Desafío 2: UdeAStay

Elaborado por:

Thomas Mejía Moncada

Ivan Darío Gonzalez Santana

Grupo: 06

Aníbal José Guerra soler

Augusto Enrique Salazar

Profesores

Universidad de Antioquia “UdeA”

Facultad de ingeniería

Programa de: ingeniería electrónica/Telecomunicaciones

Curso: Informática II

Medellín

16 de mayo de 2025

Introducción

Este trabajo aborda el desarrollo de UdeAStay, una plataforma de gestión de estadías hogareñas (*homestays*) diseñada para conectar anfitriones y huéspedes, facilitando la reserva de alojamientos en el departamento de Antioquia.

El proyecto se enmarca en los principios de la Programación Orientada a Objetos (POO**)** en C++ manejo de archivos para garantizar un sistema robusto y escalable. Las funcionalidades clave incluyen: reservas de alojamientos con filtros personalizados, cancelaciones, consultas de disponibilidad y actualización de históricos, todo ello respaldado por un diseño eficiente de estructuras de datos y algoritmos que optimizan el uso de recursos.

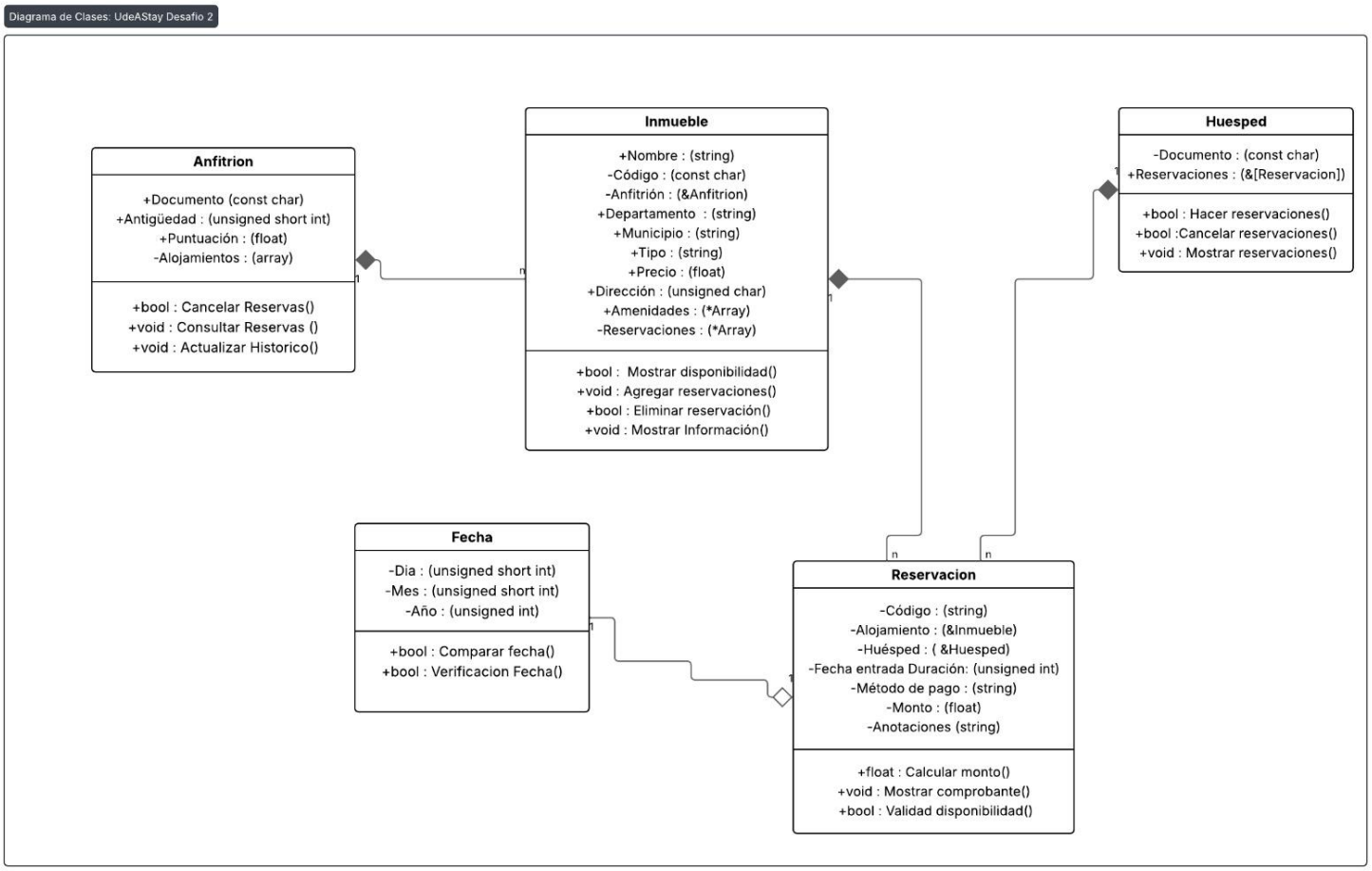
Analisis textual del problema

El desafío consiste en desarrollar UdeAStay, un sistema de gestión de alojamientos temporales (*homestays*) que conecta anfitriones y huéspedes, utilizando Programación Orientada a Objetos (POO) en C++ bajo restricciones específicas. El sistema debe manejar cuatro entidades principales: Alojamientos, Reservaciones, Anfitriones y Huéspedes, cada una con atributos y relaciones definidas. Los alojamientos tienen características como tipo (casa/apartamento), precio por noche, amenidades (piscina, aire acondicionado, etc.) y un calendario de fechas reservadas. Las reservaciones requieren validar disponibilidad, evitar solapamientos de fechas y generar comprobantes con formato específico (ej: *"martes, 20 de mayo del 2025"*). Los anfitriones administran sus propiedades y consultan reservas, mientras los huéspedes buscan, reservan o cancelan alojamientos aplicando filtros (municipio, precio máximo o puntuación mínima del anfitrión).

