Desarrollo de un sistema de gestión hotelera *Sumas*

Jhosep Rodrigo Arocutipa Mamani[1], Alexander Bernabe [2] Flores Cutipa [2,*]

1. Introducción:

Actualmente, la industria hotelera enfrenta desafíos como la gestión eficiente de reservas, la atención personalizada al cliente y la optimización de recursos. La implementación de un sistema de gestión hotelera basado en tecnología avanzada permite automatizar procesos, mejorar la experiencia del cliente y gestionar información de manera más eficiente.

Este proyecto se propone Implementar diagrama de clases usando la programación orientada a objetos, viendo su eficiencia y eficacia en el desarrollo de sistema de gestión hotelera. Mediante el uso del diagrama de clases, se busca modelar de manera estructurada las entidades principales del sistema, tales como clientes, habitaciones, reservas, pagos y empleados, así como las relaciones existentes entre ellas. La correcta aplicación de los principios de la programación orientada a objetos —encapsulamiento, herencia, polimorfismo y abstracción— permitirá obtener un diseño modular, escalable y fácil de mantener. Asimismo, se evaluará cómo este enfoque contribuye a la reducción de errores, mejora la organización del código y optimiza el proceso de desarrollo del software, garantizando una solución eficiente y funcional para la gestión hotelera.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

Citation: To be added by editorial staff during production.

Copyright: © 2025 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https//creativecommons.org/licenses/by/4.0/).

2. Titulos

Sistema de Gestion hotelera ***Sumas***

2. Autores:

Jhosep Rodrigo Arocutipa Mamani
Alexander Bernabe Flores Cutipa

4. Planteamiento del problema:

Uno de los problemas más frecuentes en la gestión hotelera es la falta de centralización de la información. Esto puede llevar a errores en la asignación de habitaciones, reservas duplicadas y mala atención al cliente, lo cual afecta la rentabilidad y la experiencia del usuario.

5. Objetivos:

5.1. Objetivos generales:

Diseñar e implementar un sistema de gestión hotelera utilizando un diagrama de clases para optimizar la gestión de reservas, la asignación de habitaciones y la atención al cliente.

5.2. Objetivos específicos:

- Entender como funciona los sistenas de gestión para su correcto analisis y desarrollar un software..
- Crear un diagrama de clases que modele los componentes del sistema de gestión hotelera.
- Implementar funcionalidades clave, como registro de clientes, gestión de reservas, asignación de habitaciones, entre otros.
- Evaluar el rendimiento y la eficiencia del sistema desarrollado en un entorno de prueba.

6. Marco Teorico:

6.1.Programacion orientada a objetos(POO):

La programación orientada a objetos (POO) es un paradigma que organiza el software en objetos, que son instancias de clases. En el contexto de un sistema de gestión hotelera, el diagrama de clases es una herramienta visual fundamental para representar los elementos del sistema (como clientes, habitaciones, reservas) y sus relaciones, lo que facilita su implementación.

6.1.2 Principios fundamentales del POO:

a) Encapsulamiento: Encapsulación se refiere a la implementación de métodos dentro de una estructura de un programa, haciendo invisible con los métodos toda la información de mis atributos. El objetivo de la encapsulación es la de garantizar la integridad de mis datos por los medios de acceso a los atributos o métodos, los cuales pueden ser públicos, privados o protegidos. Con esta característica de la programación orientada a objeto, los usuarios de una clase desconocen la implementación de mi código.

b) Abstraccion: La abstracción consiste en reducir las dificultades y complejidades del mundo real realizando un modelado solo de lo más importante de cada objeto, no tomando en cuenta detalles innecesarios permitiendo a los desarrolladores de programas dar importancia solo a la parte de mayor interés. La abstracción se dice es la mar importante

de los pilares de la POO ya que de aquí parte un buen análisis y diseño orientado a objetos [1]

c) Herencia: La herencia en el paradigma orientada a objeto se refiere al mecanismo por el cual una clase o plantilla hereda los atributos y características de otra clase. Utilizando esta característica podemos decir que nuestros códigos de programación serán más rápidos y eficiente, además nos permite la reutilización de código, logrando así código mucho más limpio y entendible. Es importante mencionar que existe tipos de herencia manejables en los lenguajes de programación de alto nivel: Herencia simple, múltiple, jerárquica e hibrida. Con el manejo de herencia tenemos una gama de posibilidades y maneras de desarrollar eficientemente y estructurada nuestros programas [2].

d) Poliomorfismo: De manera literal podemos definir el polimorfismo como la virtud de un objeto para adquirir múltiples formas o comportamientos. En términos de la programación se refiere a un mismo nombre a varios métodos, pero con distintos comportamientos. Cuando se utiliza esta característica en el paradigma de la programación orientada a objeto se tiene un sinfín de ventajas, por ejemplo: Permiten que los objetos se comporten de manera diferente, ayuda a ejemplificar la lógica del programa, expandir la funcionalidad del programa, etc.

