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.

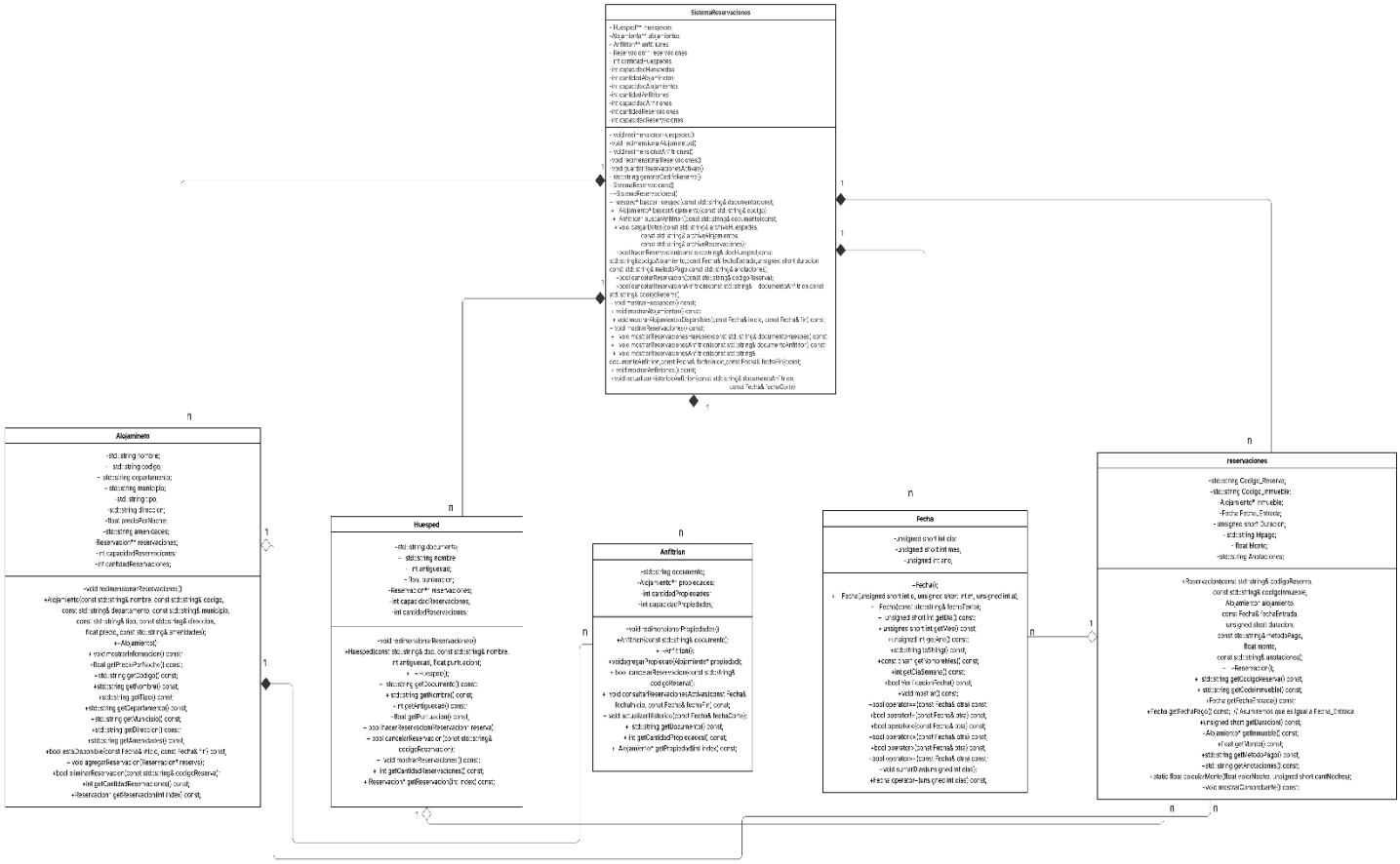
Analisis textual del problema

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)



solución al problema

Estructuras Principales

1. **Arreglos Dinámicos de Punteros:**
    - Huesped** huespedes: Arreglo dinámico de punteros a objetos Huesped
    - Alojamiento** alojamientos: Arreglo dinámico de punteros a objetos Alojamiento
    - Anfitrion** anfitriones: Arreglo dinámico de punteros a objetos Anfitrion
    - Reservacion** reservaciones: Arreglo dinámico de punteros a objetos reservación

