**Informe Análisis Desafío Numero 2.**

**Jeronimo Herrera Espinosa.  
1046667210.**

**Introducción.**

El propósito de este segundo desafío es crear un sistema de reserva de alojamientos a partir de tres actores principales, Huéspedes, Anfitriones y Alojamientos, este mismo sistema debe proporcionar ciertas funciones a los huéspedes como las de consultar alojamientos, crear y borrar reservas, etc.

En este informe se detallarán precisamente cual fue el enfoque adoptado, explicando principalmente el diagrama de clases UML simplificado que fue presentado en clase, y la explicación de la creación y uso de los archivos externos utilizados para el manejo de los datos en este desafío.

**descripción del problema.**

En este desafío, seremos los programadores encargados de la aplicación “UdeAStay”, la cual, como ya fue mencionade, es un sistema de reserva de alojamientos dirigida a huéspedes y anfitriones de alojamientos, en donde estos podrán entre sus funciones:

-huéspedes:

* Reservar un alojamiento a partir de una fecha, un municipio especifico y una cantidad de noches, permitiendo también filtrar dependiendo del costo por noche, puntuación mínima del anfitrión, o incluso ambas, se debe tener también en cuenta que estas reservas no se pueden “solapar” las unas a las otras en cuestión de fechas,

-anfitriones:

* Consultar las reservaciones hechas en alguno de los alojamientos que el anfitrión posea

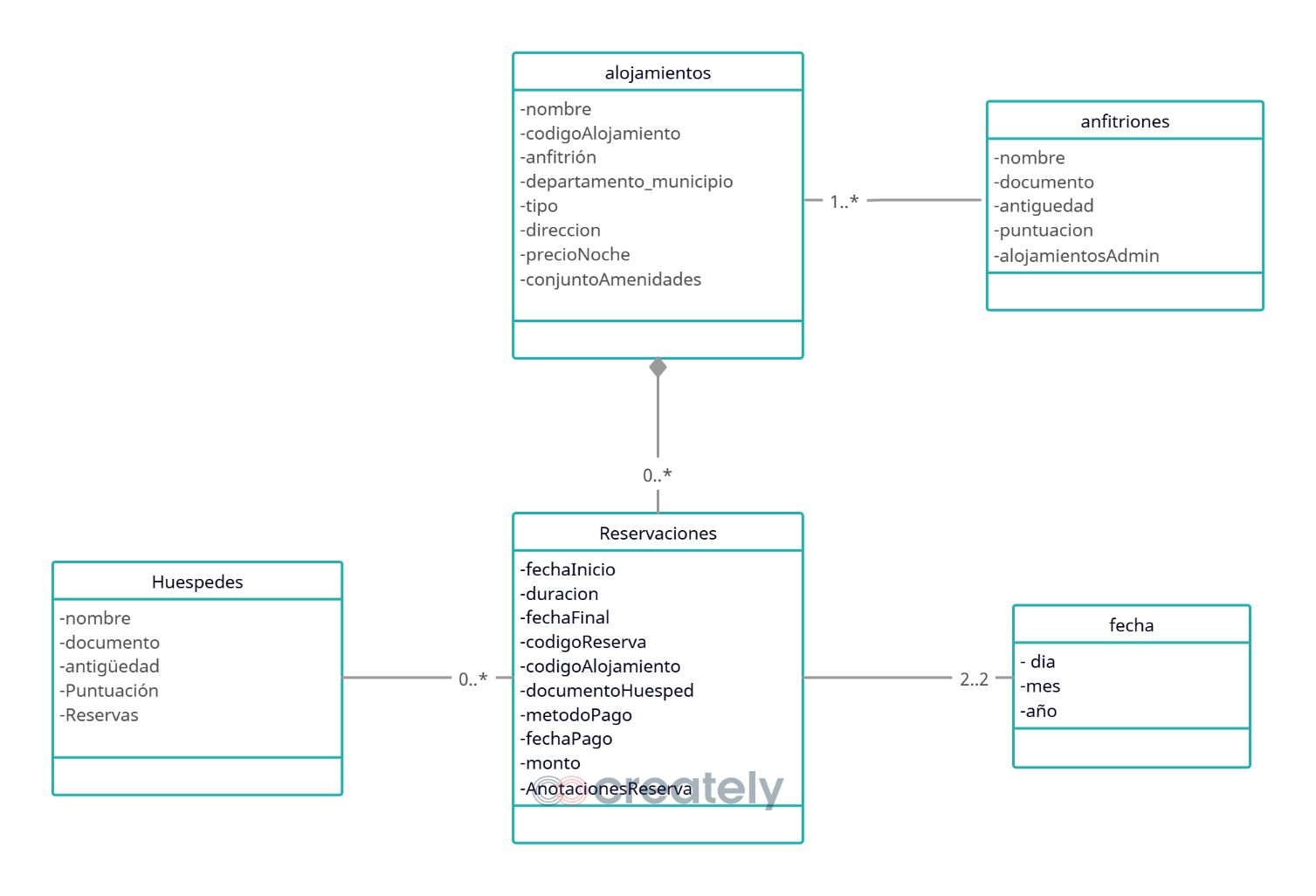
Adicionalmente tanto huéspedes como anfitriones tienen la posibilidad de anular una reservación existente.

