Trabajo Práctico Final: Laboratorio 3: “Sistema de administración de un hotel”

Integrantes: Grupo 7

- Aristegui, Federico
- Artaza, Nehuen
- Gonzalez, Valentín

Comisión: 3

Profesor: Guillermo Giménez

Materia: Laboratorio 3

Año: 2024

Mar del Plata, Argentina

INDICE

1. Resumen de aplicación y proceso de desarrollo

El trabajo consiste en el desarrollo de un sistema para la administración de un hotel. Permite manejar al personal, habitaciones, reservas, pasajeros, entre otras cosas, además de guardar un historial de todas las estadías de los clientes que han transitado por el hotel. El proyecto se basa en cubrir todas las necesidades básicas que necesitaría manejar un hotel, permitiendo además la implementación de nuevas funciones a futuro, acorde a las necesidades específicas del cliente.

Durante los inicios del desarrollo, el enfoque principal del proyecto consistió en la confección de un diagrama UML con el objetivo de plantear todas las estructuras del sistema, junto a sus variables, métodos, dependencias y relaciones mutuas. Otro enfoque durante el desarrollo del UML y posterior código, fue el objetivo de implementar todos los conceptos y herramientas aprendidas durante la cursada de Programación/Laboratorio 3. Una vez finalizado el proceso de creación del UML, iniciamos con la creación del código utilizando lenguaje Java. Con el progreso del código, y al presentarse situaciones que ameritan la toma de decisiones, fue necesario cambiar ciertas estructuras que habían sido planteadas en el concepto original, y con eso se modificó el UML acorde a los cambios realizados.

Una vez finalizados todos los bloques de código, comenzó la creación del menú, junto a todas las validaciones necesarias para cada caso, además de las ya implementadas en cada método. Luego, inició una etapa de prueba de todas las funcionalidades, sus métodos, el correcto guardado de los datos en sus archivos correspondientes, así como la corrección de errores. Este último punto se llevó a cabo recorriendo el programa como un usuario del mismo.

2. Informe técnico de la aplicación

Se crearon distintos paquetes para encapsular las diferentes clases e interfaces con sus respectivos atributos y métodos, a modo de organizar mejor el código.

2.1 “Services”, el cual cuenta con:

2.1.1 Habitación: se diferencian por número. Tienen su estado para administrar las reservas y su ocupación. La capacidad determina la cantidad máxima de personas que pueden ocuparla. Cuenta es un acumulador de los gastos extra realizados a nombre de dicha habitación (compras, consumos, etc.), monto que será abonado por el pasajero al momento del check-out.

2.1.2 Gestor Habitación: esta clase contiene un ArrayList, almacenando la información de cada instancia de la clase Habitacion en el sistema. Además posee los métodos para la administración de las habitaciones del sistema y devolución de atributos asociados a dicha clase.

2.1.3 Historial: se utiliza para llevar el historial de cada período individual que un pasajero ocupó satisfactoriamente una habitación en el hotel. Contiene fechas de ingreso y egreso. Los datos de la Habitación ocupada.

2.1.4 Reserva: esta clase guarda todos los datos necesarios y validados a la hora de poder efectuar una reserva. Contiene las fechas de ingreso y egreso. La cantidad de personas que ocupan la habitación durante la estadía. El dni del pasajero que realizó la reserva. El estado actual de una reserva (determinado por un enum “ReservaStatus”). El numeroHabitacion vinculada a la reserva en cuestión.

2.1.5 Gestor Reserva: esta clase contiene un ArrayList, almacenando la información de cada instancia de la clase Reserva en el sistema (limitado a reservas activas, o que la habitación se encuentre activamente ocupada). Contiene los métodos necesarios para poder realizar o cancelar una reserva, consultar estados de reservas por número de habitación o dni del pasajero vinculado, además de los métodos para mostrar estados de reservas actuales.

2.1.6 Producto: se utiliza para representar los productos/consumos que puede realizar un pasajero mientras se encuentre ocupando una habitación activamente. Dichas compras son acumuladas y abonadas al momento de retirarse del hotel durante el check-out. Se distinguen nombre y precio que determinan las características esenciales de cada producto disponible.

