**Informe Desafío II UdeA Stay**

**Estudiantes**: Esteban Rodríguez Monsalve   
**Profesor**: Aníbal José Guerra Soler

**Curso**: Informática II

**Introducción**

En este proyecto se propuso desarrollar un sistema que simula el funcionamiento de estadías hogareñas llamada “**UdeA Stay”**. Este sistema permite conectar anfitriones, que ofrecen alojamientos, con huéspedes, que desean reservar estancias en distintas zonas designadas.

### **Objetivo del Sistema**

El objetivo principal es brindar un sistema que:

* Administrar los alojamientos disponibles.
* Permitir a los huéspedes buscar, reservar o cancelar estancias.
* Permitir a los anfitriones gestionar sus alojamientos y consultar las reservas realizadas.
* Mantener un historial de reservas y gestionar la disponibilidad de fechas.
* Realice operaciones como cálculo de costos, validación de métodos de pago y evaluación de compatibilidad entre usuarios y alojamientos.

### **Enfoque de Desarrollo**

Para resolver este desafío, se utilizó el enfoque de Programación Orientada a Objetos (POO). Esto permitió modelar de forma clara las entidades del sistema, estableciendo relaciones, responsabilidades y flujos entre ellas.

El sistema se implementó en C++, sin utilizar STL ni herencia, priorizando un diseño limpio, modular y con uso de punteros estáticos para la gestión de datos en memoria.

**Estructura de Clases**

El sistema se compone de 6 clases principales:

* **Sistema**: Clase central que gestiona todos los procesos, enlaza y coordina el resto de clases. Controla la carga y guardado de datos.
* **Anfitrión**: Representa a los usuarios que administran alojamientos. Pueden agregar alojamientos y consultar reservas.
* **Huésped**: Representa a los usuarios que realizan reservaciones. Pueden buscar, reservar o cancelar una reserva.
* **Alojamiento**: Contiene la información del alojamiento (ubicación, precio, servicios, disponibilidad).
* **Reservación**: Representa las transacciones de hospedaje entre huésped y alojamiento, incluye fechas, duración, costos y forma de pago.
* **GestiónArchivos**: Permite cargar y guardar datos persistentes del sistema (alojamientos y reservaciones), además de administrar históricos.

### **Relaciones entre Clases**

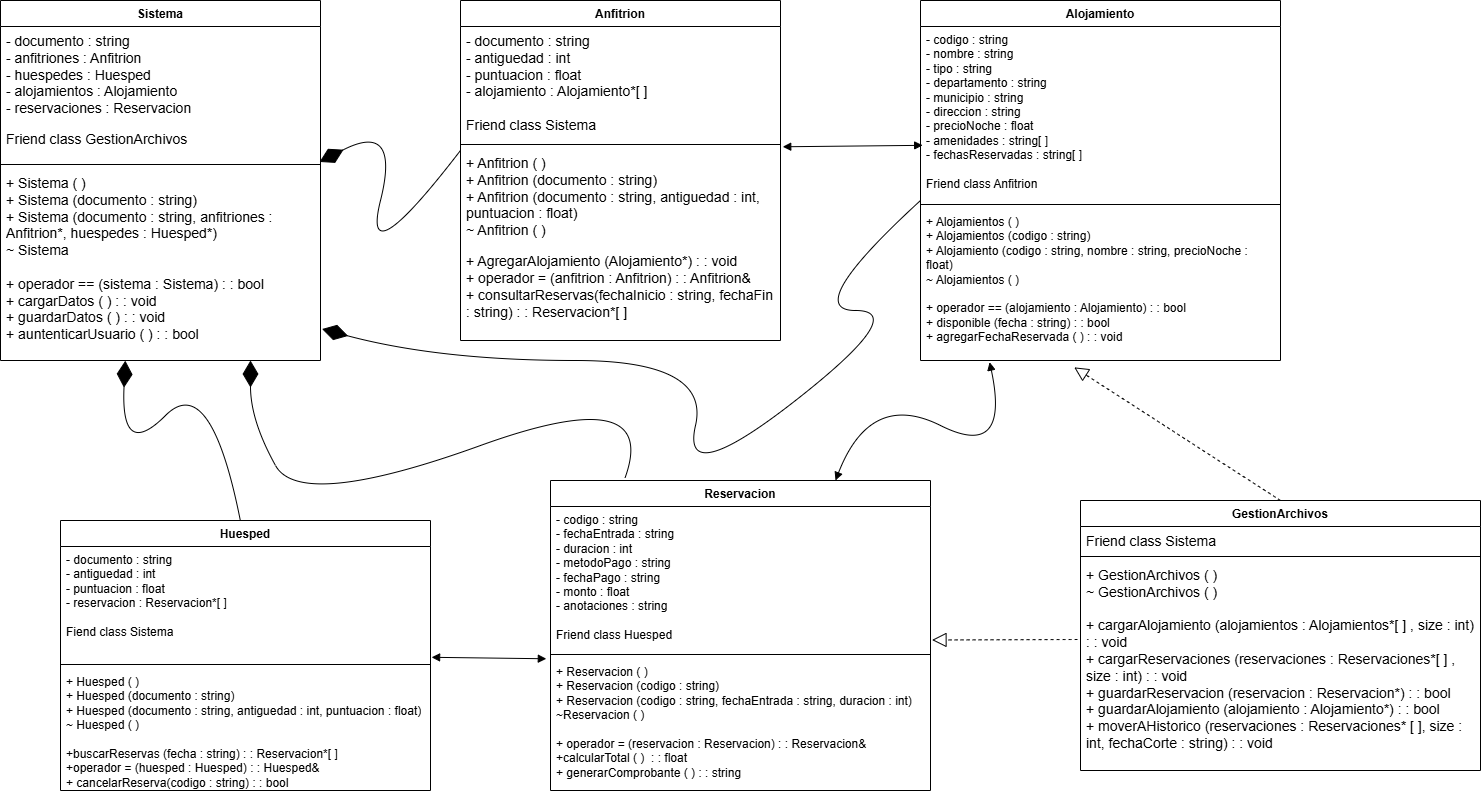
A través del diagrama UML podremos ver las relaciones claramente:

* **Sistema** mantiene instancias y acceso directo a las demás clases (amigos).
* **Anfitrión** administra uno o varios **Alojamientos**.
* **Huésped** posee varias Reservaciones.
* **Reservación** contiene datos de conexión entre **Huésped** y **Alojamiento**.
* **GestiónArchivos** se encarga de manejar datos de **Alojamientos** y **Reservaciones**.

### **Funcionalidades Clave Implementadas**

* Inicio de sesión como huésped o anfitrión.
* Realización y anulación de reservas.
* Consulta de reservas activas o históricas.
* Verificación de disponibilidad.
* Cálculo de costos totales.
* Carga y guardado de datos desde/hacía archivos.
* Interfaz interactiva por menú de consola.

**Diagrama de clases UML**



**Base de datos**

Se van a utilizar archivos de texto plano con una estructura como base de datos que nos permitirá guardar la información para poder acceder a ella.

las tengo pensado diseñar 5 base de datos de una forma sencilla utilizando “ | “ como separador entre cada dato que se pide y que considero importante para cada clase.