Sistema de reservas para un hotel

Paradigmas de Programación 2023

Integrantes:

Paredes Solorzano, Adan Smith
Colque, Julieta Milagros
Gramajo, Cristian Ismael
Muñoz Quiroga, David Ignacio
Flores, Jorgelina

Docentes a cargo:

Lic. María Cristina Werenitzky Curia
Mg. Héctor A. Valdecantos

Carreras:

Lic. Informática
Programador Universitario
Ingeniería en Informática

Introducción

En este documento, explicaremos en detalle el transcurso del proyecto vigente. Desde los requerimientos para ser llevado a cabo, los cuales fueron dados tanto de manera concreta y a través de entrevistas con el docente a cargo, y también las consideraciones que, en grupo, decidimos que son necesarios para el funcionamiento esperado del sistema, sin tener que alejarnos demasiado de los requerimientos planteados. Además, también se tomarán en cuenta las etapas de desarrollo del sistema, es decir, la identificación de clases y sus relaciones; el diseño y modelo de la solución del problema usando un lenguaje de modelado (UML), y, por último, la implementación del sistema usando el lenguaje de programación C++.

1. Requerimientos del Sistema

En esta sección, se explicarán cuáles han sido los requerimientos (o requisitos) que se llegaron a dar de forma concreta como con entrevistas con el docente a cargo. Para ello, a continuación, se presentarán los mismos entre los elementos que se consideraron como destacados dentro del sistema.

1.1. Descripción general

CLIENTE:

Un cliente puede ser titular o acompañante, sus atributos solicitados son:

- Titular: nombre, apellido, DNI o pasaporte, nacionalidad, provincia/ciudad, email, domicilio, patente del vehículo y número de celular.
- Acompañante: nombres y nº de DNI/pasaporte.

Brinda los requerimientos para realizar su reserva como ser fecha de entrada/salida, número de camas, tamaño y capacidad de la habitación a reservar.

HABITACIÓN:

El estado de una habitación que puede estar *reservada*, *ocupada* o *libre*, se calcula de acuerdo a un periodo de tiempo, y cada estado está determinado de la siguiente manera:

- ***Reservada***: cuando se genera una reserva, la habitación que forma parte de la misma toma el estado de Reservada.
- ***Ocupada***: cuando se realiza el CHECK-IN en el hotel, la habitación pasa a estar ocupada todo el tiempo que dure la reserva.
- ***Libre***: cuando se realiza el CHECK-OUT del hotel, se libera la habitación para ponerla a disponibilidad de futuras reservas.

SISTEMA:

El sistema maneja la información relacionada a habitaciones y reservas realizadas en las mismas, huéspedes, amenities, productos de bar y gastos por consumo dentro del hotel. Además, el sistema revisa la disponibilidad de habitaciones. La búsqueda se realiza en aquellas habitaciones que se encuentren disponibles durante el periodo solicitado por el cliente. Sólo se tienen en cuenta aquellas habitaciones que se encuentran ***libre*** en dicho período.

La reserva se confirma mediante seña mínima del 30% (o más) del costo total de la misma.

CHECK-IN:

Se registra la fecha en la cual el cliente se presenta físicamente en el hotel para iniciar su estadía. En esta instancia, el cliente debe proporcionar información de sus acompañantes, si los tuviera. Es requisito necesario que el check-in pueda realizarse a partir de las 13hs del día de la reserva.

AMENITIES:

Con el número de habitación y una vez realizado el check-in, tanto el cliente como sus acompañantes, pueden solicitar el uso de amenities (por ejemplo gimnasio, pileta, etc).

El uso máximo de cualquier amenity es de 1 hora.

COMPRAS:

Desde la habitación, el cliente podrá realizar compras en el bar o cafetería del hotel. Dichas compras son facturadas al momento de hacer el check-out.

CHECK OUT:

El check-out se realiza antes de las 11hs de la fecha de salida y al momento de procesarlo, se debe calcular el importe a pagar teniendo en cuenta:

- Seña abonada durante la reserva.
- Monto por el uso de la habitación.
- Reserva de amenities.
- Compras realizadas.
- Penalidad**:**
    - **Media noche** de alojamiento si abandona la habitación entre las 11 y 13 hs.
    - **Noche completa** de alojamiento si se encuentra fuera de las horas mencionadas anteriormente.

