UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería en Informática y Sistemas

INFORME FINAL

Proyecto: Hotel Hermanos Flores

Curso: Programación Avanzada

Docente: MSc. Ing. Hugo Manuel Barraza Vizcarra

INTEGRANTES:	**CÓDIGO:**
Dany Ronaldo Muriel Herencia	2024-119048
Mendoza Flores Jair Mark	2024-119004
Anette Matiel Quispe Huillca	2024-119051

Tacna - Perú

2025

ÍNDICE GENERAL

1. Resumen

El presente proyecto desarrolla un software de gestión hotelera orientado a mejorar la organización y el manejo de los registros del Hotel Los Hermanos Flores, un negocio familiar que actualmente utiliza herramientas poco actualizadas como hojas de cálculo. Esta situación genera desorden en la información, dificultades en el control de las operaciones y limita la interacción adecuada con los clientes. Como solución, se propone un sistema desarrollado en C++, aplicando el paradigma de la programación orientada a objetos y una estructura modular basada en archivos .h y .cpp. El software permite una gestión más ordenada de la información, incorpora la generación de facturas en archivos de texto y cuenta con un diseño escalable mediante un diagrama de clases bien definidas. De esta manera, el proyecto busca optimizar los procesos administrativos del hotel y contribuir a una mejor organización y proyección del negocio.

2. Antecedentes

El Hotel Los Hermanos Flores es un negocio familiar que, tradicionalmente, ha gestionado sus operaciones de forma manual y mediante hojas de cálculo. Esta dependencia de herramientas poco actualizadas ha derivado en desorden informativo, errores frecuentes en los registros de reservas y una limitada capacidad para atender a los clientes de manera eficiente. Ante este escenario, se hace indispensable la transición hacia un sistema automatizado que centralice la información, optimice la gestión de recursos y profesionalice la administración general del establecimiento.

3. Planteamiento del problema

a. Problema

El Hotel Los Hermanos Flores enfrenta dificultades en la gestión de sus operaciones debido al uso de herramientas manuales y poco eficientes, como hojas de cálculo. Esto genera desorganización en la información, errores en los registros y una limitada capacidad para atender a los clientes de manera eficiente.

b. Justificación

Es importante automatizar este proceso para:

i. Mejorar la eficiencia en la gestión de habitaciones, empleados y stock.
ii. Reducir errores en los registros y facturación.

iii. Optimizar la interacción con los clientes y la generación de reportes financieros.

c. Alcance

El software desarrollado tendrá las siguientes limitaciones:

i. Ejecución en consola.
ii. Gestión básica de habitaciones, empleados, stock y reservas.
iii. Generación de facturas en archivos de texto.

4. Objetivos

a. General

Desarrollar una aplicación en C++ que demuestre el uso de la programación orientada a objetos (POO) y manejo de archivos para la gestión hotelera.

b. Específicos

i. Implementar encapsulamiento, herencia y polimorfismo.
ii. Gestionar persistencia de datos mediante ficheros de texto.
iii. Aplicar algoritmos de búsqueda y ordenación.

5. Marco teórico

Conceptos sobre Programación Orientada a Objetos, flujo de archivos (fstream) y algoritmos.

a. Programación Orientada a Objetos (POO)

La Programación Orientada a Objetos (POO) es un paradigma de programación basado en el concepto de "objetos", que son instancias de clases y contienen tanto datos (atributos) como funciones (métodos) que operan sobre esos datos. Los principales pilares de la POO son la encapsulación, herencia, polimorfismo y abstracción, lo que permite crear programas modulares, reutilizables y fáciles de mantener.

b. Flujo de Archivos (fstream)

En C++, la biblioteca fstream permite manipular archivos de texto y binarios. Las principales clases que utiliza son:

- ofstream: para crear y escribir en archivos.
- ifstream: para leer archivos.
- fstream: para leer y escribir en archivos.

El flujo de archivos se maneja abriendo el archivo, realizando operaciones (lectura/escritura) y cerrándose al finalizar. Por ejemplo, se puede leer un archivo línea por línea usando funciones como getline() y cerrar el archivo con close().

c. Algoritmos

Un algoritmo es una secuencia de pasos bien definidos para resolver un problema o realizar una tarea específica. En informática, los algoritmos se estructuran en tres partes principales:

i. Entrada (input): datos iniciales.
ii. Proceso: conjunto de instrucciones o pasos a seguir.
iii. Salida (output): resultados obtenidos tras el proceso.

