**Análisis del Problema**

El Desafío II aborda la creación de un sistema de administración para el mercado de estancias en UdeAStay, aplicando los fundamentos de la programación orientada a objetos (POO) en C++. Esta herramienta debe facilitar la interacción entre dos tipos de usuarios: los huéspedes y los anfitriones. Cada grupo contará con funciones específicas, además de acceso a características comunes del sistema.

**Estructura general de la solución**

La solución se centra alrededor de una clase principal o controladora, que organiza la lógica general del software. Esta clase fundamental es la responsable de mostrar el menú principal al usuario y activar las funciones correspondientes según el tipo de usuario, ya sea huésped o anfitrión.

Para una correcta modularidad y mantenimiento del código, se han creado dos archivos. h y sus correspondientes. cpp: uno para el módulo de anfitriones y otro para el de huéspedes. Cada módulo incluye las clases, atributos y métodos necesarios para manejar la lógica propia de su categoría de usuario.

**Funciones compartidas y específicas**

**Funciones de uso general:**

Carga y actualización de datos: Esta función se realiza automáticamente al iniciar el sistema, permitiendo cargar información persistente de huéspedes, anfitriones, alojamientos y reservaciones desde archivos. También se ejecuta al concluir una acción que altera los datos, lo cual asegura la consistencia del sistema.

Se utiliza la biblioteca estándar para manejar la lectura y escritura de archivos.

Los datos deben almacenarse en archivos estructurados, con un formato que facilite su lectura e interpretación.

**Funciones específicas del huésped:**

**Reservar alojamiento:** Permite filtrar por ubicación, fechas, precio máximo o puntuación del anfitrión.

**Consultar y cancelar reservaciones:** Para gestionar sus reservas activas.

**Funciones específicas del anfitrión:**

**Consultar reservaciones:** Acceso a todas las reservas de los alojamientos que maneja.

**Actualizar histórico:** Posibilita archivar las reservas anteriores y mantener organizada la información.

**Funcionalidad adicional:**

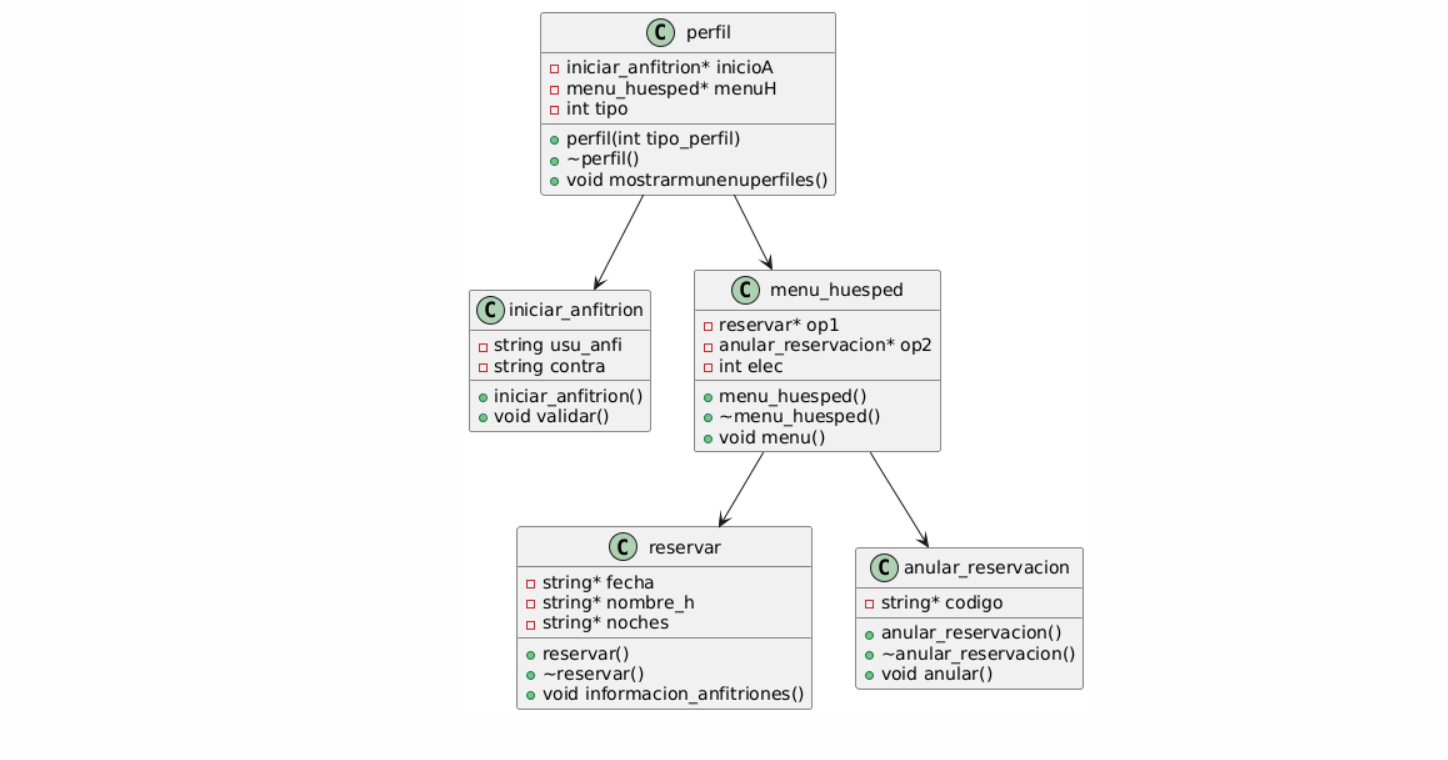
**Medición del uso de recursos:** El sistema debe registrar y mostrar la cantidad de iteraciones realizadas y el uso de memoria al final de cada función, con fines académicos para el análisis de rendimiento.

**Consideraciones de diseño**

Es esencial garantizar la validez de las operaciones, como asegurarse de que un alojamiento esté disponible antes de realizar una reserva, o verificar que un huésped no tenga reservas duplicadas en la misma fecha.

El sistema debe emplear eficientemente las estructuras de datos y aplicar principios de encapsulamiento, abstracción y reutilización.

Se deben implementar constructores, destructores, métodos get y set, y sobrecarga de funciones u operadores cuando sea necesario.

**Diagrama de clase**