6.2 UML(lenguaje unificado de Modelado):

El lenguaje de modelado UML es el estándar más utilizado para especificar y documentar cualquier sistema de forma precisa. Sin embargo, el hecho de que UML sea una notación de propósito muy general obliga a que muchas veces sea deseable poder contar con algún lenguaje más específico para modelar y representar los conceptos de ciertos dominios particulares. Los Perfiles UML constituyen el mecanismo que proporciona el propio UML para extender su sintaxis y su semántica para expresar los conceptos específicos de un determinado dominio de aplicación. En este artículo analizaremos los mecanismos de extensión que se utilizan para definir un Perfil UML. También discutiremos la utilidad y relevancia de estos Perfiles UML en el contexto del modelado guiado por arquitecturas (MDA).

6.2.1. Arquitectura de metamodelos definida por OMG

En general, podemos pensar que un modelo describe los elementos que pueden existir en un sistema, así como sus tipos. Por ejemplo, si definimos la clase "Persona" en un modelo, podremos utilizar instancias de dichas clases en nuestro sistema (por ejemplo, "Luis"). De igual forma, si el sistema que queremos modelar es un sistema UML, entonces los elementos que componen dicho sistema serán "Class", "Association", Package", etc. ¿Quién define esos elementos, y cómo se definen? OMG define una arquitectura basada en cuatro niveles de abstracción que van a permitir distinguir entre los distintos niveles conceptuales que intervienen en el modelado de un sistema. Esos niveles se les denomina comúnmente con las iniciales M0, M1, M2 y M3.[3]

– El nivel M0 – Las instancias. El nivel M0 modela el sistema real, y sus elementos son las instancias que componen dicho sistema. Ejemplos de dichos elementos son el

cliente "Juan" que vive en "Paseo de la Castellana, Madrid" y ha comprado el ejemplar número "123" del libro "Ulises". [3]

– El nivel M1 – El modelo del sistema. Los elementos del nivel M1 son los modelos de los sistemas concretos. En el nivel M1 es donde se definen, por ejemplo, los conceptos de "Cliente", "Compra" y "Libro", cada uno con sus correspondientes atributos ("dirección", "numero de ejemplar", "título", etc.). Existe una relación muy estrecha entre los niveles M0 y M1: los conceptos del nivel M1 definen las clasificaciones de los elementos del nivel M0, mientras que los elementos del nivel M0 son las instancias de los elementos del nivel M1. [3]

– El nivel M2 – El modelo del modelo (el metamodelo). Los elementos del nivel M2 son los lenguajes de modelado. El nivel M2 define los elementos que intervienen a la hora de definir un modelo del nivel M1. En el caso de un modelo UML de un sistema, los conceptos propios del nivel M2 son "Clase", "Atributo", o "Asociación". Al igual que pasaba entre los niveles M0 y M1, aquí también existe una gran relación entre los conceptos de los niveles M1 y M2: los elementos del nivel superior definen las clases de elementos válidos en un determinado modelo de nivel M1, mientras que los elementos del nivel M1 pueden ser considerados como instancias de los elementos del nivel M2. [3]

– El nivel M3 – El modelo de M2 (el meta-metamodelo). Finalmente, el nivel M3 define los elementos que constituyen los distintos lenguajes de modelado. De esta forma, el concepto de "clase" definido en UML (que pertenece al nivel M2) puede verse como una instancia del correspondiente elemento del nivel M3, en donde se define de forma precisa ese concepto, así como sus características y las relaciones con otros elementos (p.e., una clase es un clasificador, y por tanto puede tener asociado un comportamiento, y además dispone de un conjunto de atributos y de operaciones). [3]

Resumiendo, la semántica de UML viene descrita por su meta-modelo, que va a venir expresado en MOF. Si quisiéramos definir un lenguaje diferente a UML, también lo expresaríamos en MOF.[3]

6.3. Diagrama de clases:

Diagrama de clases Un diagrama de clases representa en un esquema gráfico, las clases u objetos intervinientes y como se relacionan en su escenario, sistema o entorno. Con estos diagramas, se logra diseñar el sistema a ser desarrollado en un lenguaje de programación, generalmente orientado a objetos. Estos diagramas los incorporan algunos entornos de desarrollo, tal es el caso de Eclipse con el plugin Papyrus o Netbeans con su respectivo plugin UML. Es un buen hábito generar proyectos UML con sus respectivos diagramas de clases para luego automáticamente obtener código fuente que nos colabore en el desarrollo del sistema o software.