2. Variables de Control:

Para cada arreglo hay dos variables:

cantidadX: Número actual de elementos almacenados

capacidadX: Capacidad máxima actual del arreglo

3. Relaciones entre Clases

1. **SistemaReservaciones**:
    - Contiene y gestiona todos los objetos del sistema
    - Coordina las interacciones entre huéspedes, anfitriones, alojamientos y reservaciones

2. **Huesped**:
    - Puede tener múltiples reservaciones
    - Relacionado con Alojamiento a través de Reservacion
3. **Anfitrion**:
    - Puede tener múltiples propiedades (Alojamientos)
    - Relacionado con Alojamiento directamente
4. **Alojamiento**:
    - Pertenece a un Anfitrion
    - Puede tener múltiples reservaciones
    - Relacionado con Huesped a través de Reservacion
5. **Reservacion**:
    - Conecta un Huesped con un Alojamiento
    - Contiene información específica de la reserva (fechas, pago, etc.)

4. Gestión de Memoria

- Redimensión dinámica: Cuando un arreglo se llena, se duplica su capacidad mediante las funciones redimensionarX()
- Liberación de memoria: El destructor se encarga de liberar toda la memoria asignada

Ventajas de esta Estructura

1. **Flexibilidad**: Permite crecer dinámicamente según las necesidades
2. **Acceso rápido**: Búsqueda directa por índices en los arreglos
3. **Relaciones claras**: Las conexiones entre objetos están bien definidas
4. **Encapsulamiento**: Cada clase maneja su propia información y comportamiento

Búsqueda Secuencial:

- Implantación en métodos como buscarHuesped(), buscarAlojamineto(), buscarAnfitrion()
- Recorre arreglos linealmente hasta encontrar coincidencia

1. **Relaciones entre Objetos**:
    - Un Anfitrion contiene punteros a sus Alojamientos
    - Un Alojamiento contiene punteros a sus Reservaciones
    - Un Huesped contiene punteros a sus Reservaciones
- **Navegación Bidireccional**:
    - De Reservación → Alojamiento → Anfitrión
    - De Reservación → Huésped
    - De Anfitrión → Alojamiento → Reservación → Huésped

Conclusión

El desarrollo de UdeAStay representa un esfuerzo significativo por construir un sistema completo de gestión de estadías hogareñas, utilizando los principios de la Programación Orientada a Objetos en C++ y el manejo manual de archivos y memoria dinámica. Este proyecto no solo cumple con los requisitos funcionales de autenticación, reservas, cancelaciones y actualización de históricos, sino que también demuestra una implementación cuidadosa de estructuras de datos eficientes, relaciones entre clases y gestión responsable de los recursos del sistema.

La arquitectura basada en arreglos dinámicos de punteros, junto con las relaciones bien definidas entre las clases principales (Huésped, Alojamiento, Anfitrión y Reservación), permitió establecer una navegación coherente y flexible entre los elementos del sistema. A pesar de las limitaciones impuestas —como la no utilización de STL—, se logró construir un sistema robusto y escalable, capaz de adaptarse a futuras expansiones.

Desde una perspectiva reflexiva, el mayor reto no fue únicamente técnico, sino conceptual: diseñar un sistema que reflejara de forma realista las interacciones del mundo real, respetando la lógica de negocio y asegurando la integridad de los datos, todo dentro del paradigma POO y sin herramientas avanzadas externas. Resolver problemas como la validación de disponibilidad, la prevención de solapamientos de fechas, o la gestión bidireccional entre entidades, exigió una comprensión profunda de la programación estructurada y orientada a objetos.

Este proyecto es testimonio del valor del pensamiento lógico, la disciplina en el diseño de clases y la perseverancia ante los errores. En definitiva, UdeAStay no solo es una solución funcional a un problema real, sino también una experiencia formativa enriquecedora que evidencia el poder de la programación orientada a objetos bien aplicada y el esfuerzo constante por mejorar.