Los algoritmos son fundamentales en la programación, ya que permiten resolver problemas de manera eficiente y predecible, y son la base para el desarrollo de software y sistemas inteligentes.

6. Desarrollo de la propuesta

a. Tecnologías de desarrollo

El sistema se desarrolló utilizando un conjunto de tecnologías orientadas a mantener una arquitectura modular, escalable y fácilmente mantenible. A continuación, se describen los componentes tecnológicos principales.

i. Backend (Núcleo del Sistema)

El núcleo del sistema está implementado en **C++** (estándar C++17) utilizando **Programación Orientada a Objetos (POO)**. Se hace uso de librerías estándar como *iostream*, *vector*, *fstream* y *string* para la gestión de datos y entrada/salida.Para el modo híbrido (consola y web), se implementa la **gestión de hilos y procesos** mediante llamadas al sistema (*system calls*), lo que permite ejecutar el binario de C++ desde un servicio externo.

ii. Frontend (Interfaz Web)

La capa de presentación se implementa mediante tecnologías web modernas:

1. **HTML5**: Proporciona la estructura semántica de las vistas principales, incluyendo Login y Dashboard.

2. **CSS3**: Permite un diseño responsivo, estilizado moderno y organización de elementos en tarjetas mediante Grid Layout.

3. **JavaScript (Vanilla ES6+)**: Maneja la lógica del cliente, manipulación del DOM y peticiones asíncronas al middleware utilizando Fetch API.

iii. Middleware

Se desarrolló un componente intermedio para comunicar la interfaz web con el núcleo en C++. Este middleware está implementado en **Python 3.x** y utiliza:

1. **FastAPI**: Framework que expone endpoints REST para recibir y responder peticiones HTTP.

2. **Uvicorn**: Servidor ASGI para desplegar la API de manera local.

3. **Subprocess**: Módulo de Python que permite la ejecución controlada del binario C++ desde la API.

iv. Persistencia

La información del sistema se almacena en **archivos de texto plano (.txt)**, funcionando como una base de datos ligera y portable. La estructura de los archivos está delimitada por caracteres especiales (por ejemplo, |) para separar campos.

v. Herramientas de Desarrollo

El proyecto utilizó las siguientes herramientas:

1. **IDE:** Visual Studio Code

2. **Compilador:** G++ (MinGW-w64)

3. **Control de Versiones:** Git, utilizado para la gestión del proyecto y control de cambios

b. Metodología de la implementación

El desarrollo del proyecto siguió un enfoque **iterativo e incremental**, diseñado para construir primero un núcleo robusto en C++ y posteriormente integrar una capa web moderna y amigable para el usuario. La metodología refleja el pensamiento del equipo de desarrollo, combinando diseño, modularidad y pruebas constantes.

1. **Diseño del Dominio de Clases**

   El proyecto comenzó con el diseño del dominio de clases, donde se estableció la estructura general del sistema y las carpetas que organizarían el código. Utilizando la herramienta Mermite, se modelaron las clases principales del negocio hotelero, incluyendo Usuario, Administrador, Cliente, Empleado y sus relaciones, definiendo cómo interactuaron entre sí.

   En esta fase también se definió la **modularización** del proyecto, decidiendo qué responsabilidades tendría cada clase y cómo se organizarían los archivos *.h* y *.cpp*. El objetivo fue garantizar un código limpio, fácilmente mantenible y escalable.

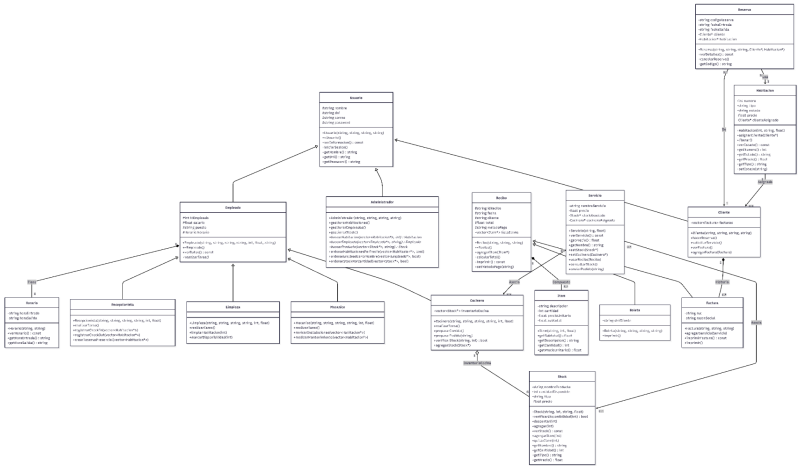
2. **Implementación de Clases**

   Una vez definido el dominio, se procedió a la implementación de los diagramas en código. Cada clase se tradujo a su respectivo archivo de cabecera (.h) y archivo fuente (.cpp), implementando atributos, métodos y la lógica necesaria para reflejar el comportamiento del negocio.

