simulator was developed, capable of managing data centers, virtual machines, and client requests. The methodology prioritized the use of Object-Oriented Programming (OOP) and the creation of custom data structures, such as linked lists and priority queues. By avoiding native libraries, we accurately replicated the logic of a real orchestrator, requiring a rigorous code design. The system integrates XML file processing for data management and utilizes Graphviz to generate visual reports of the system's status. This implementation demonstrated the importance of solid software design and the correct application of algorithms for building complex systems in the future.

### Keywords

CloudSync, resource management, cloud, OOP, XML

## Introducción

En la actualidad, la computación en la nube es el motor tecnológico global, exigiendo sistemas capaces de gestionar procesos masivos sin interrupciones. Bajo este contexto surge CloudSync, un proyecto diseñado para profundizar en la gestión de recursos mediante la simulación de centros de datos, máquinas virtuales y solicitudes de usuarios.

El objetivo central es comprender la organización interna y la flexibilidad de la infraestructura virtual. El sistema procesa configuraciones mediante archivos XML, permitiendo crear recursos, realizar migraciones y atender peticiones según su prioridad. Para lograrlo, se empleó programación orientada a objetos (POO) y estructuras de datos personalizadas, replicando la lógica de un orquestador real. Este trabajo detalla el diseño e implementación de una solución técnica que aplica conceptos fundamentales de ingeniería de software para resolver problemas complejos de la infraestructura digital actual.

## Descripción del proyecto

CloudSync es un sistema que empieza a funcionar gracias a un archivo XML que le da toda la Estructura inicial. En ese archivo, definimos la infraestructura: los edificios físicos (centros de datos), las computadoras virtuales (máquinas virtuales), las aplicaciones (contenedores) y lo que los clientes piden (solicitudes).

El sistema funciona con estos componentes principales:

**Centros de datos:** El lugar físico o lógico donde se alojan todos los recursos como CPU, RAM y almacenamiento.

**Máquinas virtuales (VMs):** Unidades lógicas que consumen esos recursos y pueden tener varias aplicaciones dentro.

**Contenedores:** Aplicaciones aisladas que corren dentro de las VMs, usando solo una parte de sus recursos.

**Solicitudes:** Tareas de clientes (como hacer un *deploy* o un *backup*) que el sistema debe procesar.

Una vez que cargamos el XML, el programa organiza internamente toda la información con las estructuras de datos que implementamos. Luego, nos permite interactuar con un menú en la consola para hacer consultas, migraciones y generar reportes visuales.

## Análisis y postura sobre el proyecto

El enfoque que le dimos a CloudSync tiene puntos muy sólidos. Al usar programación orientada a objetos logramos que el sistema sea súper modular en pocas palabras es fácil entender qué hace cada parte porque cada componente actúa como una entidad real de la nube. También decidimos armar nuestras propias estructuras de datos, como las listas y colas de prioridad, lo que nos dio un control total sobre cómo fluye la información. Es cierto que al no tener el conocimiento más avanzado y no poder usar ni comprender las librerías optimizadas del lenguaje, quizás el rendimiento no sea el más capacitado para algo a gran escala, pero el valor real de CloudSync está en el aprendizaje. Priorizamos entender cómo funciona "el

"motor" por dentro antes que la velocidad pura y eso ayuda muchísimo a entender cómo se gestionan los recursos en la vida real.

### **La Lógica Del Código**

Para que el código fuera manejable y claro, usamos el enfoque de Programación Orientada a Objetos (POO). Creamos una clase para cada cosa (Centro de Datos, Máquina Virtual, etc.), lo que hizo más fácil organizar la información y ver cómo se relacionaban entre sí, algo que plasmamos en nuestro diagrama de clases.

Un punto clave es que no usamos las listas nativas del lenguaje. Tuvimos que crear nuestras propias listas anidadas, nodos, y colas de prioridad para simular cómo se administra la información en un sistema real de este tipo.