2. Consideraciones particulares de diseño

En esta parte, se explicarán cuáles fueron las consideraciones que se fueron tomando en cuenta a partir de los requerimientos explicados en la sección anterior y agregando ciertos criterios para facilitar la manipulación de la información, por ejemplo búsquedas, entre otras operaciones.

CLIENTE:

En una reserva de habitación un cliente es acompañante, pero en otra reserva de habitación este cliente puede ser titular.

El cliente **NO** tiene acceso al número de reserva, sólo puede operar con el número de habitación, ya que en lo cotidiano los números de reserva son desconocidos para el cliente.

HABITACIÓN:

La habitación *NO* tiene asociada una propiedad de estado, ya que es un campo calculado y el estado varía según las condiciones de fecha y reservas asociadas. Además, una reserva de habitación, solo contiene reservas válidas para la fecha, no un histórico, o sea, el programa no trabaja sobre una base de datos, sino con lo que tiene en memoria en tiempo de ejecución.

RESERVA HABITACIÓN:

La reserva de una habitación cuenta con la siguiente información:

- *fechaEntrada, horaEntrada, fechaSalida* y *horaSalida* de la reserva de habitación,
- *fechaIn*, *horaIn, fechaOut* y *horaOut*, que corresponden a la fecha y hora del check-in y check-out respectivamente.
- *seniaMin*: refiriéndose a la seña mínima que es requerido para confirmar una reserva.

También es necesario considerar si un cliente realizó o no una compra, y si agregó o no una reserva de algún amenity.

AMENITY - RESERVA AMENITY:

Se consideraron amenities que van incluidos en la estadía y otros que tienen un costo, por otro lado, no se puede hacer una reserva del mismo amenity para la misma hora en ese mismo día.

SISTEMA:

Además de llevar la verificación de la disponibilidad de habitaciones, el sistema se encarga de realizar el check-in y check-out de la reserva en cuestión ya que es parte de la gestión de reservas de habitaciones.

3. Etapas de desarrollo

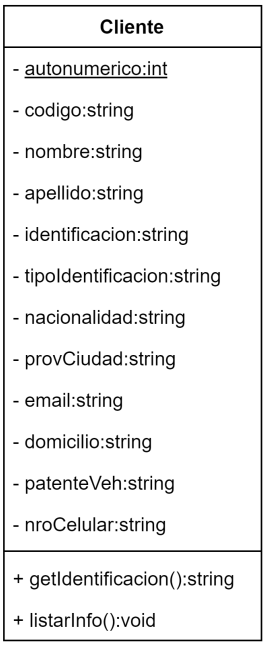
ETAPA N°1 - Clases

En esta etapa se identifican las clases, sus respectivos atributos y métodos de acuerdo a ciertas consideraciones.

CLIENTE:

En un diseño inicial se planteó una generalización en donde las subclases *Titular* y *Acompañante* heredan de la clase base *Persona*. Pero considerando que un acompañante podría ser titular en un futuro, se decidió plantear una nueva solución:

- Se diseñó una única clase **Cliente** y en caso de que en una nueva reserva un acompañante quiera ser titular, se completarán los datos faltantes.
- **Relación**: La clase **Cliente** tiene el tipo de relación agregación con la clase *Reserva Habitación* de dos maneras, una como acompañantes y otra como titular.



HABITACIÓN:

La clase *Habitación* contiene los datos principales de una habitación y, en cuanto a funcionalidades, se encuentran principalmente:

- determinar la disponibilidad de acuerdo al periodo fijado,
- calcular el costo total de la reserva por noche.

La disponibilidad de la habitación se realiza a partir de los registros de reservas de la misma, teniendo en cuenta que el periodo a querer fijar para una reserva no se solape con un periodo de reserva registrado.

Por ejemplo, si un cliente quiere reservar en una habitación, además de dar a conocer las características específicas de la misma, lo hace para el periodo 20/12/2023 - 23/12/2023. Luego se desea registrar una nueva reserva para el 22/12/2023 - 27/12/2023, pero no se podrá concretar la reserva debido a que ya existe una reserva cuyo periodo intersecta con el periodo solicitado por este nuevo cliente.