Previo al desarrollo, en etapas de análisis y diseño de sistemas los diagramas de clases juegan un papel muy importante ya que permiten visualizar a partir de las clases y sus vínculos, como los objetos interactúan en el entorno propuesto. Una clase va a representar a los objetos que se produzcan a partir de haberla instanciado, indicando claramente las propiedades y métodos que poseen. Si la clase es abstracta no podrá ser instanciada sino a partir de sus clases derivadas. Una relación representa el detalle del vínculo entre dos clases, destacando el tipo (cual es la relación), la aridad o multiplicidad (cantidad de objetos de una y otra clase) y la navegabilidad (que objeto puede observar a otro). Ante un diseño orientado a objetos, es importante conocer la

diversidad de relaciones que se pueden producir, necesitar o establecer entre clases. - Tipos de relaciones Aprovecharemos la descripción de las relaciones para orientar a nuestros estudiantes hacia el código Java que involucran. Para esto describiremos en cada tipo de relación.

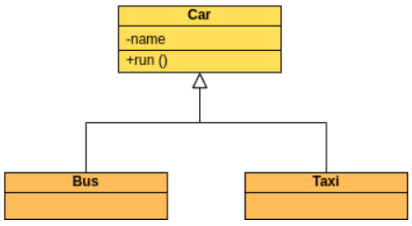
6.4.Relaciones entre clases:

- **Herencia:**

  La herencia también se denomina **generalización** y se utiliza para describir la relación entre las clases padre e hijo. Una clase principal también se denomina clase base y una subclase también se denomina clase derivada.

En la relación de herencia, la subclase hereda todas las funciones de la clase principal y la clase principal tiene todos los atributos, métodos y subclases. Las subclases contienen información adicional además de la misma información que la clase principal.

Por ejemplo: autobuses, taxis y automóviles son automóviles, todos tienen nombres y todos pueden estar en la carretera.

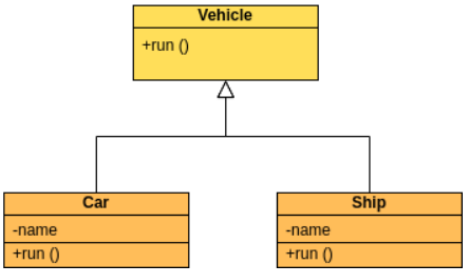


- **Realización / Implementación**

La implementación (Implementación) se utiliza principalmente para especificar **la relación entre las interfaces y las clases de implementación** .

Una interfaz (incluida una **clase abstracta**) es una colección de métodos. En una relación de implementación, una clase implementa una interfaz y los métodos de la clase implementan todos los métodos de la declaración de la interfaz.

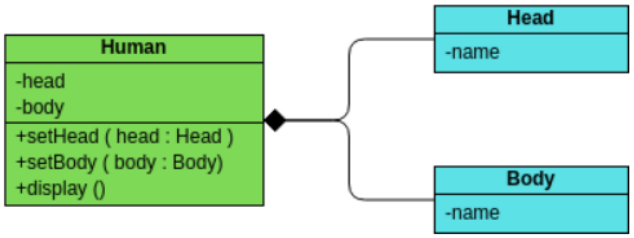
Por ejemplo: los automóviles y los barcos son vehículos, y el vehículo es solo un concepto abstracto de una herramienta móvil, y el barco y el vehículo realizan las funciones móviles específicas.



- **Relación de composición**

Composición: **La relación entre el todo y la parte, pero el todo y la parte no se pueden separar** .

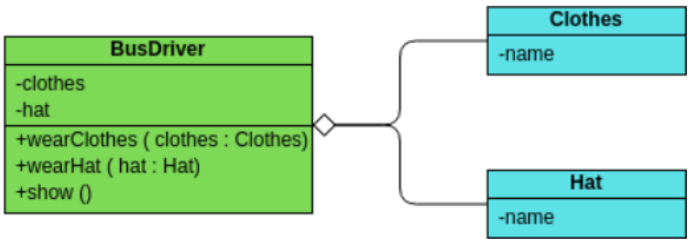
La relación de combinación representa la relación entre el todo y la parte de la clase, y el total y la parte tienen una duración constante. Una vez que el objeto general no existe, algunos de los objetos no existirán y todos morirán en la misma vida. Por ejemplo, una persona está compuesta por una cabeza y un cuerpo. Los dos son inseparables y coexisten.



