Informe del desafío 2

# Hugo Esteban Barrero García – Juan Angel Omaña Montañez

# Universidad de Antioquia

# Informática 2

# Aníbal José Guerra Soler – Augusto Salazar

# Medellín, Colombia

Mayo, 2025

**1. Introducción**

El presente informe acompaña la primera entrega del Desafío II de la asignatura Informática II, correspondiente al diseño del **Diagrama de Clases UML** para el sistema denominado **UdeAStay**. Este sistema busca representar una plataforma de gestión de reservas de alojamiento entre anfitriones y huéspedes, modelando de manera clara las entidades involucradas, sus atributos, métodos y relaciones.

**2. Objetivo del diseño**

El objetivo principal fue desarrollar un modelo de clases que permita:

* Gestionar alojamientos ofrecidos por anfitriones.
* Permitir que huéspedes realicen reservas.
* Registrar información detallada sobre cada reservación.
* Validar disponibilidad y fechas.
* Controlar estructuras dinámicas de datos, sin uso de STL ni herencia.

El diagrama se ha elaborado en concordancia con las restricciones del enunciado: no uso de herencia, buena modularidad, manejo dinámico de memoria, encapsulamiento, y uso eficiente de tipos de datos.

**3. Clases implementadas**

Se definieron seis clases principales, cada una representando una entidad central del sistema:

* **Fecha**: utilizada para representar fechas de entrada, pago y corte. Se conecta con Reservacion y UdeAStay.
* **Reservacion**: modela las reservas hechas por los huéspedes. Se conecta a Huesped, Alojamiento y Fecha.
* **Huesped**: representa a los usuarios que realizan reservas. Se conecta a Reservacion.
* **Alojamiento**: representa los espacios ofrecidos. Se conecta a Reservacion y Anfitrion.
* **Anfitrion**: representa al usuario que administra alojamientos. Se conecta directamente con Alojamiento.
* **UdeAStay**: clase principal del sistema. Administra las listas de todas las entidades anteriores y controla la lógica general del programa.

**4. Relaciones entre clases**

El diseño considera únicamente relaciones necesarias y funcionales, respetando principios de cohesión y bajo acoplamiento:

* UdeAStay se conecta con todas las clases porque gestiona directamente las listas dinámicas.
* Reservacion se conecta a Huesped y Alojamiento, además de usar Fecha para los campos fechaEntrada y fechaPago.
* Alojamiento está vinculado a un único Anfitrion, lo que permite acceder a las reservas del anfitrión a través de sus alojamientos.

**5. Implementación de clases**

A continuación se detallan las decisiones clave tomadas en la implementación de las primeras clases:

**5.1 Clase Fecha**

Se diseñó como una clase sencilla para representar días, meses y años. No utiliza memoria dinámica, pero se le incluyó constructor por defecto, constructor de copia y operador de asignación para cumplir con la regla de los tres.  
Se agregó el método toString() que retorna la fecha como texto dinámico. También se definieron operadores == y < para comparar fechas de forma clara, facilitando operaciones como disponibilidad.

**5.2 Clase Anfitrion**

Modela al usuario dueño de alojamientos. Maneja un arreglo dinámico de punteros a alojamientos, y permite agregar nuevos alojamientos, ver reservas dentro de un rango, y anularlas de forma simulada.  
La memoria dinámica se gestiona con redimensionamiento progresivo para mantener eficiencia. El uso de punteros favorece la referencia cruzada entre anfitrión y alojamiento sin duplicar datos.

**5.3 Clase Alojamiento**

Representa los lugares ofertados por los anfitriones. Contiene información textual y numérica del espacio, su precio, tipo y amenidades. Estas últimas se gestionan mediante una codificación bitmask, que permite activar o desactivar múltiples características de forma eficiente.  
Cada alojamiento mantiene su propio historial de fechas reservadas, y permite verificar disponibilidad, registrar nuevas reservas, y mostrar las que se encuentren dentro de un intervalo.  
Se incluyeron métodos privados como copiarTexto y redimensionarReservas para mantener encapsulamiento, y se mejoró la seguridad al reemplazar funciones peligrosas como strcpy por strncpy.

**6. Pruebas realizadas**

Se desarrollaron pruebas específicas para cada clase, verificando su comportamiento esperado:

* En Fecha se probaron comparaciones, copias, conversiones a texto y operadores personalizados.
* En Alojamiento se verificó la gestión de amenidades, el registro de reservas, la disponibilidad por fechas y la impresión de datos.
* En Anfitrion se comprobó la correcta asociación con alojamientos y la capacidad de consultar y anular reservas.

### **10. Bibliografía y Referencias**

* Documentación oficial de Qt. Disponible en: <https://doc.qt.io>
* Apuntes de clase de la asignatura Informática II.
* Archivo entregado “Desafío I v1.pdf” con instrucciones oficiales del reto.
* Explicaciones técnicas desarrolladas durante el trabajo práctico.
* Discusiones y ayudas recibidas a través del canal del curso.
* Asistencia generada con ChatGPT (OpenAI).
* Consultas complementarias realizadas con DeepSeek.
* Apoyo puntual de Gemini (Google).
* Repositorio del proyecto: [https://github.com/Hugo-24/Desafio-2](https://github.com/Hugo-24/Desafio-21).