   Durante esta fase, se prioriza mantener la lógica de negocio en el núcleo C++, asegurando que todas las operaciones críticas del hotel, como reservas, facturación y control de habitaciones, fun

   Tras la implementación de la lógica, se abordó la **persistencia de datos**. Se incorporaron archivos de texto plano (*.txt*) para guardar información clave como estados de habitaciones, nómina de empleados e inventario.

   Algunas clases, como *Administrador*, fueron diseñadas para interactuar directamente con estos archivos, garantizando que las acciones diarias, como gestión de reservas o actualización de stock, pudieran ejecutarse de manera confiable y segura.



(https://drive.google.com/file/d/1f_UGR4-5ctAKFwzl5_qOO3btkOGCavvA/view?usp=sharing)

3. **Desarrollo de la Capa Híbrida**

   Con el núcleo funcionando, el equipo buscó modernizar el sistema sin comprometer la robustez de la programación orientada a objetos. Para ello, se refactoriza *main.cpp* para soportar **modo dual de ejecución**:

   Modo Interactivo: Menú clásico en la consola, permitiendo probar todas las funcionalidades directamente.

   Modo Headless: Ejecución silenciosa de comandos específicos, diseñada para ser consumida por software externo o una interfaz web.

   Esta decisión permitió mantener la lógica central en C++, evitando acoplamientos innecesarios con la capa visual.

4. **Integración Web y API Gateway**

   Finalmente, se implementó la **capa de presentación web** y el **API Gateway**. Utilizando **Python 3.x** y **FastAPI**, se construyó un servidor local que recibe peticiones HTTP desde el navegador y ejecuta comandos en el binario C++ del sistema.

   El frontend se desarrolló con **HTML, CSS y JavaScript**, creando un Dashboard intuitivo que permite gestionar reservas, monitorear habitaciones y realizar operaciones administrativas de manera visual y amigable. Esta integración ofrece “lo mejor de ambos mundos”: la seguridad y robustez de la programación orientada a objetos en C++ junto con la facilidad de uso de una interfaz moderna.

i. Visión

Nuestra visión consiste en poner en práctica la implantación del sistema efectivo y moderno de gestión hotelera para el Hotel Hermanos Mendoza. De tal forma, se tiene planeado dejar de usar archivos y registros en Excel, por un software concebido en C++ al que se accede mediante una ventana virtual a través de una interfaz web. A través de este sistema, pretendemos automatizar los procesos relacionados con la administración, gestionar mejor las habitaciones y a los trabajadores, disminuir el tiempo invertido en los diferentes registros y reducir errores, lo cual ayude a establecer una administración más ordenada y una mejor calidad en el servicio.

ii. Especificación de requerimientos de software

1. Requerimientos funcionales

RF 01 – Registrar los datos del huésped

El sistema permite registrar la información básica del huésped, específicamente su nombre completo y DNI, los cuales se utilizan como datos principales de identificación. De manera interna, el sistema genera automáticamente un correo electrónico y asigna una contraseña por defecto (“123”), permitiendo que el huésped pueda acceder posteriormente al sistema sin necesidad de un registro adicional.

RF 02 – Reservación de habitaciones online

Los clientes pueden realizar la reservación de habitaciones a través de la interfaz web, seleccionando únicamente habitaciones que se encuentren disponibles. El sistema valida dicha disponibilidad utilizando un puente de comunicación entre Python y C++, ejecutando el archivo hotel.exe para procesar la lógica de negocio.

RF 03 – Modificación de datos del huésped

El administrador del sistema tiene la capacidad de visualizar, buscar y gestionar los registros tanto de huéspedes como de empleados. Además, se permite la eliminación de empleados utilizando el DNI como identificador único.

RF 04 – Método de pago

El sistema genera automáticamente comprobantes de pago, ya sea Factura o Boleta, al momento de realizar una transacción. Estos comprobantes registran el monto total acumulado por los servicios utilizados o productos consumidos durante la estadía del huésped.