- **Relación de agregación**

Agregación: **La relación entre el todo y la parte, y el todo y la parte se pueden separar.** Las relaciones agregadas también representan la relación entre el todo y una parte de la clase, los objetos miembros son parte del objeto general, pero el objeto miembro puede existir independientemente del objeto general.

Por ejemplo, los conductores de autobús y la ropa y los sombreros de trabajo son parte de la relación general, pero se pueden separar. La ropa de trabajo y los sombreros se pueden usar en otros conductores. Los conductores de autobuses también pueden usar otra ropa de trabajo y sombreros.



- **Relaciones de asociación**

Asociación: indica que **una propiedad de una clase contiene una referencia a una instancia (o instancias) de otra clase** .

La asociación es la relación **más utilizada** entre una clase y otra clase, lo que significa que existe una conexión entre un tipo de objeto y otro tipo de objeto. **Las combinaciones y agregaciones también pertenecen a las relaciones asociativas** , pero las relaciones entre clases de afiliaciones son más débiles que las otras dos.

Hay cuatro tipos de **asociaciones** : **asociaciones bidireccionales** , **asociaciones unidireccionales** , **autoasociación y asociaciones** de **números múltiples** .

Por ejemplo: coches y conductores, un coche corresponde a un conductor en particular y un conductor puede conducir varios coches.

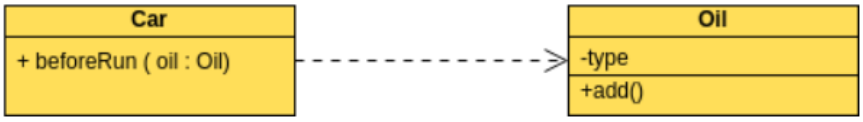
En los diagramas UML, las asociaciones bidireccionales pueden tener **dos flechas** o **ninguna** , y las asociaciones unidireccionales o autoasociaciones tienen **una flecha** .

En una relación de multiplicidad, puede agregar un número directamente a la línea asociada para indicar el número de objetos en la clase correspondiente.

- 1..1: Sólo uno
- 0..*: cero o más
- 1..*:uno o mas
- 0..1: Ninguno o solo uno
- m..n: al menos m, como máximo n (m<=n)