Relaciones: la clase *Habitación* tiene una asociación binaria con la clase *ReservaHabitación* por lo que posee una lista de reservas propia para cada objeto de tipo *Habitación*.

Habitacion
- nroHabitacion:int - nombrePopular:string - tamanio:float - capacidad:int - nroCamas:int - precioPorNoche:float
+ getNroHab(): int + getNroCamas(): int + getCapacidad(): int + getTamanio(): int + getPrecioPorNoche():float + hayDisponibilidad(fechaEntrada:Fecha, fechaSalida:Fecha): tipoEstado {Reservada, Ocupada, Libre} + costoTotalReserva(fechaE: Fecha, fechaS: Fecha):float + agregarReserva(reserva):void

RESERVA HABITACIÓN:

Responsabilidades principales:

- Check-in, check-out.
- Verifica titular/acompañantes.
- Calcula penalizaciones.

Relaciones:

- Asociación binaria con la clase habitación. Su estructura tiene un objeto de tipo *Habitación*.
- La clase *ReservaHabitacion* tiene una relación de composición con la clase *Sistema,* es decir, que la clase *Sistema* será la encargada de crear y eliminar los objetos de tipo ReservaHabitacion.
- La clase *Consumo* tiene una relación de composición con la clase *ReservaHabitacion* por lo que esta última clase será la encargada de crear y eliminar objetos de tipo consumo.

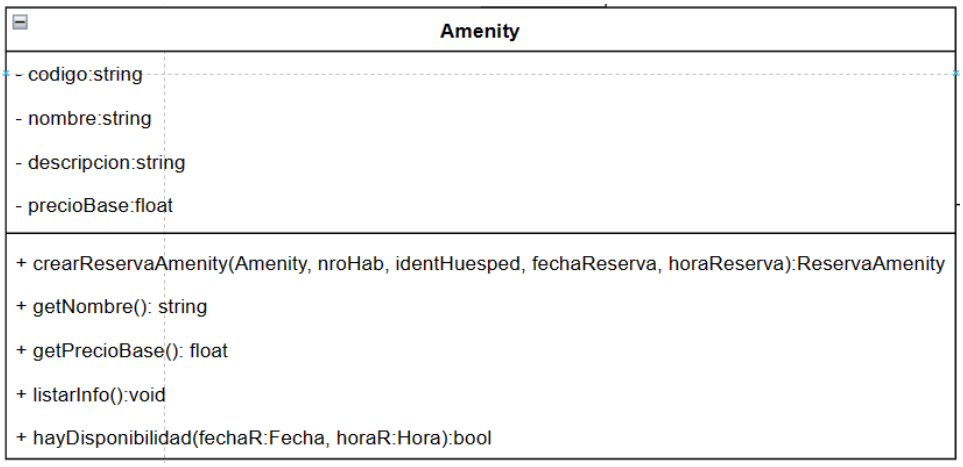
AMENITY:

Relaciones:

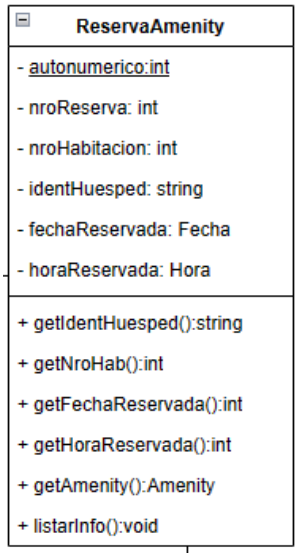
- Tiene una asociación binaria con la clase *ReservaAmentity*. Recibe un objeto de esta última para poder crear y registrar la reserva de un amenity.
- La clase *Amenity* es quien crea las reservas del amenity en cuestión, y la clase *ReservaAmenity* tiene una referencia a un objeto de la clase *Amenity*. Si un objeto de tipo *Amenity* desaparece, las reservas asociadas a la misma también.
- Tiene una relación por composición con la clase *Sistema*, esta crea los objetos de tipo *Amenity* y su tiempo de vida depende de la vida del sistema.

Responsabilidades principales:

- Verifica la disponibilidad de los amenities cuando los huéspedes quieren reservar.
- Crear reservas.



RESERVA AMENITY



Relaciones:

- Asociación simple con la clase *Amenity*, en su estructura tiene un objeto de tipo amenity.
- Agregación con la clase *Consumo*.