RF 05 – Inventario de snacks

El sistema cuenta con un módulo de inventario que gestiona los insumos del hotel, clasificados en las categorías Comida, Bebida y Limpieza. Para cada producto se registra su nombre, cantidad disponible y precio unitario.

RF 06 – Visualización de disponibilidad de habitaciones en pantalla del recepcionista

La interfaz permite visualizar en tiempo real la información de cada habitación, incluyendo su número, tipo (Simple, Doble, Suite o Matrimonial), precio y estado actual (Disponible u Ocupada).

RF 07 – Registro de check-in y check-out

El personal de recepción puede registrar el check-in y check-out de los huéspedes, lo que implica cambiar el estado de la habitación entre “Disponible” y “Ocupada”. Durante el proceso de check-out, el sistema calcula automáticamente los cargos acumulados del huésped.

RF 08 – Automatizar el periodo contable del hotel

El sistema garantiza que todas las transacciones, deudas y estados de cuenta sean almacenados de forma automática en archivos de texto plano, permitiendo su posterior revisión administrativa.

RF 09 – Integración de redes sociales en la interfaz para clientes

La página web del sistema incorpora una sección de contacto junto con enlaces a redes sociales, facilitando la comunicación entre el hotel y los clientes.

RF 10 – Búsqueda rápida de huésped

El sistema permite realizar búsquedas rápidas y eficientes de huéspedes o empleados utilizando el DNI como criterio único, optimizando la gestión de registros.

RF 11 – Envío de confirmación de reserva

El sistema deberá enviar automáticamente una notificación de confirmación de reserva (correo electrónico o WhatsApp) al huésped.

RF 12 – Registro de pago electrónico (Yape, Plin, etc.)

El sistema deberá registrar pagos realizados a través de plataformas electrónicas específicas (Yape, Plin o similares).

RF 13 – Consulta de balance pendiente

El sistema deberá mostrar el balance o monto pendiente de pago asociado a la cuenta de una habitación o huésped.

RF 14 – Consulta de asignación de limpieza móvil

El sistema deberá permitir al personal de limpieza visualizar su lista de asignaciones pendientes en una interfaz optimizada para dispositivos móviles.

RF 15 – Reporte de fin de limpieza

El personal de limpieza puede actualizar el estado de una habitación una vez finalizadas sus labores, informando al área de recepción que la habitación se encuentra lista para ser asignada nuevamente.

RF 16 – Reporte de objetos olvidados

El sistema deberá permitir al personal de limpieza registrar un objeto olvidado, asociándolo a la habitación y fecha de check-out.

RF 17 – Reporte de incidencia de mantenimiento

El sistema deberá permitir al personal de limpieza reportar un problema de mantenimiento en una habitación, detallando la avería.

RF 18 – Seguimiento de tareas de mantenimiento

El sistema permite que el personal de mantenimiento revise el estado de las instalaciones de las habitaciones y registre las acciones realizadas, asegurando que la información quede actualizada en el sistema.

RF 19 – Registro de salida de producto

Cuando un huésped solicita alimentos o bebidas, el sistema descuenta automáticamente la cantidad correspondiente del inventario y añade el costo a la cuenta del cliente.

RF 20 – Búsqueda y filtrado de inventario

El sistema deberá permitir la búsqueda y filtrado rápido de productos por nombre o categoría dentro del inventario.

RF 21 – Generación de reporte de ingresos diario detallado

El sistema deberá generar un reporte que detalle los ingresos del día, desglosados por origen (hospedaje, cafetería) y método de pago.

RF 22 – Visualización del estado de reserva en línea

El sistema deberá permitir a los clientes consultar el estado actual de su reserva (pendiente, confirmada, cancelada) desde la página web.

RF 23 – Asignación de habitación en recepción

El sistema deberá permitir al recepcionista asignar manualmente una habitación disponible al huésped durante la reserva presencial.

RF 24 – Emisión de factura física o digital

El sistema deberá permitir imprimir o enviar por correo la factura final al huésped al momento del check-out.

RF 25 – Consulta de cantidad de huéspedes hospedados

El sistema deberá permitir conocer en tiempo real la cantidad total de huéspedes que se encuentran hospedados en el hotel, con el fin de facilitar la planificación de los desayunos por parte del personal de cocina.

