INFORME FINAL - DESAFÍO II – UdeAStay

ANYELA MARTÍNEZ ORTEGA

AUGUSTO ENRIQUE SALAZAR JIMENEZ

INFORMÁTICA II

UNIVERSIDAD DE ANTIOQUIA FACULTAD DE INGENIERÍA

MAYO DE 2025

1. Análisis del problema y consideraciones para la alternativa de solución propuesta

Desde el primer momento, entendí que el objetivo del Desafío II era aplicar los conocimientos adquiridos sobre programación orientada a objetos en C++, con un enfoque realista, simulando un sistema similar a Airbnb. El sistema debía ser capaz de gestionar reservaciones de alojamientos, permitiendo operaciones tanto para anfitriones como para huéspedes.

Mi enfoque fue diseñar una estructura de clases sencilla pero funcional, sin utilizar STL ni estructuras complejas, ajustándome a los lineamientos de programación básica. Identifiqué los actores principales (‘huesped’, ‘anfitrion’, ‘alojamiento’, ‘reservacion’ y ‘fecha’) y sus responsabilidades dentro del sistema. Con base en esto, desarrollé una solución modular, donde cada clase tiene tareas específicas y comunicación limitada pero efectiva con las demás.

2. Diagrama de clases de la solución planteada

El sistema se compone de las siguientes clases:

- fecha: gestiona fechas y operaciones básicas como comparación, suma de días y formatos bonitos para comprobantes.

- alojamiento: representa un alojamiento, puede cargar información desde archivo, verificar disponibilidad, filtrar resultados por municipio, precio y puntuación, y mostrar la información por código.

-reservacion: estructura básica para leer y mostrar datos de reservas, así como para actualizar los históricos.

-anfitrion: permite login, consultar y anular reservas; también permite ver todos los alojamientos que administra.

-huesped: permite login, reservar, anular, y generar comprobantes de reserva con los datos del sistema.

Cada clase interactúa mediante lectura y escritura de archivos de texto plano (.txt). El sistema se basa completamente en archivos planos.

3. Descripción de alto nivel de las tareas no triviales

mostrarAlojamientosDisponibles: busca en el archivo ‘alojamientos.txt’ aquellos registros que coincidan con el municipio ingresado y luego consulta en ‘reservas.txt’ si están disponibles para la fecha solicitada.

mostrarAlojamientosFiltrados: filtra los alojamientos por precio máximo y puntuación mínima del anfitrión, leyendo datos de ‘usuariosA.txt’.

consultarReserva y updateHistorial: recorren las reservas para buscar las correspondientes al anfitrión o moverlas al archivo histórico según fecha de corte.

reservarAlojamiento y generarComprobante: toman datos de entrada, validan disponibilidad y solapamiento de reservas, crean un registro en ‘reservas.txt’ y generan un archivo comprobante.

4. Algoritmos implementados intra-documentados

Cada método incluye comentarios aclarando:

Inicio y fin del código medido con ‘contador_iteraciones’.

Validaciones de punteros con ‘strtok’ para evitar errores por ‘nullptr’.

Conversión segura de ‘char*’ a ‘string’ para evitar errores en las comparaciones y manipulaciones.

Lógica condicional explicada paso a paso para facilitar la comprensión y el mantenimiento del código.

Al finalizar cada funcionalidad, el sistema imprime en pantalla la cantidad de iteraciones realizadas y la memoria utilizada, cumpliendo con los requerimientos de medición de recursos.

5. Problemas de desarrollo que afronté

Debo ser honesta: me enredé mucho con el enfoque de programación orientada a objetos. No entendí bien cómo funcionan las clases ni cómo se usan los constructores y destructores en la práctica. A medida que avanzaba, me costó bastante estructurar el sistema usando POO; no tanto por la dificultad de escribir las funciones, sino porque no me quedaba claro dónde ubicarlas, por qué crear ciertas clases y por qué algunas funciones debían ir en una clase y no en otra. Por momentos pensé en abandonar el desafío y prepararme mejor para el proyecto final, porque organizar el código en

clases me resultó mucho más difícil que programar con funciones sueltas. Esta experiencia me hizo darme cuenta de los vacíos que aún tengo en el manejo de la POO y la necesidad de practicar más cómo diseñar sistemas orientados a objetos.

6. Evolución de la solución y consideraciones para implementación

Durante el desarrollo, tuve que buscar varios ejemplos de otros sistemas hechos con POO para poder guiarme, ya que al principio me costaba mucho avanzar. A pesar de todas las complicaciones y dudas, logré estructurar el sistema y que funcionara de acuerdo con lo pedido en el desafío. Soy consciente de que no es el sistema más eficiente ni el más avanzado, e incluso reconozco que puede que falte alguna funcionalidad para que esté 100% completo. Sin embargo, creo que hice un buen esfuerzo y que el sistema cumple con lo esencial solicitado en el enunciado. El proceso me aportó una valiosa experiencia para futuros proyectos y me ayudó a comprender mejor la importancia de la estructura y la lógica en la programación orientada a objetos.