### **Tenemos un par de algoritmos importantes:**

**Gestión de Solicitudes:** Las peticiones de los clientes van a una cola de prioridad. El sistema siempre saca primero la tarea más urgente y revisa si hay recursos disponibles para hacerla.

**Migración de VMs:** Si necesitamos mover una máquina de un centro de datos a otro, el algoritmo se asegura de que el nuevo centro tenga suficiente espacio y actualiza todo correctamente.

### **Interacción y Funcionalidades**

El sistema no es solo un programa que corre solo, está pensado para que el usuario pueda interactuar con él a través de un menú en la consola. Esto nos da control total sobre la simulación:

**Cargar Archivo:** La primera opción, es cargar el XML inicial que contiene toda la configuración y las instrucciones a seguir.

**Gestión de Centros de Datos:** Aquí podemos ver detalles específicos. Podemos listar todos los centros, buscar uno por su ID para ver cuántos recursos totales, disponibles y usados tiene, o incluso pedirle al sistema que nos diga cuál es el centro con más recursos libres.

**Gestión de Máquinas Virtuales:** Nos permite buscar una VM por su ID para ver su sistema operativo y recursos, listar todas las VMs de un centro específico, o realizar la migración entre centros que mencionamos antes, validando siempre la disponibilidad.

**Gestión de Contenedores:** Podemos desplegar nuevos contenedores dentro de una VM existente, listar los que ya están corriendo, cambiarles el estado (pausar, reiniciar, etc.) o eliminarlos para liberar recursos.

**Gestión de Solicitudes:** Aquí es donde agregamos nuevas tareas a la cola de prioridad, procesamos la solicitud más urgente en ese momento, o procesamos un número específico de solicitudes N en orden de prioridad.

### **Estructura de Archivos: XML de Entrada y Salida**

El corazón de la comunicación del sistema son los archivos XML. Tienen un formato específico que debemos respetar, tanto para leer la entrada como para generar la salida.

## **Archivo de Entrada**

El archivo de entrada (cloudSync.xml) es fundamental. No solo contiene la lista de recursos iniciales (centros de datos, VMs, contenedores), sino que también incluye una sección de <instrucciones>. Estas instrucciones le dicen al programa qué acciones específicas debe tomar *después* de cargar todo, como crear VM, migrar VM o procesar Solicitudes.

## **Archivo de Salida**

Al final de la ejecución, el sistema genera un archivo de salida que resume todo lo que pasó. Este archivo incluye un timestamp del momento en que se generó y estadísticas detalladas del estado final de cada centro de datos, como el porcentaje de utilización de CPU y RAM, y la cantidad total de VMs y contenedores activos.

## **Reportes**

Para que fuera fácil ver cómo está el sistema, generamos reportes gráficos usando una herramienta llamada Graphviz. Esto nos permite ver mapas visuales de los centros de datos, las máquinas, los contenedores y hasta el orden de las solicitudes pendientes, todo a nuestra creatividad, pero respetando la estructura

## **Conclusiones.**

Desarrollar CloudSync fue la oportunidad perfecta para poner en práctica todo lo que vimos en teoría sobre la nube, POO y estructuras de datos. Con este proyecto nos dimos cuenta de que en sistemas tan complejos si el código no está bien organizado y los recursos no se gestionan con cuidado todo se cae. Como grupo vemos de suma importancia analizar primero que hay debajo de cada programa para maximizar el aprendizaje y eso lo pudimos observar mediante la creación del proyecto. Este proyecto nos deja pensando en cómo llevar estos modelos a algo más grande y nos marca el camino para empezar a explorar herramientas más potentes en el futuro.

