Informe Preliminar Desafío II

Marx Eusdav Lopez Montero, Daniel Antonio Londoño Godoy

Facultad De Ingeniería, Universidad De Antioquia

Informática II: 2598521

17 de mayo de 2025

El desarrollo de sistema UdeAStay implica la gestión de alojamientos, reservaciones, anfitriones y huéspedes mediante Programación Orientada a Objetos. Para garantizar la eficiencia, se adoptó una estructura modular, con almacenamiento en archivos de texto y administración de memoria dinámica.

El problema central se enfoca en modelar interacciones reales de un servicio de estadías, asegurando que la información sea consistente, rápida de consultar y actualizable. Un desafío clave es la gestión de fechas, evitando solapamientos en reservaciones y optimizando el acceso a datos.

Se definieron cinco clases fundamentales para estructurar el sistema. Estas son: Huésped, Anfitrión, Alojamiento, Fecha y Reserva. Aunque los huéspedes y anfitriones comparten atributos como identificación y antigüedad en la plataforma, sus roles son completamente distintos: los anfitriones administran alojamientos, mientras que los huéspedes gestionan reservaciones, lo que justifica su separación en clases independientes. La clase Alojamiento encapsula características específicas de cada propiedad, incluyendo ubicación, precio, amenidades y fechas reservadas, evitando redundancia en otras entidades. Por otro lado, Reserva se encarga de la gestión de hospedajes, validando disponibilidad y registrando pagos, lo que exige su propia estructura. Finalmente, Fecha facilita operaciones temporales dentro del sistema, permitiendo cálculos precisos de duración y disponibilidad sin afectar directamente la lógica de negocio de Reserva o Alojamiento. Esta arquitectura permite la carga eficiente de datos desde archivos de texto (.txt).

Se tomó la decisión de utilizar matrices dinámicas en la gestión de los archivos y las reservas, ya que responde a la necesidad de gestionar grandes volúmenes de datos de manera eficiente, evitando la duplicación innecesaria de información y optimizando el uso de memoria.

Se decidieron crear cinco archivos .txt para trabajar. Se estableció un formato estructurado para cada archivo, asegurando coherencia y facilidad de lectura/escritura, con formato:

- Alojamiento → (0)códigoIdentificador, nombre, documentoAnfitrión, departamento, municipio, tipo [1 (casa) o 0 (apartamento)], dirección, precioPorNoche, amenidades [array de 1s y 0s de tamaño 5], reservacion0 - reservacion1 - … reservacionn
- Anfitrion → documento, antigüedad (meses), puntuación (0 a 5.0), alojamiento1 - … - alojamienton
- Huésped → documento, antigüedad (meses), puntuación (0 a 5.0), reservacion0 - reservacion1 - … - reservacionn
- Histórico y Reservas→ (1)códigoDeReservación, (0)códigoAlojamiento. fechaDeEntrada, Duración (en cantidad de noches), fechaDeSalida, documentoDelHuésped, métodoDePago [1 (tarjeta) o 0 (PSE)], fechaDePago, monto, anotaciones

Para mejorar la velocidad de búsqueda, se utilizará búsqueda binaria, lo que permitirá localizar rápidamente una fecha dentro de un conjunto ordenado sin necesidad de recorrer toda la lista de manera secuencial. Además, cada vez que se registre una nueva reserva, las fechas dentro de la estructura serán ordenadas dinámicamente para mantener la coherencia y garantizar que futuras búsquedas sean eficientes. En lugar de realizar una ordenación completa tras cada modificación, se empleará inserción ordenada, permitiendo actualizar la estructura sin afectar significativamente el tiempo de ejecución.

Diagrama de clases