**Informe Desafío II**

**Alumno: Brayan Stick López López**

**Docente: Augusto y Anibal**

**UNIVERSIDAD DE ANTIOQUIA**

**Teoría de Informática**

**Caucasia, Antioquia**

**2025**

# Informe desafío II: Mercado de estadías hogareñas UdeAStay.

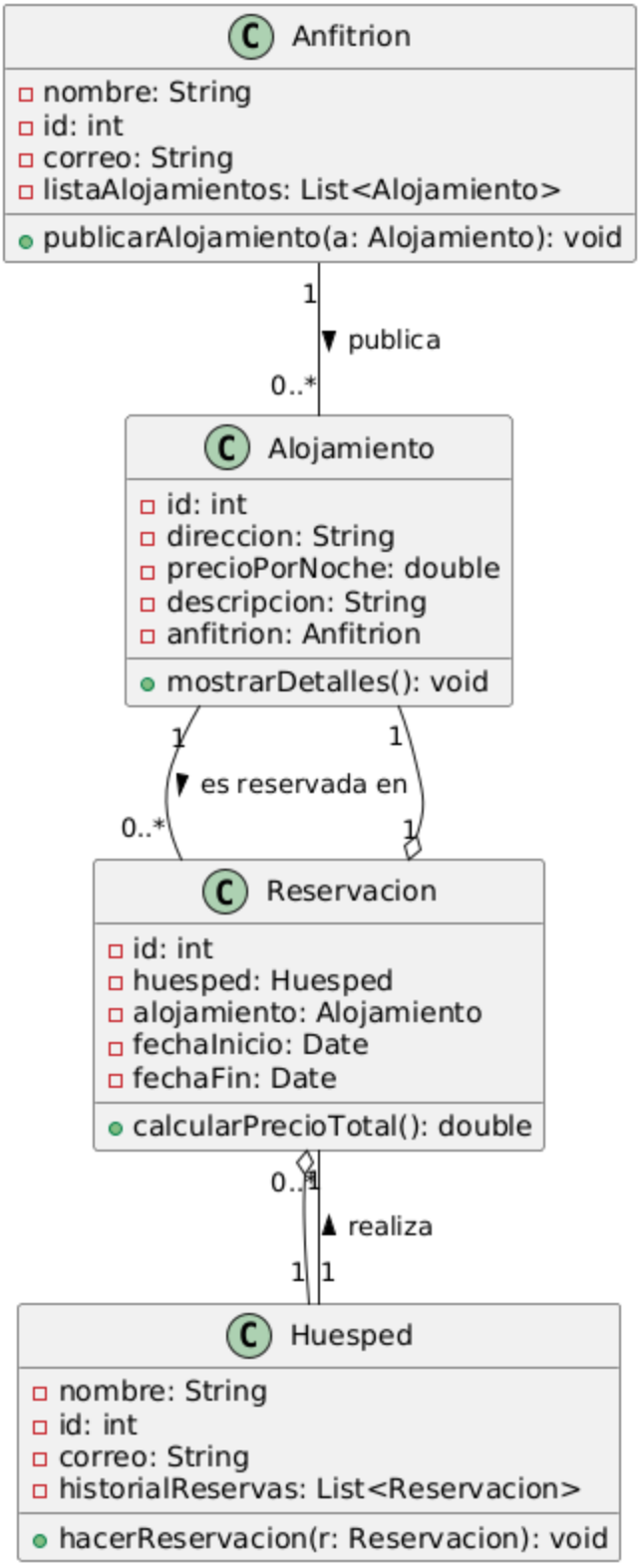
**Carga:** En el desafío mi análisis al problema se basa en crear una base de datos a través de archivos en el que contendrán la información de los usuarios el cual se podrá modificar e implementar al código para tener tanto organización e información necesaria, implementare cual sea necesaria la función de leer línea por línea o carácter por carácter el archivo el que vea mas conveniente para leer la información del huésped, anfitrión, alojamiento y reservaciones, también se creara un apartado de tipo de inicio y registro para que el usuario pueda interactuar.

- Captura de datos y almacenamiento.
- Lectura de archivos txt.
- Algoritmo con condicionales para apartado de huésped y anfitrión.
- El programa será actualizable

**Reserva:** en este apartado implementare la función de que el huésped usuario pueda acceder a una lista que se mostrara en la terminal tanto fecha, municipio y una cantidad de noches, esto pienso implementar como primero dar una lista y al seleccionar número de opción del alojamiento entregue la información completa en una lista de aquella opción tomada por el huésped

- Apartado de opciones.
- Mostrar alojamientos

**Diagrama de Clases - UdeAStay**



**Publicar alojamiento (Anfitrión)**
El anfitrión puede registrar múltiples alojamientos, lo cual se implementa agregando objetos Alojamiento a su lista interna.
**Realizar reservación (Huésped)**
El huésped selecciona un alojamiento disponible y define fechas de inicio y fin. Se crea un objeto Reservacion que se asocia tanto al huésped como al alojamiento.
**Cálculo del precio total**
Para una reservación, se calcula el precio total multiplicando la cantidad de noches por el precio por noche del alojamiento, usando diferencias entre fechas.

```
double Reservacion::calcularPrecioTotal() {
    // Calcula la duración de la reserva en días
    int dias = fechaFin - fechaInicio;
    // Multiplica por el precio del alojamiento por noche
    return dias * alojamiento.getPrecioPorNoche();
}
```

```
void Huesped::hacerReservacion(Reservacion r) {
    // Agrega la reservación al historial del huésped
    historialReservas.push_back(r);
}
```

```
void Anfitrion::publicarAlojamiento(Alojamiento a) {
    // Publica un nuevo alojamiento en la lista del anfitrión
    listaAlojamientos.push_back(a);
}
```

**Problemas de desarrollo que se afrontaron**

- **Diseño de clases**: Determinar qué responsabilidades y atributos debía tener cada clase y evitar duplicidades.
- **Gestión de fechas**: Se necesitó una lógica clara para manejar diferencias entre fechas de inicio y fin, y evitar errores de cálculo.
- **Interrelaciones**: Garantizar que las relaciones entre clases no generaran referencias cíclicas no deseadas o problemas de memoria.

Al principio, tenía en mente una solución mucho más simple: solo pensaba en usar dos clases, Usuario y Alojamiento, y desde ahí manejar todo. Pero a medida que fui avanzando y

entendiendo mejor cómo debía funcionar el sistema, me di cuenta de que eso se iba a quedar corto y que iba a complicar más las cosas en lugar de simplificarlas.

Por eso, decidí organizar mejor el modelo y separar bien las responsabilidades. Terminé usando clases distintas para Anfitrión, Huésped, Alojamiento y Reservación, lo cual me permitió manejar las relaciones de manera más clara y lógica. Evité usar herencia para no enredarme más de la cuenta, y preferí mantener todo lo más directo posible.

Para la implementación final, pienso que sería bueno guardar los datos usando archivos