RF 26 – Visualización de cantidad de platos a preparar

El sistema deberá permitir al personal de cocina visualizar automáticamente la cantidad de platos que deben prepararse según el número de huéspedes registrados y el tipo de servicio (desayuno, almuerzo o cena).

RF 27 – Registro de menú diario

El sistema deberá permitir al encargado de cocina seleccionar y registrar desde su dispositivo móvil el menú diario que se servirá (por ejemplo, tipo de desayuno o almuerzo).

RF 28 – Iniciar sesión en el sistema

El sistema cuenta con un mecanismo de inicio de sesión por roles, diferenciando entre Administrador (DNI 9999), Empleados y Clientes, otorgando permisos específicos según el tipo de usuario.

RF 29 – Registro de inicio y fin de limpieza

Cada vez que se confirma una reserva o se finaliza una orden de limpieza, el sistema actualiza automáticamente el estado de la habitación en el archivo habitaciones.txt, evitando modificaciones manuales.

RF 30 – Visualización de desayunos en pantalla del recepcionista

El sistema deberá permitir al recepcionista visualizar en su panel principal la información relacionada con los desayunos programados para los huéspedes hospedados, mostrando el tipo de desayuno, cantidad de platos a preparar y estado de preparación

junto con la disponibilidad de habitaciones y el inventario de snacks.

2. **Requerimientos no funcionales**

RNF 01 - Copia de seguridad

El sistema utiliza un modelo de persistencia basado en archivos de texto (.txt) almacenados en la carpeta datos. Se recomienda realizar copias de seguridad manuales de esta carpeta para asegurar la protección de la información.

RNF 02 - Intuitivo en su uso

La interfaz web presenta un diseño moderno basado en Glassmorphism, utilizando la tipografía Inter y una disposición visual mediante tarjetas, lo que facilita una navegación clara e intuitiva para clientes y personal.

RNF 03 – Arquitectura de puente (Bridge)

La comunicación entre el frontend web y la lógica de negocio se realiza mediante una API REST desarrollada con FastAPI, la cual ejecuta comandos del sistema C++ de manera eficiente.

iii. Arquitectura de software

El sistema propuesto para el Hotel Los Hermanos Flores representa un cambio tecnológico importante. Se pasará de un manejo manual y propenso a errores con hojas de cálculo a una solución de software profesional más sólida. Desarrollado completamente en C++, el sistema usa una estructura modular que separa la lógica del negocio en archivos de encabezado (.h) y archivos de implementación (.cpp). Esto asegura un código limpio y fácil de mantener.

Aspectos Positivos del diagrama de clases:

Integridad y Persistencia de Datos: A diferencia de la utilización de hojas de cálculo, donde los datos pueden ser borrados sin querer, el presente

modelo usa el uso de archivos secuenciales y de acceso directo para asegurar que cada reserva, cliente o factura quede irreversiblemente registrada en los archivos de textos y pueda ser recuperado de forma segura una vez se cierra el programa.

Arquitectura Escalable (POO): El presente modelo presenta mucha flexibilidad gracias al paradigm Orientado a Objetos, y la herencia permite al hotel gestionar diferentes perfiles de usuario (Administrador, Recepcionista, Limpieza) sin duplicar código, ya que todos comparten la base de Usuario, pero cada uno hace funciones diferentes gracias al polimorfismo.

Eficiencia en la Búsqueda y Ordenación: El presente modelo permite mejorar el tiempo de respuesta administrativa al usar algoritmos de búsquedas y ordenación, por ejemplo, el administrador, puede filtrar habitaciones rápidamente por precio, la búsqueda de un cliente por su DNI evita la búsqueda visual de filas de Excel.

Automatización de Reportes y Facturación: Una de las ventajas del presente modelo es la automatización en la generación de reportes de gestión y facturas en formato TXT, esto también es Beneficioso desde el punto de vista del cliente, ya que el servicio queda más profesionalizado y se generan de forma automática un historial de auditoría para el negocio.

1. **Vista de datos**

2. **Clases**
    a. **Clase:Usuario**

        Representación y Propósito: Es la raíz de la jerarquía de personas. Existe para centralizar la seguridad y los datos de identidad (DNI, Nombre, Password), evitando la duplicidad de lógica de acceso en otras partes del sistema.

Encapsulamiento: Utiliza niveles **Protegidos (#)** para los datos de identidad. Esto se decidió para que las clases hijas (Empleado, Cliente) heredan directamente los atributos sin perder la protección frente a agentes externos (Seguridad de datos).

