Manejo de archivos en clase anfitrión será de la siguiente forma:

**1. 📁 Archivos .txt creados**

Se diseñaron 5 archivos de texto que simulan una base de datos plana para manejar el sistema de reservas de UDESTAY. Cada archivo utiliza una estructura de campos separada por guiones - y comas , para representar datos relacionados con anfitriones, huéspedes, alojamientos y reservas.

**✅ anfitriones.txt**

* **Formato:** documento-nombre-antigüedad-puntuación-códigosAlojamientos
* **Ejemplo:**

CopiarEditar

100000001-Laura Restrepo-15-4.8-UDE001,UDE002

**✅ huespedes.txt**

* **Formato:** documento-nombre-antigüedad-puntuación-códigosReservas
* **Ejemplo:**

css

CopiarEditar

200000001-María Pérez-8-4.7-R001,R020

**✅ alojamientos.txt**

* **Formato:** código-nombre-departamento-municipio-tipo-dirección-duración-precio-amenidades
* **Ejemplo:**

bash

CopiarEditar

UDE001-Casa Verde-Antioquia-Medellín-casa-Cra 10 #12-30-180000-piscina,parqueadero

**✅ reservas.txt**

* **Formato:** códigoReserva-fechaInicio-duraciónDías-códigoAlojamiento-documentoHuesped-metodoPago-fechaPago-monto-anotaciones
* **Ejemplo:**

CopiarEditar

R001-2025-06-01-4-UDE001-200000001-TCredito-2025-05-25-720000-Vista al jardín

**✅ historico.txt**

* **Formato:** fechaEvento-tipoEvento-códigoRelacionado-descripción
* **Ejemplo:**

css

CopiarEditar

2025-06-01-CANCELACION\_HUESPED-R005-Cancelación por problemas personales del huésped

**2. 🛠 Funciones implementadas**

Se diseñaron funciones universales que permiten interactuar con cualquier archivo .txt del sistema de forma flexible y segura.

**📌 leerArchivoTexto(...)**

cpp

CopiarEditar

bool leerArchivoTexto(const string& nombreArchivo, string\* &lineas, int &cantidadLineas);

**Descripción:**  
Lee todas las líneas de un archivo .txt, las guarda en un arreglo dinámico de string, y devuelve la cantidad de líneas leídas.

**Uso:**  
Se puede usar con cualquier archivo: anfitriones.txt, reservas.txt, etc.

**📌 reescribirArchivoFiltrado(...)**

cpp

CopiarEditar

bool reescribirArchivoFiltrado(const string& nombreArchivo, bool (\*filtro)(const string& linea));

**Descripción:**  
Reescribe un archivo conservando solo las líneas que cumplan con un criterio de filtro. Se usa para eliminar registros, como cancelar reservas o dar de baja anfitriones.

**Manejo seguro:**

* Crea un temp.txt
* Copia las líneas válidas
* Luego reemplaza el archivo original

**📌 eliminarAnfitrionPorDocumento(...)**

cpp

CopiarEditar

bool eliminarAnfitrionPorDocumento(const string& documento);

**Descripción:**  
Función especializada que usa reescribirArchivoFiltrado() y elimina un anfitrión por su número de documento, recibido desde cin.

**3. ⚙️ ¿Cómo funcionan los bool en las funciones?**

* En C++, las funciones que devuelven bool retornan true o false para indicar **éxito o error**.

**📌 Ejemplo en leerArchivoTexto(...)**

cpp

CopiarEditar

if (leerArchivoTexto("anfitriones.txt", lineas, cantidad)) {

// Se pudo leer el archivo correctamente

} else {

// Ocurrió un error (archivo no existe, no se pudo abrir)

}

**📌 En filtros personalizados**

Un filtro como:

cpp

CopiarEditar

bool filtroGenerico(const string& linea) {

return linea.substr(0, documentoAEliminar.length()) != documentoAEliminar;

}

👉 Devuelve true si la línea debe conservarse, y false si debe eliminarse.

**4. 💡 Estado actual del sistema**

| **Componente** | **Estado** |
| --- | --- |
| Archivos de datos | ✅ Creados y poblados |
| Función de lectura | ✅ lista y funcional |
| Función de reescritura | ✅ lista para eliminar registros |
| Eliminación de anfitrión | ✅ Funciona desde consola |
| Consola cin | ✅ Configurada correctamente |
| Datos de prueba | ✅ 10 anfitriones, 50 huéspedes, 50 reservas, 20 históricos |

**📚 Próximos pasos sugeridos**

* Crear clases Huesped, Alojamiento, Reserva con métodos cargarDesdeArchivo() y mostrar()
* Implementar lógica de nuevas reservas y validaciones
* Añadir bitácora en historico.txt cuando se cancelan o completan acciones