Otra función que se debe implementar es la de actualizar el histórico, en donde se guardaran todas las reservaciones de fechas pasadas para luego eliminarlas, para esta función, el usuario anfitrión deberá proporcionar la fecha actual, la cual también servirá como punto de partida para establecer los próximos 12 meses que se habilitaran para reservaciones.

Otro requerimiento para este desafío será mostrar el total de memoria que se está consumiendo y la cantidad de iteraciones requeridas para completar cada función.

**análisis del enfoque adoptado.**

Teniendo en cuenta los requerimientos anteriores:

1. **Clases:** para la óptima solución de este desafío decidí la creación de 5 clases, siendo estas: Alojamientos, Anfitriones, huéspedes, Reservaciones y fecha.
2. **Archivos externos .txt:** para este desafío decidí hacer uso de 6 archivos principales
   1. **Huespedes.txt:** en este archivo se guardarán todos los atributos de cada uno de los objetos de la clase “Huéspedes”, algunos de estos atributos siendo número de documento, antigüedad en la plataforma, puntuación, etc.
   2. **Anfitriones.txt:** en este se guardarán los atributos de cada uno de los objetos de la clase “Anfitriones”, algunos de estos atributos siendo número de documento, puntuación, alojamientos que administra.
   3. **Alojamientos.txt:** en este se guardarán los atributos de cada uno de los objetos de la clase “Alojamientos”, algunos de estos atributos siendo el nombre, un código identificador, su anfitrión responsable.
   4. **Reservas.txt:** en este se guardarán los atributos de cada uno de los objetos de la clase “Reservas”, algunos de estos atributos siendo fecha de entrada, duración (en noches), un código identificador.
   5. **Historicos.txt:** en este archivo se llevará un recuento de las reservas que ya pasaron, es decir aquellas cuya fecha de registro es anterior a la fecha actual.
   6. **Codigos.txt:** como se dieron cuenta en algunas de las clases se requieren códigos identificadores, en este archivo .txt se guardarán estos códigos ya actualizados, para no repetir códigos.
3. **Funciones:** las funciones que usare para la solución de este desafío serán:
   1. **IngresoPlataforma:** esta función realizara la verificación y tipo de usuario según una credencial (documento de identidad) ingresado por el usuario, para desplegar las funcionalidades dependiendo si es huésped o anfitrión.
   2. **reservarAlojamiento:** esta función es única de los huéspedes y como su nombre lo indica esta función se encarga de realizar las reservas, desplegando los alojamientos disponibles y permitiendo al usuario filtrar por ciertos parámetros, esta función al finalizar debe desplegar el código de la reserva, nombre del usuario, código del alojamiento, la fecha del inicio y de finalización (estas fechas deben de estar en el formato: nombreDia, día “de” nombreMes “del” año.
   3. **anularReserva:** tanto huéspedes como anfitriones deben de tener la posibilidad de anular una reserva existente.
   4. **consultarReservas:** esta función permite a un anfitrión visualizar las reservas realizadas para un alojamiento de su posesión.
   5. **actualizarHistorico:** esta función actualiza el archivo.txt “histórico” con las fechas pasadas con respecto a la fecha actual (ingresada por el usuario).

**Todas estas funciones deben desplegar, al finalizar, la cantidad de iteraciones realizadas y el total de memoria utilizado.**

**Conclusión.**

Haciendo uso de las clases, archivos externos y funciones ya mencionadas, buscaremos crear un sistema que cumpla con las funcionalidades requeridas, y que mantenga un sano equilibrio entre la facilidad de lectura y la eficiencia de iteraciones y almacenamiento utilizado.