Atributos y Métodos: Contiene variables de texto para credenciales y métodos de validación como iniciarSesion. Son fundamentales para garantizar que solo personal autorizado manipule el software.

b. Clase:Cliente

Representación y Propósito: Modela al beneficiario de los servicios. Su existencia permite asociar consumos reales y reservas a una persona específica para la emisión de comprobantes legales.

Encapsulamiento: Posee un **vector privado (-)** de facturas. Esto garantiza que el historial de pagos solo pueda ser modificado por métodos autorizados del hotel, protegiendo la integridad financiera del cliente.

Atributos y Métodos: Maneja un historial de documentos. Sus métodos permiten "pedir" servicios, lo cual dispara la creación de ítems en su cuenta personal.

c. Clase:Empleado (Clase Abstracta)

Representación y Propósito: Define el estándar laboral del hotel. Existe para agrupar sueldo, ID y horario bajo un solo tipo de dato, permitiendo que el Administrador gestione a todo el personal de forma masiva.

Encapsulamiento: Sus atributos son **Protegidos (#)** para que las especialidades (Cocinero, Mecánico) puedan leer su propio salario o puesto sin intermediarios, facilitando la lógica interna de cada rol.

Atributos y Métodos: Destaca el método virtual puro realizarTarea(). Es la pieza clave del polimorfismo: define "qué" debe hacer un empleado, pero deja que cada subclase decida "cómo" hacerlo.

d. Clase:Administrador

Representación y Propósito: Es el controlador lógico del sistema. Existe para orquestar la base de datos y asegurar que los recursos (habitaciones y personal) estén siempre actualizados.

Encapsulamiento: Sus operaciones son **Públicas (+)** ya que actúa como la interfaz principal entre el usuario de alto nivel y el motor del programa.

Atributos y Métodos: Sus métodos de **búsqueda y ordenación** son vitales; permiten encontrar una habitación o empleado específico en milisegundos dentro de vectores con cientos de registros.

e. Clase:Recepcionista

Representación y Propósito: Modela el contacto operativo. Existe para sincronizar la llegada del cliente con la disponibilidad física de las habitaciones en el sistema.

Encapsulamiento: No posee atributos propios, pues utiliza los heredados. Se enfoca en la visibilidad de métodos de "escritura" sobre el vector de habitaciones.

Atributos y Métodos: Sus métodos de **Check-In y Check-Out** son los únicos autorizados para cambiar el estado de una habitación a "Ocupada", manteniendo el orden logístico.

f. Clase:Limpieza

Representación y Propósito: Gestiona la higiene de la infraestructura. Existe para retroalimentar al sistema una vez que una habitación está apta para un nuevo huésped.

Encapsulamiento: Métodos públicos de actualización. Su importancia radica en la comunicación con el objeto Habitación.

Atributos y Métodos: El método marcarDisponibilidad es esencial para el flujo de ingresos; sin este proceso, las habitaciones quedarían bloqueadas permanentemente como "Sucias".

g. **Clase:Cocinero**

Representación y Propósito: Encargado del área de producción. Existe para transformar insumos tangibles en servicios consumibles, conectando el inventario con el cliente.

Encapsulamiento: Mantiene una **Agregación Privada (-)** con el Stock. Esto evita que el cocinero altere el inventario general sin un proceso de pedido formal.

Atributos y Métodos: El método verificarStock asegura que no se vendan platos que no pueden ser preparados, evitando quejas del cliente.

h. **Clase:Mecánico**

Representación y Propósito: Soporte técnico preventivo. Existe para gestionar las averías y asegurar que los activos del hotel (habitaciones) no dejen de producir dinero.

Encapsulamiento: Interacción directa con punteros de habitación para modificar estados de mantenimiento.

Atributos y Métodos: Operaciones de revisión de instalaciones que cambian el estado de la habitación a "En Reparación", bloqueando su venta temporalmente.

i. **Clase:Habitación**

Representación y Propósito: Es la unidad de negocio central. Existe para modelar el alojamiento y es el objeto más consultado por todas las demás clases.

Encapsulamiento: Todos sus atributos son **Privados (-)**. Es la aplicación más estricta de la POO en el proyecto; nadie puede cambiar un precio o un estado si no es a través de métodos seguros (Setters).

Atributos y Métodos: Posee número, tipo y precio. Sus métodos de asignación de cliente vinculan la memoria de la habitación con la del huésped de forma dinámica.

j. **Clase:Reserva**

Representación y Propósito: Contrato temporal de uso. Existe para planificar la demanda futura y evitar el colapso operativo por falta de espacio.