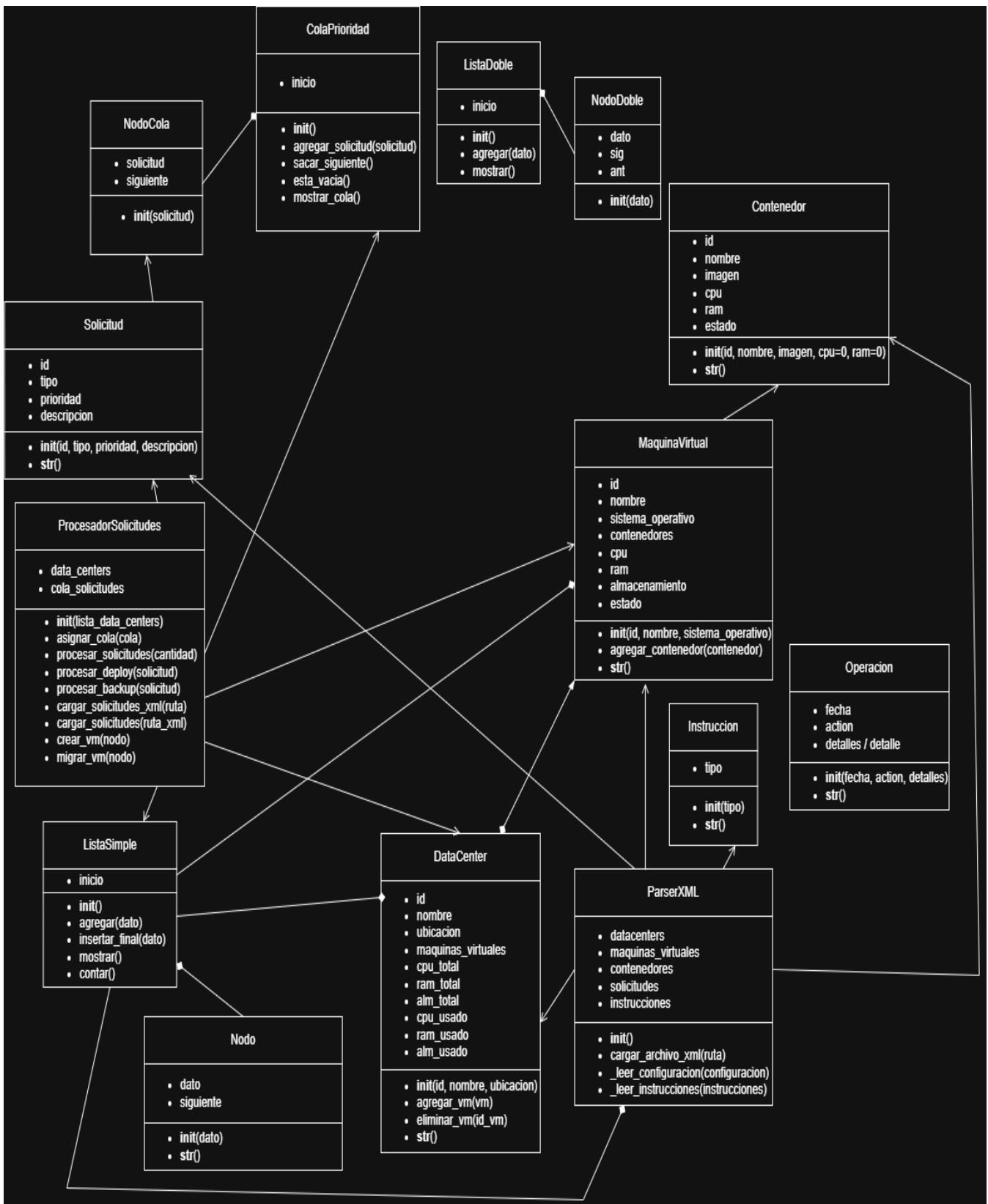


Figura 1. Diagramas de Clases.