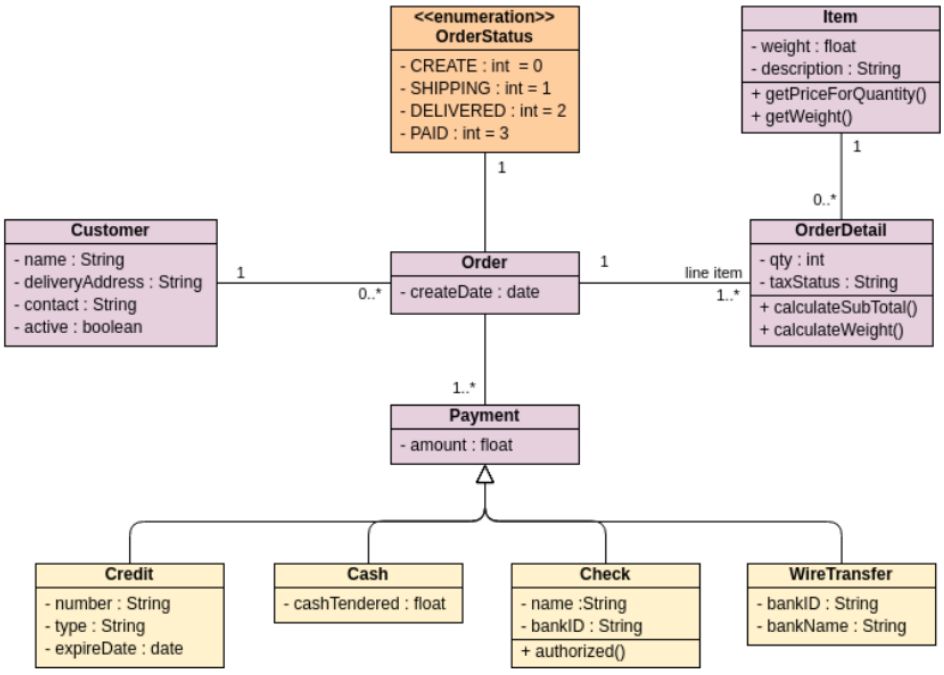
- **dependencias**

Dependencia: suponga que un cambio en la clase A provoca un cambio en la clase B, luego diga que la clase B depende de la clase A.
En la mayoría de los casos, **las dependencias se reflejan en los métodos de una clase que utilizan el objeto de otra clase como parámetro** .
Una relación de dependencia es una relación de "uso". Un cambio en una cosa en particular puede afectar a otras cosas que la usan, y usar una dependencia cuando es necesario indicar que una cosa usa otra. Por ejemplo: El auto depende de la gasolina. Si no hay gasolina, el automóvil no podrá conducir.



Ejemplo-Diagrama de Clases

Diagrama de clases – Sistema de pedidos
El siguiente diagrama de clases modela un pedido de cliente de un catálogo minorista. La clase central es la **Orden** . Asociados a ella están el **Cliente** que realiza la compra y el **Pago** . Un **pago** es uno de cuatro tipos: **efectivo** , **cheque** , **crédito o transferencia bancaria** . El pedido contiene **OrderDetails** (artículos de línea), cada uno con su **artículo** asociado .



6.5. Sistemas de Gestión de hotelera:

Debes saber que un sistema de gestión hotelera o PMS (Property Management System en inglés) se trata de un software que te permite gestionar diferentes aspectos elementales en la operatividad de un hotel de una forma eficiente, ágil y, sobre todo, totalmente integrada. Una de sus principales ventajas es que te permite automatizar muchos procesos, de forma que no se produzcan errores y tengas información actualizada y confiable siempre disponible para una buena toma de decisiones.

6.5.1.Procesos clave de la gestión hotelera:

- **Gestión de Reservas y Recepción:** Desde la consulta hasta el check-in/check-out, incluyendo la asignación de habitaciones y la comunicación con el huésped.
- **Gestión de Housekeeping (Ama de Llaves):** Mantenimiento, limpieza y asignación de habitaciones para asegurar la comodidad y presentación.
- **Gestión Financiera y Revenue Management**: Control de ingresos, gastos, tarifas y estrategias de precios para maximizar beneficios.
- **Gestión de Alimentos y Bebidas (F&B):** Operaciones de restaurantes, bares y servicios de catering.
- **Marketing y Ventas**: Promoción del hotel, gestión de canales de venta y estrategias de fidelización.

6.5.2.Tecnologia en la gestión Hotelera:

- **Sistemas de Gestión de Propiedades (PMS):** Software centralizado para gestionar todas las operaciones.
- **Automatización:** En reservas, check-in/out, tareas administrativas y limpieza.
- **Gestión de Canales (CMS):** Para distribuir inventario en OTAs (Online Travel Agencies) y web propia.
- **Inteligencia Artificial (IA**): Para chatbots, personalización y análisis de datos.
- **Sistemas de Fidelización Digital:** Apps móviles, programas de puntos y comunicación personalizada.

6.5.3.Eficiencia y eficacia en la gestion hotelera:

-Productivisdad Hotelera:

La productividad hotelera se refiere a la eficiencia con la que un hotel utiliza sus recursos como el tiempo, el personal, la tecnología y los materiales, para ofrecer experiencias de alta calidad a los viajeros, al mismo tiempo que maximiza la rentabilidad.
Esto implica optimizar las operaciones en todos los departamentos, como la comunicación con los viajeros, el servicio de recepción, las tareas de limpieza y la oferta de alimentos y bebidas. Al mejorar los flujos de trabajo y reducir desperdicios, los hoteles pueden generar mayores ingresos sin comprometer la satisfacción del huésped.

Medir la productividad hotelera implica hacer seguimiento a indicadores clave de rendimiento que reflejan la eficiencia operativa, la satisfacción del huésped y la generación de ingresos. Estas métricas ofrecen información valiosa sobre áreas de mejora y orientan la toma de decisiones del equipo de gestión.
Tasa de Ocupación: Este indicador calcula el porcentaje de habitaciones disponibles que han sido vendidas en un periodo determinado. Una tasa de ocupación alta suele reflejar un mejor aprovechamiento del inventario, lo que impacta directamente en los ingresos.