2.1.7 Gestor Staff: esta clase contiene un ArrayList, almacenando los datos de instancias de la clase Persona que se envíen, especificando su rol como staff dentro del hotel. Se utiliza para mostrar los datos del staff actual.

2.1.8 Hotel: es la clase principal en la gestión del programa. Contiene tres ArrayList que almacenan la información de cada instancia de las clases GestorStaff, Pasajero, y Producto en el sistema. También tiene como atributos a las clases GestorReserva y GestorHabitacion, datos que recibirá y serán necesarios para poder llevar a cabo todas las funciones y métodos de administración que le corresponden. Dentro de sus principales métodos se encuentran operaciones como registrar y eliminar habitaciones, cambiar estados de habitaciones, gestionar staff (incluyendo cambios de rango de recepcionista a administrador y viceversa), hacer/cancelar reservas, gestionar pasajeros, productos, etc. Además, proporciona métodos para guardar y cargar datos del hotel desde archivos JSON para una correcta creación de backups y traslado de información.

2.2 "Users", el cual cuenta con las **interfaces**:

2.2.1 Recepcionable: interfaz que determina los métodos y funciones que solo tendrá acceso una Persona logueada como Recepcionista. Se desarrolla la funcionalidad que luego implementan dichas clases.

checkin: realiza el proceso de check-in para un pasajero en una habitación específica, utilizando los detalles de la reserva.

checkout: realiza el proceso de check-out para un pasajero que está dejando una habitación, utilizando los detalles de la reserva.

cancelarReserva: cancela una reserva existente en el hotel.

cambiarEstadoHabitacion: permite cambiar el estado de una habitación específica del hotel a un nuevo estado.

2.2.2 Administrable: esta interfaz implementa la interfaz Recepcionable, y además determina los métodos y funciones que solo tendrá acceso una Persona logueada como Administrador. Se desarrolla la funcionalidad que luego implementan dichas clases.

cambiarStaff: permite cambiar la información de un miembro del personal y actualiza la lista de personal del hotel con la nueva información.

borrarStaff: permite eliminar un miembro de la lista de personal del hotel mediante su dni.

mostrarStaff: muestra la lista de personal del hotel.

Además, contiene las siguientes **clases:**

2.2.3 Persona: es la clase padre que posee los atributos que definen a la identidad de cada persona, su herencia focaliza en distinguir los datos y métodos necesarios para cada clase particular. Persona contiene los atributos nombre, apellido y dni.

2.2.4 Pasajero: hereda la clase Persona. Además, declara atributos localidad, domicilio y cuenta con un ArrayList historial que almacena datos de tipo Historial con toda la información de cada período individual que un pasajero ocupó satisfactoriamente una habitación en el hotel.

2.2.5 Recepcionista: hereda la clase Persona, implementa la interfaz Recepcionable. Implementa los métodos de dicha interfaz, con los nombres y funciones desarrollados en 2.2.1

2.2.6 Administrador: hereda la clase Persona, implementa la interfaz Administrable. Implementa los métodos de dicha interfaz, con los nombres y funciones desarrollados en 2.2.2

2.3 "Utilities", que cuenta con 3 enum:

2.3.1 HabitaciónStatus: contiene: EN_LIMPIEZA, EN_REPARACION, EN_DESINFECCION, DISPONIBLE y OCUPADO. A la hora de realizar una reserva o check-in, se valida que la habitación se encuentre en estado disponible para que los pasajeros puedan ingresar correctamente.

2.3.2 Mes: contiene los doce meses del año con los días de cada uno. El sistema las utiliza para administrar las fechas a la hora de manejar reservas.

2.3.3 ReservaStatus: contiene: ACTIVA, EN_PROCESO, FINALIZADA y CANCELADA. Se utiliza para visualizar el estado de las reservas a la hora de imprimir dicha información.

3. Bibliografía

- A la hora de decidir el tema del proyecto y durante la toma de decisiones en la confección del diagrama UML, del código, menú, etc. estuvieron basadas principalmente por experiencias y conocimientos propios trabajando en hotelería en la ciudad de Mar del Plata.
- Teorías y ejemplos proporcionados en el campus de la cátedra.