Fuente: elaboración propia

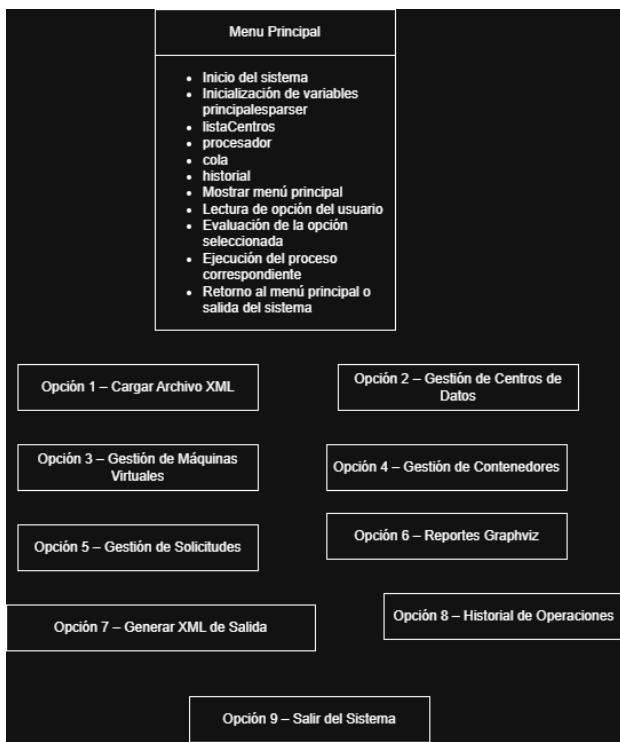


Figura 2 Diagramas de Actividades 0.

Fuente: elaboración propia

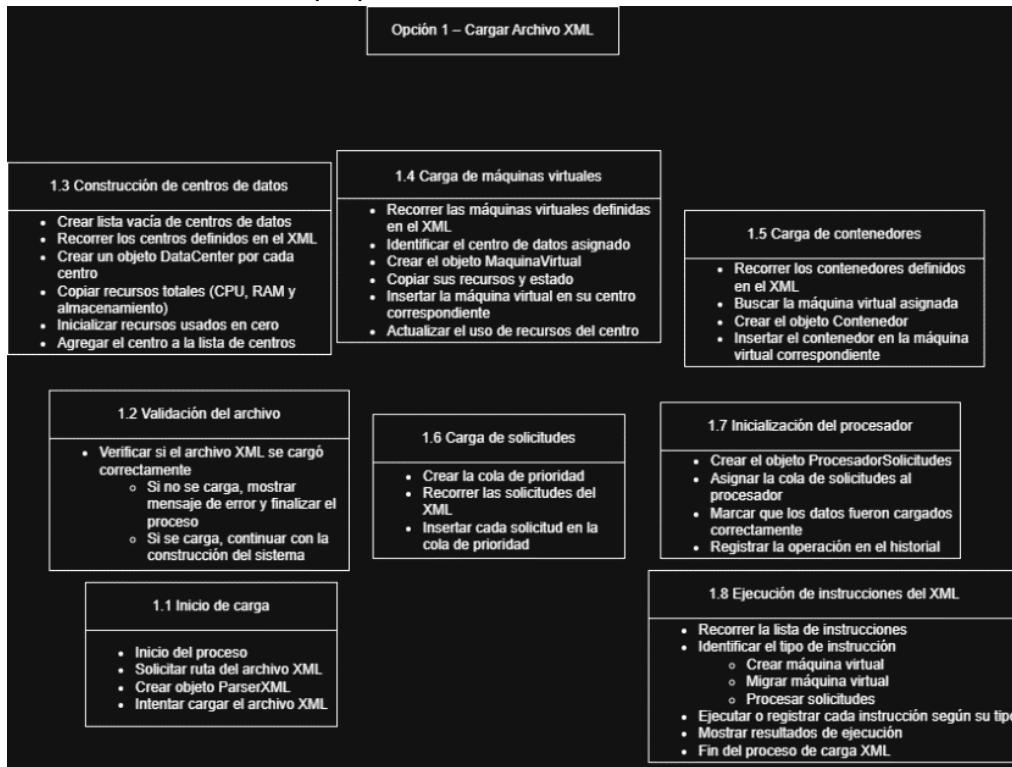


Figura 3. Diagramas de Actividades 1.