Tarifa Promedio Diaria (ADR): La ADR mide el ingreso promedio por habitación vendida. Refleja la estrategia de precios del hotel y su capacidad para optimizar los ingresos por reservas.
Ingreso por Habitación Disponible (RevPAR): El RevPAR combina la tasa de ocupación y la ADR para ofrecer una visión integral del rendimiento

financiero del hotel. Es un indicador clave para evaluar la capacidad del hotel de generar ingresos, independientemente de su nivel de ocupación.

Ratio de Productividad del Personal: Este ratio calcula los ingresos generados por cada miembro del personal. Ayuda a evaluar la eficiencia del trabajo humano e indica qué tan bien se están utilizando los recursos humanos para contribuir a los resultados financieros. Ratios más altos indican un uso más eficiente del equipo.

Índice de Satisfacción del Huésped: Los comentarios de los clientes, recogidos a través de encuestas y plataformas de reseñas, proporcionan una visión directa sobre la experiencia del huésped. Una alta satisfacción suele estar asociada con una mayor productividad, ya que indica que el hotel ofrece un servicio de calidad de forma eficiente.

Al monitorear regularmente estos indicadores, los gerentes hoteleros pueden identificar tendencias, optimizar procesos e implementar mejoras que eleven tanto la eficiencia operativa como los resultados financieros.

6.5.4. Calidad y servicio:

La calidad en la gestión hotelera es la estrategia integral para superar las expectativas del cliente mediante la mejora continua de procesos (desde recepción hasta post-estancia) y servicios, combinando factores tangibles (instalaciones, limpieza, eficiencia) con intangibles (atención personalizada, empatía, proactividad), buscando la satisfacción del huésped y la rentabilidad del negocio, y se enfoca en la formación del personal y la optimización tecnológica.

Componentes clave de la calidad en gestión hotelera:

- Expectativas vs. Percepción: La calidad se mide por la diferencia entre lo que el cliente espera y lo que percibe.
- Tangibles: Limpieza, estado de las instalaciones, rapidez de procesos, seguridad.
- Intangibles: Empatía, proactividad, comunicación clara, respuesta ágil, fiabilidad del personal.
- Recursos Humanos: Personal capacitado y motivado es fundamental; la formación es clave.
- Procesos: Eficiencia en todas las áreas (recepción, A&B, housekeeping) y uso de tecnología.
- Mejora Continua: Auditorías internas, feedback del cliente y ajuste constante de estrategias (Ciclo PDCA: Planificar, Hacer, Verificar, Actuar).

Importancia para la gestión:

- Fidelización: Un servicio de alta calidad genera clientes recurrentes.
- Rentabilidad: Optimiza costos y aumenta ingresos.
- Diferenciación: Permite al hotel destacar frente a la competencia.
- Reputación: Mejora la imagen del establecimiento en general.

¿Cómo se implementa?

1. Definir Estándares: Establecer protocolos claros para cada servicio.
2. Formar al Personal: Invertir en capacitación y desarrollo de habilidades blandas y técnicas.
3. Tecnología: Utilizar sistemas de gestión y comunicación que agilicen operaciones.

4. Medir y Analizar: Usar encuestas, NPS y feedback para identificar áreas de mejora.
5. Cultura de Calidad: Involucrar a todos los *stakeholders* (clientes, empleados, proveedores) en el objetivo de la excelencia.

7.Desarrollo de propuesta:

En esta etapa reunimos los requerimientos solicitados por el cliente en la asignatura de analisis de sistemas para el desarrollo y correcto funcionamiento del sistema de gestion hotelera.los requeriemintos nos permite establecer de manera clara las funcionalidades y restricciones que el sistema debe cumplir, sirve como base para el diseño del diagrama de clases

7.1.Vision:

7.1.1 descripcion de los interesados:

Los interesados (stakeholders) del sistema de gestión hotelera son todas aquellas personas, grupos u organizaciones que influyen directa o indirectamente en el desarrollo, implementación, operación y uso del sistema, o que se ven afectados por sus resultados. Identificar y describir a los interesados es fundamental para garantizar que el sistema responda adecuadamente a sus necesidades, expectativas y responsabilidades.

a) Propietarios y Gerencia del Hotel

Son los principales interesados, ya que toman las decisiones estratégicas del negocio. Requieren información confiable y oportuna sobre ocupación, ingresos, egresos, rentabilidad, reportes financieros y desempeño general del hotel. El sistema debe facilitar la toma de decisiones mediante reportes claros, indicadores de gestión y acceso seguro a la información.