Responsabilidades principales:

- Almacenar en una sola clase la información, como identificación del huésped, número de habitación y los datos de la reserva del amenity.

PRODUCTO BAR - COMPRA - CONSUMO

Relaciones producto:

- Producto tiene una relación de composición con sistema, es decir sistema se encargará de crear los productos y eliminarlos.
- Tiene una relación de agregación con compra. La clase compra tiene en su estructura un objeto de tipo *productoBar*.

Relaciones Compra:

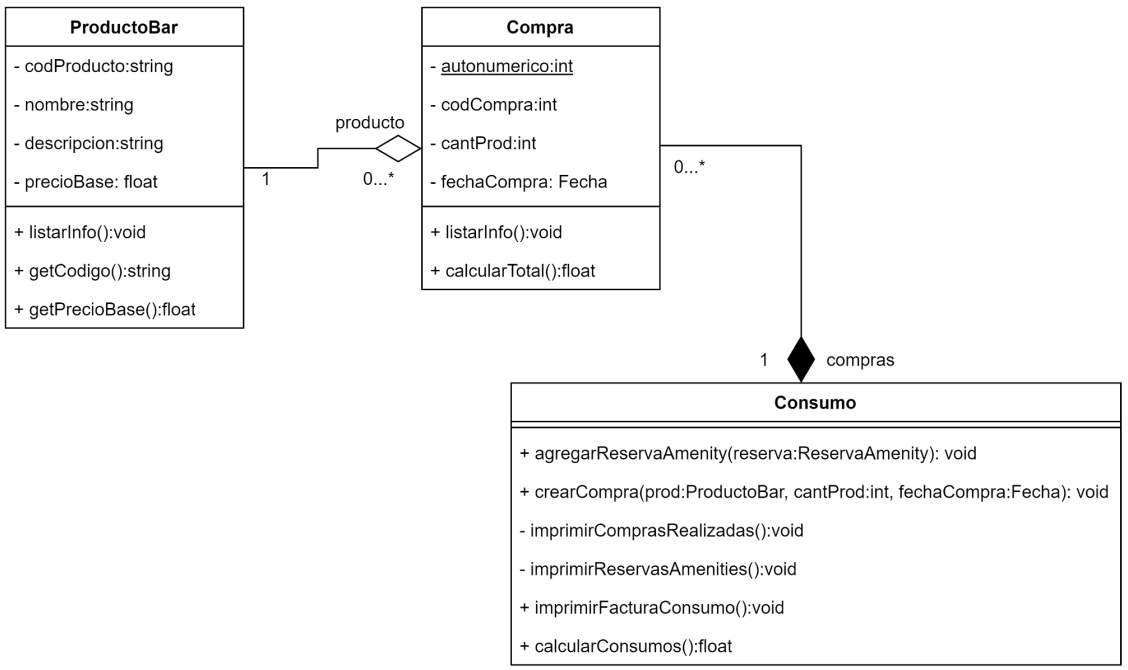
- Compra tiene una relación de composición con la clase consumo.

CONSUMO

Es una clase creada para englobar los extras de la estadía del huésped, como ser las reservas de amenities, compras en el bar o cafetería, brinda la posibilidad de acceder a sus detalles y calcular el costo total de los mismos para componer la clase *reservaHabitacion*.

Relaciones Consumo:

- Relación de agregación con *ReservaAmenity*, entonces guarda una lista de reservas de Amenities.
- Composición con la clase *reservaHabitacion*, consumo es creado y eliminado por esta.



SISTEMA

Responsabilidades:

- Crea las habitaciones, amenities, compras, productos.
- Busca las habitaciones disponibles para reservas.
- Crea reservas.
- Realiza el Check in.
- Permite reservas de amenities.
- Permite compras de productos asociadas a una habitación.
- Realiza el Check out

Relaciones:

- Composición con *Amenity*, *Habitación, ReservaHabitacion, ProductoBar*.