Fuente: elaboración propia

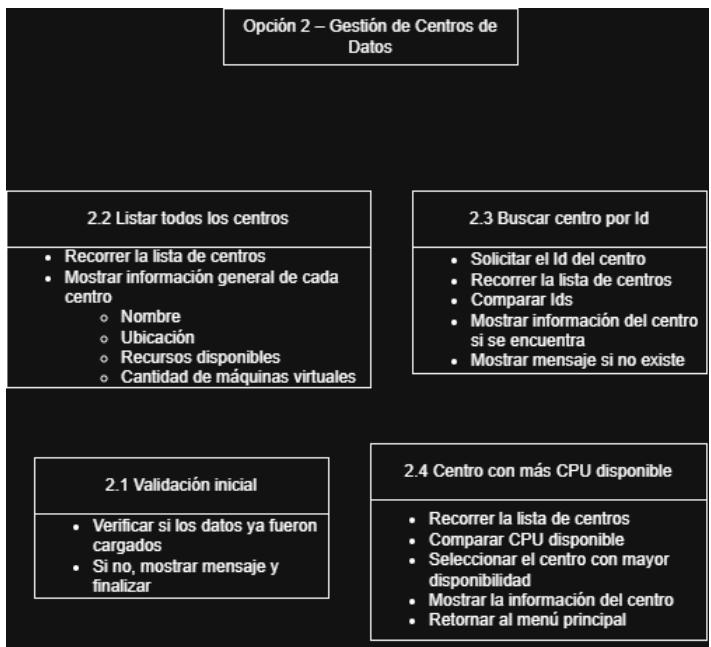


Figura 4. Diagramas de Actividades 2.

Fuente: elaboración propia

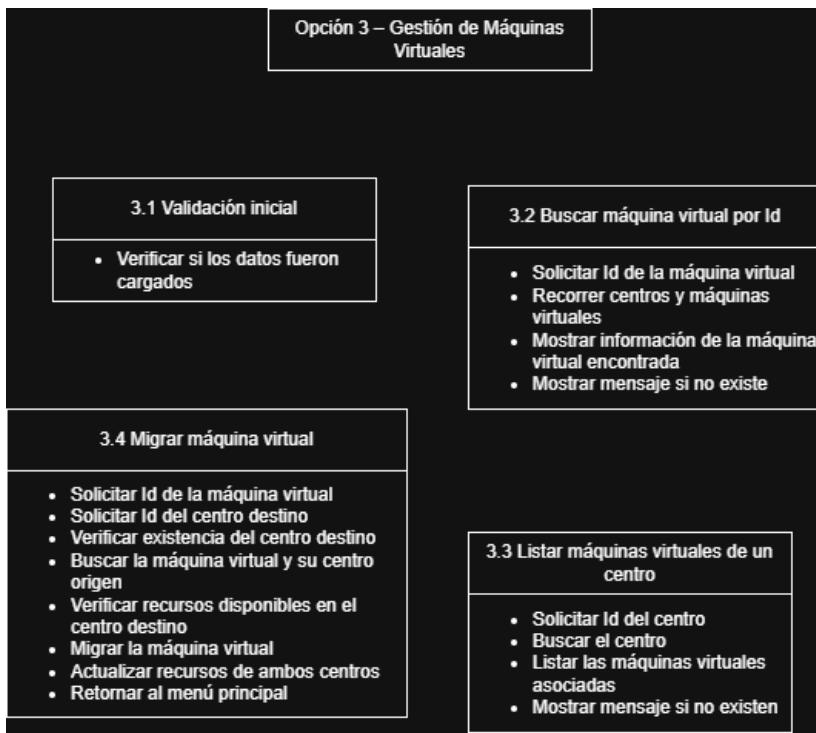


Figura 5. Diagramas de Actividades 3.