b) Administrador del Hotel

Es el responsable directo de la operación diaria. Utiliza el sistema para gestionar reservas, asignación de habitaciones, control de tarifas, facturación, reportes operativos y supervisión del personal. Necesita que el sistema sea intuitivo, eficiente y confiable para optimizar los procesos administrativos.

c) Recepcionistas

Son usuarios operativos clave del sistema. Se encargan del registro de huéspedes (check-in y check-out), gestión de reservas, actualización del estado de habitaciones y atención al cliente. El sistema debe permitir rapidez en las operaciones, reducción de errores y acceso inmediato a la información del huésped.

d) Personal de Limpieza y Mantenimiento

Estos interesados utilizan el sistema de manera indirecta o mediante módulos específicos para conocer el estado de las habitaciones, órdenes de limpieza, mantenimiento preventivo y correctivo. El sistema contribuye a mejorar la coordinación, eficiencia operativa y disponibilidad de habitaciones.

7.2. Especificacion de requerimientos:

7.2.1 Requeriemientos Funcionales:

Registrar los datos del huésped
Reservación de habitaciones online
Modificación de datos del huésped
Método de pago
Inventario de snacks
Visualización en pantalla de las habitaciones
Registro de check-in y check-out
Automatizar el periodo contable del hotel
Integración de redes sociales en la interfaz para clientes
Consulta de Historial de Clientes Frecuentes
Búsqueda Rápida de Huésped
Anulación de Reserva
Envío de Confirmación de Reserva
Registro de Pago Electrónico (Yape, Plin, etc.)
Generación de Comprobante de Pago
Aplicación de Descuentos
Consulta de Balance Pendiente
Actualización de Estado de Habitación (Administrativo)
Generación de Asignación de Limpieza
Consulta de Asignación de Limpieza Móvil
Reporte de Fin de Limpieza
Reporte de Objetos Olvidados
Reporte de Incidencia de Mantenimiento
Seguimiento de Tareas de Mantenimiento
Carga de Consumo Automático a Habitación
Registro de Nuevo Producto de Stock
Registro de Salida de Producto
Generación de Alerta de Bajo Stock
Registro de Fecha de Caducidad
Generación de Alerta de Caducidad
Modificación de Precios de Venta de Inventario
Búsqueda y Filtrado de Inventario
Generación de Reporte de Ingresos Diario Detallado
Generación de Reporte de Ocupación
Exportación de Reportes a CSV/PDF
Visualización de Dashboard Administrativo
Creación de Cuentas de Empleados
Asignación y Modificación de Roles de Acceso
Restablecimiento de Contraseñas de Usuario

Habilitación/Deshabilitación de Cuentas
Notificación de Consumos Pendientes
Configuración de Parámetros Globales
Integración con Plataformas de Reserva Online