Sistema
+ crearAmenity(codigo:string, nombre:string, descripcion:string, precioReserva:float):void
+ crearHabitacion(nroHab:int, nombrePopular:string, tamanio:float, capacidad:int, nroCamas:int, precioPorNoche:float):void
+ crearProductoBar(codProd:string, nombre:string, descripcion:string, precioBase:float):void
+ crearReservaAmenity(nroHab:int, identHuesped:string, nombreAmenitie:string, fechaReserva:Fecha, horaReserva:Hora):void
+ crearReservaHab(titular:Cliente, nroHab:int, fechaEntrada:Fecha, fechaSalida:Fecha, seniaMin: float):void
+ crearCompra(nroHab:int, codProd:string, cantProd:int, fechaCompra:Fecha): void
+ checkIn(nroHab:int, identTitular:string, fechaLlegada:Fecha, horaLlegada:Hora, acompanantes:Lista de clientes):void
+ checkOut(nroHab:int, fechaOut:Fecha, horaOut:Hora):void
- buscarHabDisponible(fechaEntrada:Fecha, fechaSalida:Fecha, nroCamas:int, capacidad:int, tamanio:float):habitacion

MODIFICACIONES

FECHAHORA - FECHA HORA:

En un principio se diseñó la clase *FechaHora*, pero abandonamos su implementación.

Con motivo de mantener clases más atómicas y manejables, separamos las clases para que cada una opere de manera independiente.

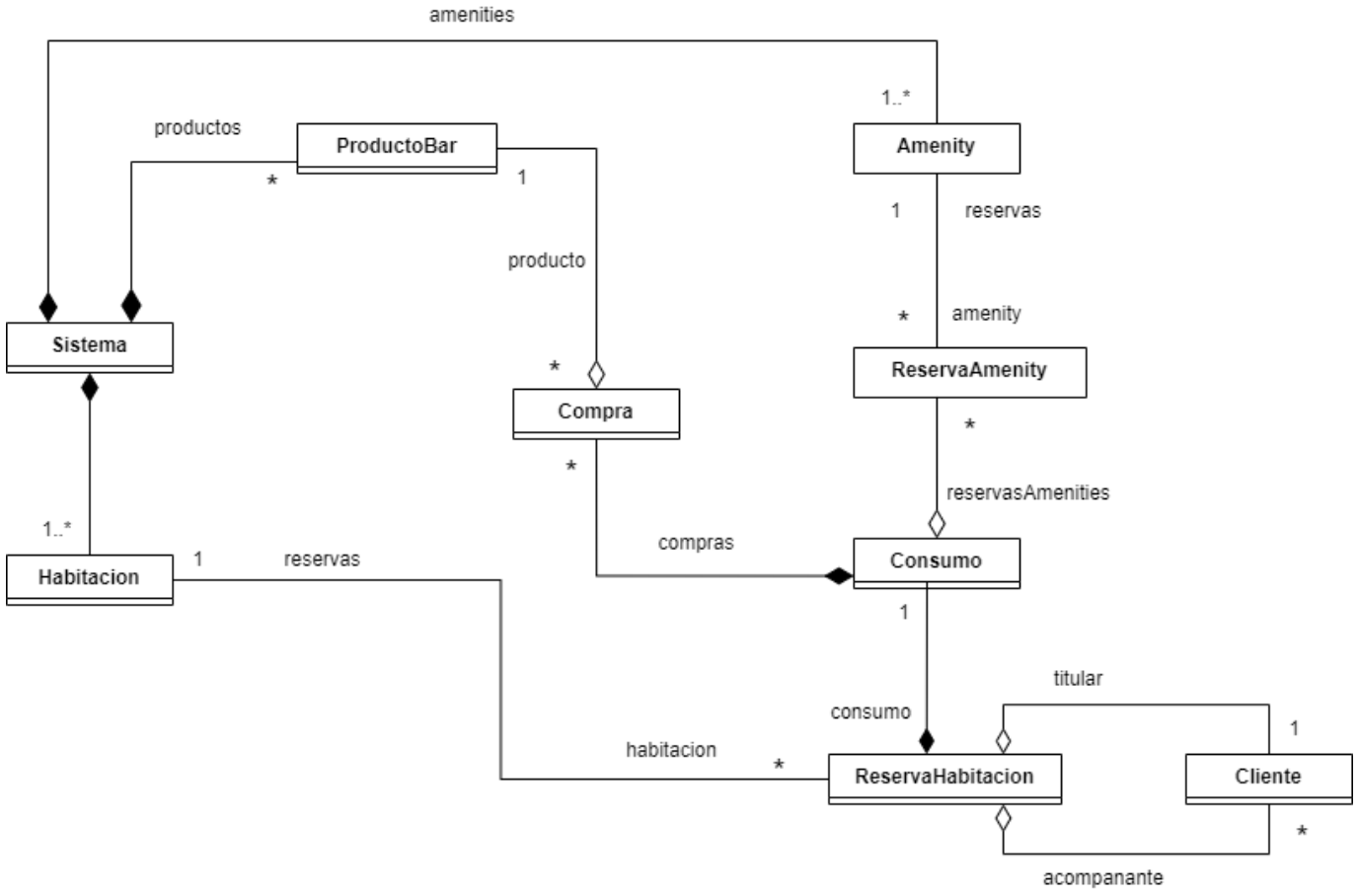
PRODUCTO - AMENITY:

En su momento, se consideró la clase *ProductoBar* y la clase *Amenity* como clases que heredan de una una clase base *Servicio*, ya que estas comparten propiedades y también tienen alguna similitudes en sus métodos. Sin embargo, se llegó a la conclusión que el contexto en el que se operaban eran totalmente diferentes, ya que el producto es un bien tangible y los amenities, por el contrario, son intangibles. Por lo tanto, se descarta la posibilidad de una herencia.

PRODUCTO BAR - COMPRA - CONSUMO

- Se modificó el nombre *ProductoAlimenticio* por *ProductoBar* para tener un nombre de clase más genérico con respecto a lo que ofrece el hotel.
- Se agregó en la clase *ProductoBar* un método para obtener el código del producto. Sirve para iterar sobre una lista de productos que tiene la clase sistema, explicada más adelante.

ETAPA N° 2 - Diagrama de clases



ETAPA N°3 - Detalles de Implementación

LIBRERIA <vector>

Se utilizó la librería <vector> para usar la clase contenedora “Vector” y realizar los registros de habitaciones, amenities, productos del bar, reservas de habitaciones y de amenities, y los acompañantes del titular de la reserva de habitación. La razón por la que usamos esta clase contenedora es debido a que es la clase con la que más trabajamos a lo largo de la materia. Por lo que, a fines de simplicidad y tiempo, ha sido la clase contenedora elegida.

DECLARACIONES INCOMPLETAS:

Para realizar las asociaciones binarias entre las clases *Habitacion-ReservaHabitacion* y *Amenity-ReservaAmenity*, fue necesario hacer declaraciones incompletas de clases para su correcto funcionamiento. En la clase ReservaHabitacion, se hizo una declaración incompleta de la clase *Habitacion* y en la clase *ReservaAmenity* se hizo una declaración incompleta de la clase *Amenity*.

HERRAMIENTAS:

Principalmente, se trabajó con Git para manejar los distintos archivos del proyecto y así se pueda tener un ambiente confortable en cuanto al manejo de las versiones del proyecto. También se usaron dos IDE’s: Visual Studio Code y Eclipse. Se utilizó Eclipse para crear la implementación del proyecto en C++, poder compilarlo y ejecutarlo. En cuanto a Visual Studio Code, su uso fue más para usar una extensión del mismo que permitió trabajar con el proyecto de una colaborativa dinámicamente en tiempo real, la cual es LiveShare.

DESTRUCTORES:

En el proyecto, existen clases donde no se definieron los destructores de las mismas, pero otras donde sí era necesario definirlos para llevar una buena gestión de memoria. Un ejemplo es en la clase *Sistema*, la cual debe eliminar los vectores de habitaciones, amenities, productos y reservas de habitaciones por la relaciones por composición que tiene con las clases *Habitacion, ReservaHabitacion, Amenity y ProductosBar*. Otro ejemplo es la clase *Amenity*, que gestiona sus propias reservas, por lo que, si el Amenity desapareciera, también se eliminarán automáticamente sus reservas.

CLASES FECHA Y HORA

Las clases Fecha y Hora que se usaron en el proyecto fueron las ofrecidas por la cátedra que ya contaban con algunos operadores sobrecargados, y se agregaron sobrecarga de otros operadores (como, por ejemplo, el operador ‘==’ para la clase Fecha) y poder realizar ciertas operaciones de algunos métodos de las clases existentes.