Guía de taller: Desarrollo de un Sistema de Gestión de Información con Múltiples Estructuras de Datos

---

Introducción

El objetivo de este taller es diseñar e implementar un sistema de *software* que sirva como plataforma para la gestión de datos. A diferencia de un proyecto tradicional, el enfoque principal no es solo la funcionalidad final, sino la **aplicación estratégica de diversas estructuras de datos**. Este proyecto les permitirá entender por qué y cuándo elegir una estructura sobre otra, conectando así la teoría con la práctica de la ingeniería de *software*.

Fases del taller

Fase 1: Diseño y arquitectura del sistema

En esta fase, su equipo definirá la funcionalidad del sistema y diseñará su arquitectura, decidiendo qué estructura de datos es la más adecuada para cada componente.

1. Definición del problema (Escenario de ejemplo: Sistema de Gestión de un Hospital)

El proyecto será un sistema para gestionar información en un hospital. Este sistema debe ser capaz de:

- Gestionar el registro de pacientes y médicos.
- Programar citas médicas y consultas de emergencia.
- Mantener un historial clínico y de tratamientos para cada paciente.
- Generar reportes y estadísticas.
- Ofrecer un sistema de recomendaciones.

2. Mapeo de funcionalidades a estructuras de datos:

Para cada funcionalidad, decidan qué estructura de datos utilizar y justifiquen su elección.

Funcionalidad	Estructura de Datos	Justificación
Registro de Pacientes/Médicos	**Tablas Hash (Hash Maps)**	Permite un acceso, inserción y eliminación de registros casi instantáneo utilizando el ID del paciente o médico como clave.
Búsqueda de Pacientes por Nombre	**Árbol de Prefijos (Trie)**	Ideal para la búsqueda por texto y la funcionalidad de autocompletado en un campo de búsqueda, ya que es más

		eficiente que una tabla *hash* para búsquedas basadas en prefijos.
Gestión de Citas (Próximas)	**Cola de Prioridad (Priority Queue)**	Las citas pueden ser ordenadas según la prioridad (por ejemplo, emergencias) para ser atendidas en el orden correcto.
Historial de Consultas de un Paciente	**Lista Enlazada Doblemente**	Permite una navegación eficiente hacia adelante y atrás a través del historial de consultas, facilitando la visualización cronológica.
Historial de Acciones del Usuario (Deshacer)	**Pila (Stack)**	La función "deshacer" sigue el principio LIFO (*Last-In, First-Out*), perfecto para revertir la última acción realizada por un usuario o administrador.
Reportes de Tendencias (Recomendaciones)	**Grafo (Graph)**	Permite modelar las relaciones entre pacientes, enfermedades y tratamientos. Esto sería útil para sugerir tratamientos o diagnósticos basados en patrones encontrados en el grafo.
Lista de Pacientes por Orden Alfabético	**Árbol Binario de Búsqueda (BST)**	Un árbol binario es perfecto para mantener una lista ordenada de forma eficiente, permitiendo búsquedas rápidas y la visualización de datos ordenados.

Exportar a Hojas de cálculo

Fase 2: Implementación modular

1. Diseño de clases (Programación Orientada a Objetos - POO):

- Definan clases como `Paciente`, `Medico`, `Cita`, `Consulta`, etc. Cada clase debe tener sus propios atributos y métodos.

2. Implementación de las estructuras de datos:

- **No usen las librerías preconstruidas** para las estructuras de datos (como `java.util.HashMap` o `collections.deque` de Python). En este taller, deben implementar desde cero al menos 3 estructuras de datos clave (por ejemplo: la tabla *hash*, el árbol binario de búsqueda y la pila). El resto se pueden usar de librerías para agilizar el proyecto.

3. Desarrollo de los módulos:

- Dividan el trabajo en módulos. Por ejemplo, un módulo para la gestión de pacientes (utilizando la tabla *hash*), otro para el historial clínico (lista enlazada), y otro para la programación de citas (cola de prioridad).

Fase 3: Integración y pruebas

1. Integración de los módulos:

- Combinen todos los módulos para construir el sistema completo. Asegúrense de que las interacciones entre las diferentes estructuras de datos funcionen correctamente.

2. Pruebas unitarias y de integración:

- Desarrollen pruebas para verificar que cada estructura de datos y cada funcionalidad se comporten como se espera.

Entregable y Evaluación

- **Código fuente:** El proyecto debe ser un repositorio de código ordenado y bien comentado.
- **Documento de diseño:** Un documento PDF que contenga:
    - Una breve descripción del sistema.
    - Un diagrama de clases (UML) del diseño del sistema.
    - La justificación detallada para cada elección de estructura de datos, tal como se hizo en la Fase 1.
- **Presentación:** Una demostración funcional del sistema, explicando el flujo de datos y cómo cada estructura contribuye al desempeño del sistema.

NOTA: Deben soportar la actividad con evidencias del uso de los recursos bibliográficos de nuestra Biblioteca.