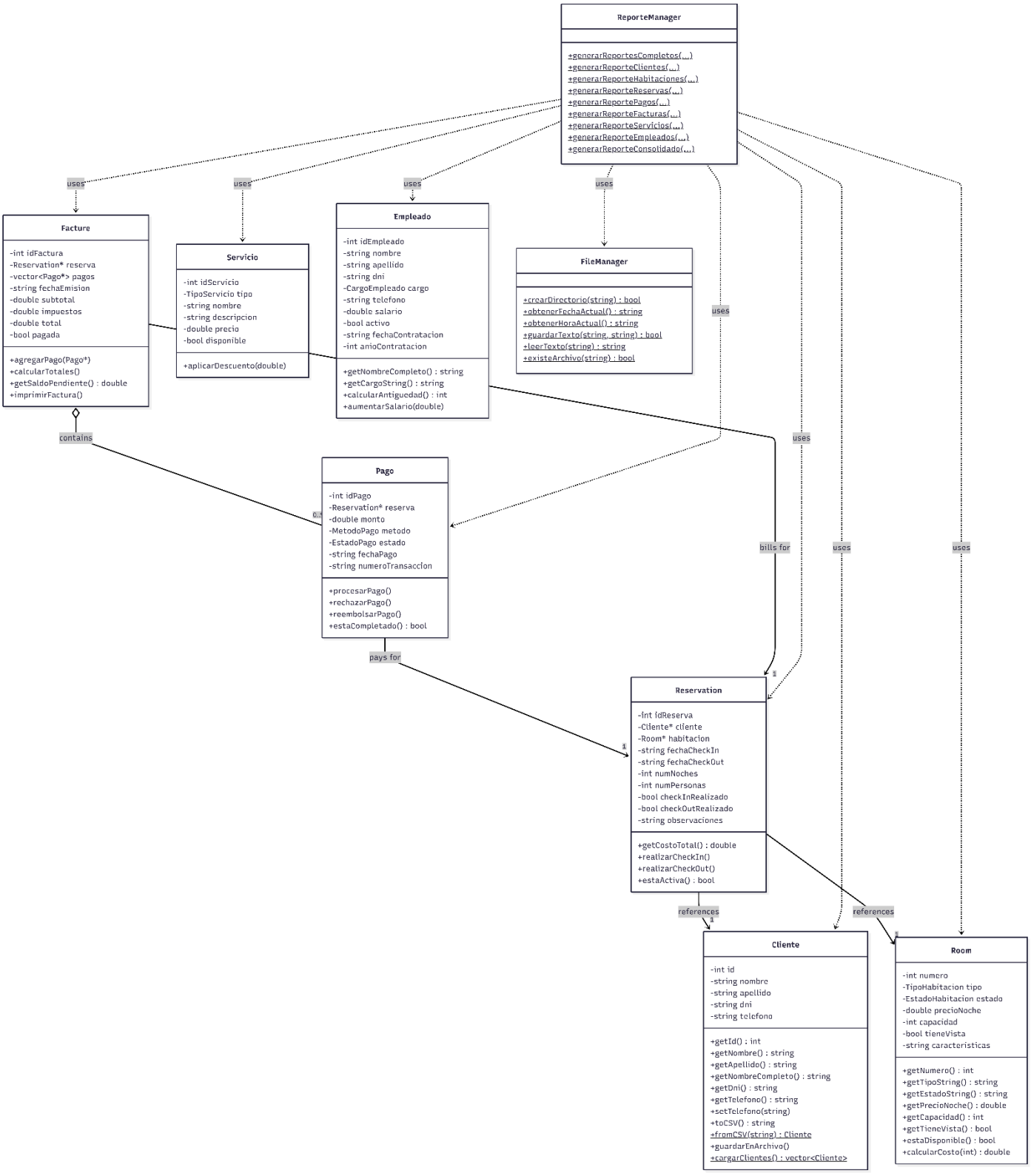
7.2.2 Requerimentos no funcionales:

Ejecución Automática de Copias de Seguridad
Usabilidad Intuitiva
Alto Nivel de Seguridad y Privacidad
Auditoría de Actividad (Trazabilidad)
Disponibilidad en la Nube
Soporte Multiplataforma

7.3.Arquitectura del software:POLLO

7.3.1. Vista de datos (Diagrama de Clases):

Diagrama de clases SUMAS:



10.Conclusiones y recomendaciones:

- La implementación del diagrama de clases permitió modelar de manera clara y estructurada los procesos fundamentales de la gestión hotelera, facilitando la comprensión del sistema y sirviendo como base sólida para su desarrollo mediante programación orientada a objetos.

- El uso de los principios de la programación orientada a objetos contribuyó significativamente a la eficiencia y eficacia del sistema, al promover un diseño modular, reutilizable y fácil de mantener, lo cual es esencial para sistemas de gestión hotelera.

- La identificación y análisis de los requerimientos funcionales y no funcionales fueron fundamentales para garantizar que el diseño del sistema responda adecuadamente a las necesidades operativas del hotel, reduciendo errores y mejorando la organización de la información.

- El desarrollo del sistema de gestión hotelera mediante un enfoque basado en diagramas de clases demuestra la importancia de una correcta fase de diseño en proyectos de programación avanzada, ya que permite optimizar el tiempo de desarrollo y facilitar futuras ampliaciones del sistema.

11.Bibliografia:

A. Shirafuji et al., "Exploring the robustness of large language models for solving programming problems", arXiv [cs.CL], 2023. https://doi.org/10.48550/arXiv.2306.14583 .[1]
B. J. López, y F. Ortin (2013). Soporte eficiente de herencia dinámica para lenguajes basados en clases y prototipos. J. Syst. Softw. , 86, 278-301. https://doi.org/10.1016/j.jss.2012.08.016.[2]
C. L Fuentes, A Vallecillo - Novática, 2004 - researchgate.net.[3]
D. https://blog.visual-paradigm.com/es/what-are-the-six-types-of-relationships-in-uml-class-diagrams/
E. https://www.siteminder.com/es/r/gestion-hotelera-significado/
F. **https://github.com/abflorec/UNIDAD_2PROUSCC**