Encapsulamiento: Atributos privados que protegen las fechas y los códigos de reserva, evitando alteraciones accidentales en el calendario.

Atributos y Métodos: Métodos de validación y cancelación que liberan o bloquean punteros de habitación y cliente.

k. **Clase:Horario**

Representación y Propósito: Parametrización del tiempo. Existe como un componente de composición dentro de Empleado para estandarizar los turnos.

Encapsulamiento: Atributos privados para entrada y salida. Se decidió así para que el horario sea un objeto inmutable una vez asignado al contrato del empleado.

Atributos y Métodos: Métodos de visualización de jornada. Su importancia es meramente organizativa pero crítica para la gestión de recursos humanos.

l. **Clase:Stock**

Representación y Propósito: Gestión de inventario físico. Existe para cuantificar los recursos disponibles y calcular el valor de los insumos del hotel.

Encapsulamiento: Niveles privados. La cantidad disponible es el atributo más sensible, ya que de él dependen los módulos de Cocina y Servicio.

Atributos y Métodos: Métodos de descuento y agregación. Su importancia radica en que evita pérdidas económicas por falta de control en los suministros.

m. **Clase:Recibo**

Representación y Propósito: Base documental contable. Existe para unificar la lógica de cobro y el manejo de múltiples ítems de consumo bajo una sola estructura de "Total".

Encapsulamiento: Atributos **Protegidos** (#) para facilitar la herencia hacia Boleta y Factura. El vector de ítems es privado para asegurar que no se borren consumos sin dejar rastro.

Atributos y Métodos: Destaca el destructor virtual que limpia la memoria de los ítems. Sus métodos calculan sumatorias de subtotales de forma recursiva.

n. **Clase:Boleta**

Representación y Propósito: Documento legal para personas naturales. Existe para registrar ventas directas asociadas a un DNI según la normativa local.

Encapsulamiento: Hereda la base de Recibo y añade atributos privados para la identidad específica del cliente natural.

Atributos y Métodos: Sobrescribe el método imprimir() para añadir el formato visual requerido por una boleta de venta simple.

o. **Clase:Factura**

Representación y Propósito: Documento legal corporativo. Existe para gestionar transacciones empresariales que requieren RUC y Razón Social.

Encapsulamiento: Posee atributos privados específicos para los datos fiscales, aislándolos de la lógica general del recibo.

Atributos y Métodos: Incluye métodos para agregar servicios y un método de impresión especializado en reportes tributarios.

p. **Clase:Ítem**

Representación y Propósito: Unidad de detalle de cobro. Existe para desglosar qué está pagando el cliente, permitiendo auditorías de consumo claras.

Encapsulamiento: Atributos privados. El precio unitario y la cantidad son inalterables una vez que el ítem entra en el vector del recibo.

Atributos y Métodos: El método getSubtotal es la operación atómica más importante del módulo financiero; sin él, no habría cálculo de totales confiable.

q. **Clase:Servicio**

Representación y Propósito: Orquestador de consumos adicionales. Existe como puente lógico para que un cliente pueda generar un gasto (como Room Service) que afecte al Stock y al Recibo simultáneamente.

Encapsulamiento: No almacena datos persistentes, sino que funciona mediante asociaciones con punteros de otras clases (Stock, Cocinero).

Atributos y Métodos: Sus métodos de "enviar pedido" disparan la lógica de negocio entre departamentos, conectando la operación con la facturación.

3. **Módulos (Organización del Proyecto)**

El proyecto se encuentra organizado en una estructura de carpetas jerárquica que separa la ejecución principal, la definición de objetos, el almacenamiento de información y la interfaz gráfica. A continuación, se describen los módulos principales del sistema.

a. **Módulo Principal (Raíz)**

Este módulo contiene el punto de entrada de la aplicación y el motor de ejecución.

Archivo: main.cpp

Función:

El módulo principal orquesta todo el sistema. Es responsable de inicializar los objetos, comprobar los argumentos de inicio (para determinar si el sistema corre en modo consola o web) y mantener el bucle principal de interacción con el usuario.

b. **Módulo de Modelo de Dominio (/model)**

Este directorio constituye el núcleo del sistema y almacena todas las definiciones de clases bajo el paradigma de Programación Orientada a Objetos. En lugar de fragmentar el código en múltiples subcarpetas, se ha optado por una organización centralizada en la carpeta /model, facilitando el acceso a los encabezados y la gestión de dependencias entre clases.