Las funcionalidades clave incluyen:

1. Autenticación por roles (huésped/anfitrión) con menús diferenciados.
2. Reservas con búsqueda flexible (por código o filtros) y validación de conflictos.
3. Cancelaciones accesibles para ambos roles.
4. Actualización de histórico, moviendo reservas pasadas a archivos permanentes.
5. Métricas de rendimiento (iteraciones y memoria usada), críticas para evaluar eficiencia.

Diagrama de clase (UML)



descripción de cada clase

reservación
-String Código -Alojamiento (referencia a el objeto alojamiento) -Huésped (referencias a el objeto huésped) -Fecha entrada -Unsigned char Duración (noches de estadía) -String Método de pago (pse, tcredito) -Float Monto -String Anotaciones (comentarios del huésped)
+Float Calcular monto () +Void Mostrar comprobante () +bool Validad disponibilidad ()

anfitrión
+String Documento +Unsigned short Antigüedad +Float Puntuación -Alojamientos (lista de objetos alojamientos)
+bool Cancelar reservación () +Void Consultar reservación () +Void Actualizar históricos ()

huésped
-String Documento +Reservaciones (lista de objetos reservaciones)
+bool Hacer reservaciones () +bool Cancelar reservaciones () +Void Mostrar reservaciones ()

Fecha
-Unsigned char Día -Unsigned char Mes -Unsigned short Año
+Comparación fecha () +bool Verificar fecha ()

Alojamiento
+String Nombre (nombre del alojamiento) -Unsigned short Código (código de alojamiento) -Anfitrión (referencia a un objeto anfitrión) +String Departamento +String Municipio +StringTipo (casa o apartamento) +Float Precio (precio por noche) +String Dirección (dirección física) +String Amenidades (lista de cosas, piscina, tv, patio, etc.) -Reservaciones (lista de objetos reservaciones (reservaciones futuras o activas))
+bool mostrar Disponibilidad () +Void agregar Reservaciones () +bool Eliminar reservación () +Void Mostrar Información ()