Informe proceso de análisis y diseño de la solución

Desafío II

Samuel Zuleta Zapata
Juan Fernando Henao Ledesma

Informática II

Universidad de Antioquia

2025

Tras haber analizado los requerimientos para la plataforma UdeAStay y teniendo en cuenta el paradigma de POO, consideramos que el paso inicial para nuestra solución es realizar un proceso de abstracción de la realidad y determinar cuántas clases van a ser necesarias. Como solución inicial se nos ocurren 5 clases, clase anfitrión, clase alojamiento, clase reservación, clase huésped y finalmente clase fecha. Como dice en el enunciado un anfitrión puede tener numerosos alojamientos, un alojamiento puede tener numerosas reservaciones y un huésped puede tener numerosas reservaciones. Fecha sería una clase que nos brindaría utilidad a la hora de verificar disponibilidades, fechas válidas, cálculos de estadías, entre otras funciones.

Los atributos de las clases están prácticamente explícitos en el documento, lo verdaderamente importante es analizar cuál va a ser la estructura de datos que vamos a usar y cómo se van a relacionar entre ellos, la primera idea es que anfitrión tenga un arreglo de objetos de la clase alojamientos, alojamientos tenga un arreglo de la clase reservaciones y huéspedes tenga un punteros a objetos tipo reservación, la clase fecha estaría incluida como atributo en la clase reservación. Todo esto al ser un análisis inicial está sujeto a cambios claramente, sin embargo en este momento nos parece una ruta viable.

Otro punto de suma importancia es el almacenamiento permanente, se define que se van a usar 5 archivos txt, uno para los huéspedes, uno para los anfitriones, uno para los alojamientos y 2 para las reservaciones, siendo uno el histórico y otro para las reservas vigentes, el formato de estos se sigue evaluando y estará evidenciado en el informe final para la legibilidad del código. La parte principal de la funcionalidad de carga es el orden en el que se va a realizar y cómo se van a ir instanciando los objetos en cuestión, esta funcionalidad debe ir abriendo archivo por archivo, crear el objeto y agregarlo a la estructura de datos correspondiente.

Otra funcionalidad clave es la medición de los recursos, las iteraciones por funcionalidad se pueden resolver simplemente a través de contadores estáticos en los bucles más internos de los métodos que se usen, la medición de la memoria es algo más complejo, ya que el reto va a estar en los datos estructurados de nuestros objetos, hay algunas clases que tienen arreglos de otros objetos, teniendo que calcular no solo datos primitivos sino datos altamente estructurados. Para ser lo más eficientes posibles en términos de memoria es esencial el correcto uso de las referencias para evitar hacer copias innecesarias de datos pesados, aparte de claramente el uso de la memoria dinámica y su correcta liberación.

El problema también plantea varias restricciones que deben ser verificadas en el código, tanto las más banales como que la fecha sea válida, hasta que el huésped no

tenga reservas para la fecha en la que quiere reservar o que el alojamiento esté disponible durante toda la totalidad de la reserva, es esencial manejar estos casos correctamente para conservar la lógica interna del problema. La funcionalidad 3 es lo más esencial del programa y por ello debe ser implementado con mucho cuidado.

Tras este análisis inicial, consideramos que tenemos una base para la posterior implementación, detectamos antes de empezar a codificar posibles dificultades como la correcta vinculación de los huéspedes a las reservaciones, sin embargo esto es algo que se seguirá analizando.