Fuente: elaboración propia

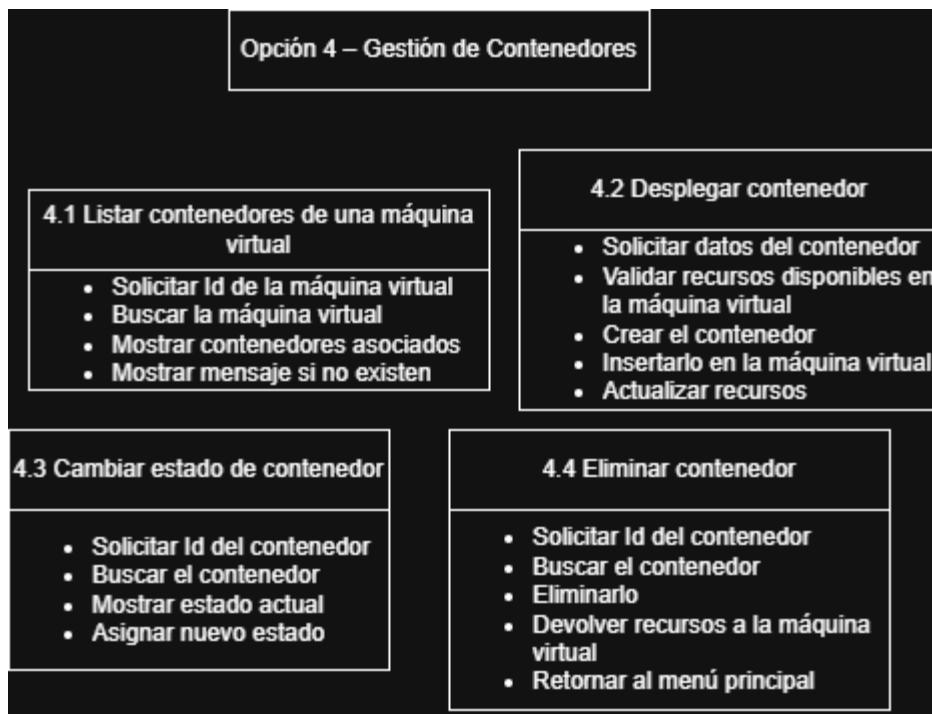


Figura 6. Diagramas de Actividades 4.

Fuente: elaboración propia

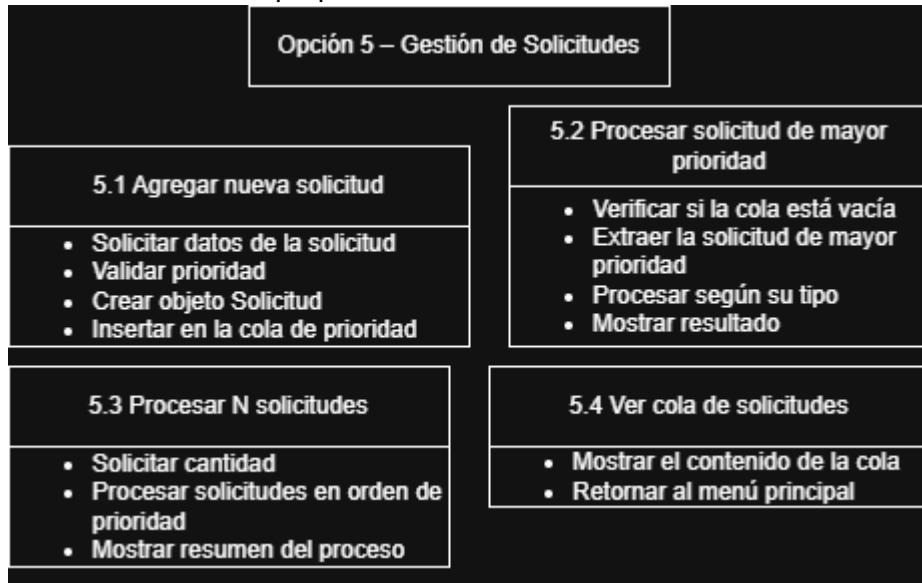


Figura 7. Diagramas de Actividades 5.

