Universidad de Antioquia
Facultad de Ingeniería
Ingeniería Electrónica
Informática II

Desafío II: UdeAStay

Sistema de gestión de estadías hogareñas

Estudiantes:
Andrés Camilo Rosero Villarreal
Santiago Ortiz Vargas

Medellín, Colombia
2025

Descripción general

Se debe desarrollar un sistema llamado UdeAStay para gestionar un mercado de estadías hogareñas. Este sistema permite a huéspedes y anfitriones interactuar con alojamientos, hacer o administrar reservaciones, y mantener un control de la información mediante archivos.

Funcionalidades principales

- Carga automática de datos desde archivos (no visible en el menú).
- Ingreso por perfil (huésped o anfitrión) con menú personalizado.
- Reservar alojamiento, aplicando filtros por municipio, precio y puntuación (huéspedes).
- Anular reservación (huésped o anfitrión).
- Consultar reservaciones activas por parte de anfitriones.
- Actualizar histórico para mover reservas antiguas.
- Medición de recursos: memoria usada y número de iteraciones.

Enfoque de desarrollo

1. **Modelado de datos**

   Se definieron 5 clases para estructurar el sistema:

   - Huésped
   - Anfitrión
   - Alojamiento
   - Reservación
   - Fecha

   Usaremos estas 5 clases aplicando el enfoque de programación orientada a objetos (POO), lo que permitirá una gestión más organizada y escalable de la información.

   Las clases Huésped y Anfitrión compartirán algunas características comunes, como el número de documento, la antigüedad en la plataforma y la puntuación. Sin embargo, cada una tendrá comportamientos y asociaciones distintas:

   El huésped podrá realizar múltiples reservaciones, y cada reservación estará asociada a un alojamiento específico, validando que no haya solapamiento de fechas.

   El Anfitrión será responsable de uno o más alojamientos, y podrá consultar y gestionar las reservaciones que estén asociadas a ellos.

   El modelo también incluirá una clase auxiliar de Fecha para facilitar la manipulación de intervalos, comparaciones y validaciones temporales, fundamentales para las reservas y el histórico.

Este diseño facilita la separación de responsabilidades y simplifica la implementación de funcionalidades como la reserva, anulación y actualización de dato

2. Diseño de almacenamiento

Para la persistencia de datos, se utilizarán archivos de texto plano (.txt), permitiendo almacenar y recuperar la información de manera estructurada y sencilla.

Se maneja un archivo por cada entidad principal del sistema:

- huéspedes.txt
- anfitriones.txt
- alojamientos.txt
- reservaciones.txt

Además, se incluirá un archivo adicional:

- histórico_reservas.txt

Este diseño facilita la carga y actualización de datos, y permite mantener un historial independiente de reservaciones pasadas para consulta futura y optimización del rendimiento del sistema.

3. Funciones no visibles en el menú

Estas funciones son esenciales para el funcionamiento del sistema, pero no deben ser accesibles directamente por el usuario desde la interfaz principal (menú). Se ejecutan automáticamente al iniciar o cerrar el programa, o cuando se actualizan datos internamente.

Incluyen:

Carga inicial de datos desde archivos:

Al iniciar el sistema, se deben leer los archivos .txt previamente diseñados que contienen la información de:

- Huéspedes
- Anfitriones
- Alojamientos
- Reservaciones actuales
- Reservaciones históricas

Actualización automática de datos:

Después de realizar acciones como crear o eliminar una reservación, actualizar datos de usuarios o alojamientos, se deben guardar los cambios correspondientes en los archivos adecuados sin intervención del usuario.

4. **Menú (Interfaz de Usuario por Rol)**

   El sistema debe presentar un menú diferenciado según el tipo de usuario que inicie sesión: huésped o anfitrión. Al identificarse, el sistema mostrará solo las opciones disponibles para su rol.

5. **Actualización del Histórico de Reservaciones**

   Esta funcionalidad permite trasladar reservaciones pasadas (anteriores a una “fecha de corte”) al archivo histórico, liberando las estructuras activas del sistema.

   La fecha de corte es ingresada por el anfitrión.

   No puede ser anterior a la última fecha registrada en el histórico.

   Tras la actualización, el sistema queda habilitado para aceptar nuevas reservaciones en los 12 meses siguientes a la fecha de corte.

   Esta operación optimiza el manejo de datos y mantiene vigente el calendario de reservas.

6. **Validaciones lógicas**

   Antes de permitir que el usuario ejecute ciertas acciones, el sistema debe verificar que se cumplan condiciones específicas, garantizando integridad y consistencia en los datos. Las validaciones más importantes son:

   Para crear una reservación:

   - El alojamiento debe estar disponible durante todas las fechas solicitadas.
   - El huésped no debe tener otra reserva en esas fechas.
   - Se deben cumplir los filtros indicados (precio máximo por noche, puntuación mínima del anfitrión).
   - Los datos de entrada deben tener el formato correcto y respetar los límites del sistema (por ejemplo, anotaciones hasta 1000 caracteres).

   Para anular una reservación:

   - La reservación debe existir.
   - Debe pertenecer al usuario actual (huésped o anfitrión vinculado al alojamiento).
   - La fecha de inicio no debe haber pasado (opcional, si se prohíben cancelaciones tardías).
   - Para actualizar el histórico:

- La fecha de corte debe ser válida y no anterior a lo ya registrado en el histórico.
- Las reservaciones a mover deben tener todas sus fechas anteriores a la fecha de corte.

7. Medición del consumo de recursos

Esta función es de uso académico y sirve para evaluar la eficiencia del sistema.Se ejecuta automáticamente después de cualquier funcionalidad principal.

Debe mostrar:

- Cantidad de iteraciones ejecutadas (ciclos, llamadas, etc.) para cumplir la tarea.
- Memoria utilizada, contabilizando los objetos creados en tiempo de ejecución (alojamientos, usuarios, reservaciones, etc.).