Estructura de Archivos del Módulo

El módulo se compone de archivos de cabecera (.h) para las declaraciones y archivos de implementación (.cpp) para la lógica. A continuación, se listan los archivos presentes:

Definiciones de Cabecera (Archivos .h): Contienen la estructura de las clases y el encapsulamiento (atributos privados y protegidos).

- Usuario.h, Cliente.h, Empleado.h, Administrador.h, Recepcionista.h, Cocinero.h, Limpieza.h, Mecanico.h.
- Habitacion.h, Reserva.h.
- Stock.h, Factura.h, Boleta.h, Recibo.h, Item.h.

Implementaciones Lógicas (Archivos .cpp): Contienen el desarrollo de los métodos y la lógica operativa de cada clase definida en los headers.

- Usuario.cpp, Cliente.cpp, Empleado.cpp, Administrador.cpp, Recepcionista.cpp, Cocinero.cpp, Limpieza.cpp, Mecanico.cpp.
- Habitacion.cpp, Reserva.cpp.
- Stock.cpp, Factura.cpp, Boleta.cpp, Recibo.cpp, Item.cpp.

Justificación de la Organización

Se ha decidido mantener todas las clases en un mismo nivel jerárquico dentro de /model por las siguientes razones técnicas:

1. **Visibilidad Directa:** Permite que las clases que heredan de otras (como Administrador de Usuario) encuentren sus dependencias de forma inmediata sin rutas complejas de inclusión.
2. **Simplicidad de Compilación:** Facilita la configuración del compilador al tener una única ruta de búsqueda para los archivos de cabecera.
3. **Cohesión:** Todas las clases pertenecen al mismo dominio de "Gestión Hotelera", por lo que mantenerlas juntas refuerza la unidad del sistema.

c. Módulo de Persistencia de Datos (/datos)

Este directorio está dedicado exclusivamente al almacenamiento de la información del sistema. Funciona como una base de datos basada en archivos de texto plano.
Contenido: Archivos .txt

Archivos Clave:

- habitaciones.txt: Guarda el estado (ocupada/disponible) y precios de las habitaciones.
- empleados.txt: Registro de personal y credenciales.
- stock.txt: Inventario de productos.
- clientes.txt: Base de datos de clientes registrados.

d. Módulo de Interfaz Web (Front-End & API)

Este módulo contiene los archivos necesarios para la ejecución del sistema en modo gráfico mediante un navegador web.

Carpeta /web:

- **index.html:** Estructura visual de monitoreo y login.

- **style.css:** Hojas de estilo para la presentación gráfica.
- **script.js**: Lógica del cliente para peticiones asíncronas.

Carpeta /api:

- **main.py:** Script puente que conecta el navegador con el ejecutable C++, permitiendo la comunicación entre ambos componentes del sistema.

7. Conclusiones y recomendaciones

El proyecto logró desarrollar un sistema de gestión hotelera que optimiza los procesos administrativos del Hotel Los Hermanos Flores. Se recomienda:
Implementar una interfaz gráfica en futuras versiones.
Integrar una base de datos para mejorar la persistencia de datos.

8. Bibliografía

CodersLink. (s. f.). *¿Qué es la programación orientada a objetos (POO) y cuáles son sus principios fundamentales?*
https://coderslink.com/talento/blog/que-es-la-programacion-orientada-a-objetos-poo-y-cuales-son-sus-principios-fundamentales/
Concepto.de. (s. f.). *Algoritmo en informática*.
https://concepto.de/algoritmo-en-informatica/
DataCamp. (s. f.). *What is an algorithm?*
https://www.datacamp.com/es/blog/what-is-an-algorithm
CEV. (s. f.). *Los 4 pilares de la programación orientada a objetos*.
https://www.cev.com/blog/4-pilares-de-la-programacion-orientada-a-objetos
W3Schools. (s. f.). *Archivos en C++*.
https://w3schoolsua.github.io/cpp/cpp_files_es.html

9. Anexos

Anexo A: Repositorio de Código Fuente
Para facilitar la revisión integral del proyecto, se ha dispuesto de un repositorio público en GitHub que contiene la arquitectura completa del sistema de gestión para el Hotel “Hermanos Flores”.
Enlace de acceso: *Repositorio - Proyecto PA*