Fuente: elaboración propia

## Opción 6 – Reportes Graphviz

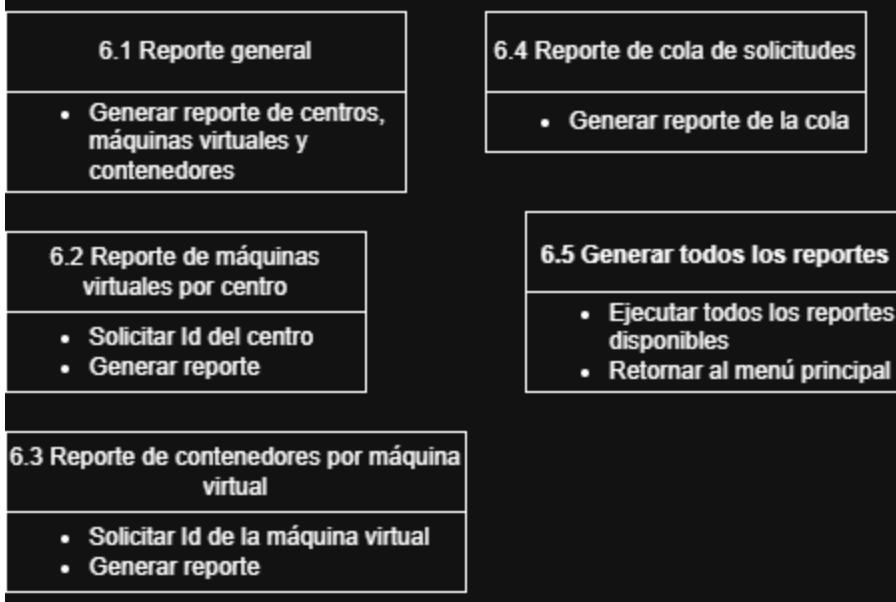


Figura 8. Diagramas de Actividades 6.

Fuente: elaboración propia

## Opción 7 – Generar XML de Salida

- Verificar que los datos estén cargados
- Generar archivo XML con el estado final del sistema
- Calcular estadísticas de recursos
- Guardar archivo XML
- Registrar operación

Figura 9. Diagramas de Actividades 7.

Fuente: elaboración propia

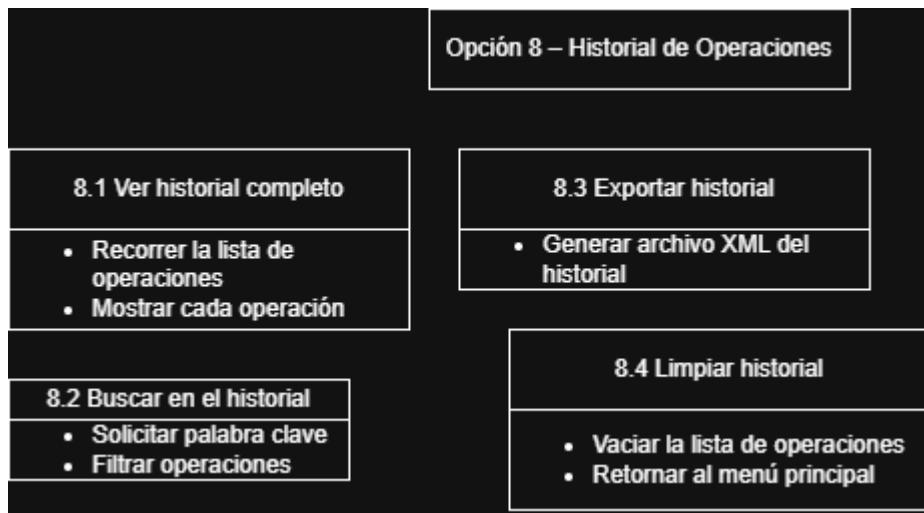


Figura 10. Diagramas de Actividades 8.

Fuente: elaboración propia

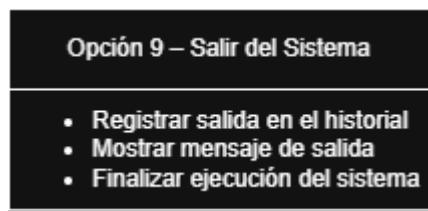


Figura 11. Diagramas de Actividades 9.

Fuente: elaboración propia