

19-08-2024



**Metodologías de la Programación II**

**Trabajo Final - Informe**

Docente: Claudia Cappelletti

Integrantes:

Lautaro Duarte

Ian Galarza

Franco Giordano

Jonathan Lekander

Tomas Rippa

Grupo N°4

Índice

[**Introducción 1**](#_ml4q8ghm6qz4)

[**Diagrama UML 2**](#_zdf2ltikak16)

[**Diagrama de secuencias 3**](#_5x0y9x87wz7y)

[**Implementación y especificación de clases utilizadas 3**](#_c6gdq1pofams)

[**Especificaciones 3**](#_ly25v2ih471u)

[**Implementaciones 4**](#_8qm76k8q3z2u)

[**Implementación de la aplicación en Smalltalk 4**](#_emalr7nddfh1)

[**Metodología utilizada en el proyecto 5**](#_g09qrlirhq3k)

[**Frameworks para desarrollar el proyecto en POO 5**](#_fwew9pjne4y1)

[**Casos de uso de Patrones 5**](#_540pt0o4t3ul)

[**Repositorios utilizados en el desarrollo del software 5**](#_uv6bvfjr70i)

[**Bibliografía 5**](#_6qdj53exmiq)

# Introducción

En el presente informe, haremos uso y aplicación de diversos conceptos y metodologías abordadas durante la cursada de la materia *Metodologías de la Programación II.* Para el actual caso, la elaboración de este informe gira en torno al desarrollo de una aplicación en lenguaje SmallTalk

Primeramente, vamos a dar una breve descripción del contexto sobre el cual parte la idea del desarrollo. A su vez, detallaremos la especificación e implementación de cada clase involucrada en el entorno planteado, como también la codificación de la aplicación que involucra a estas clases. Posteriormente, explicaremos las metodologías de trabajo empleadas por los integrantes de este equipo, tratando particularmente con aquellas que sean consideradas prácticas ágiles.

# Desarrollo

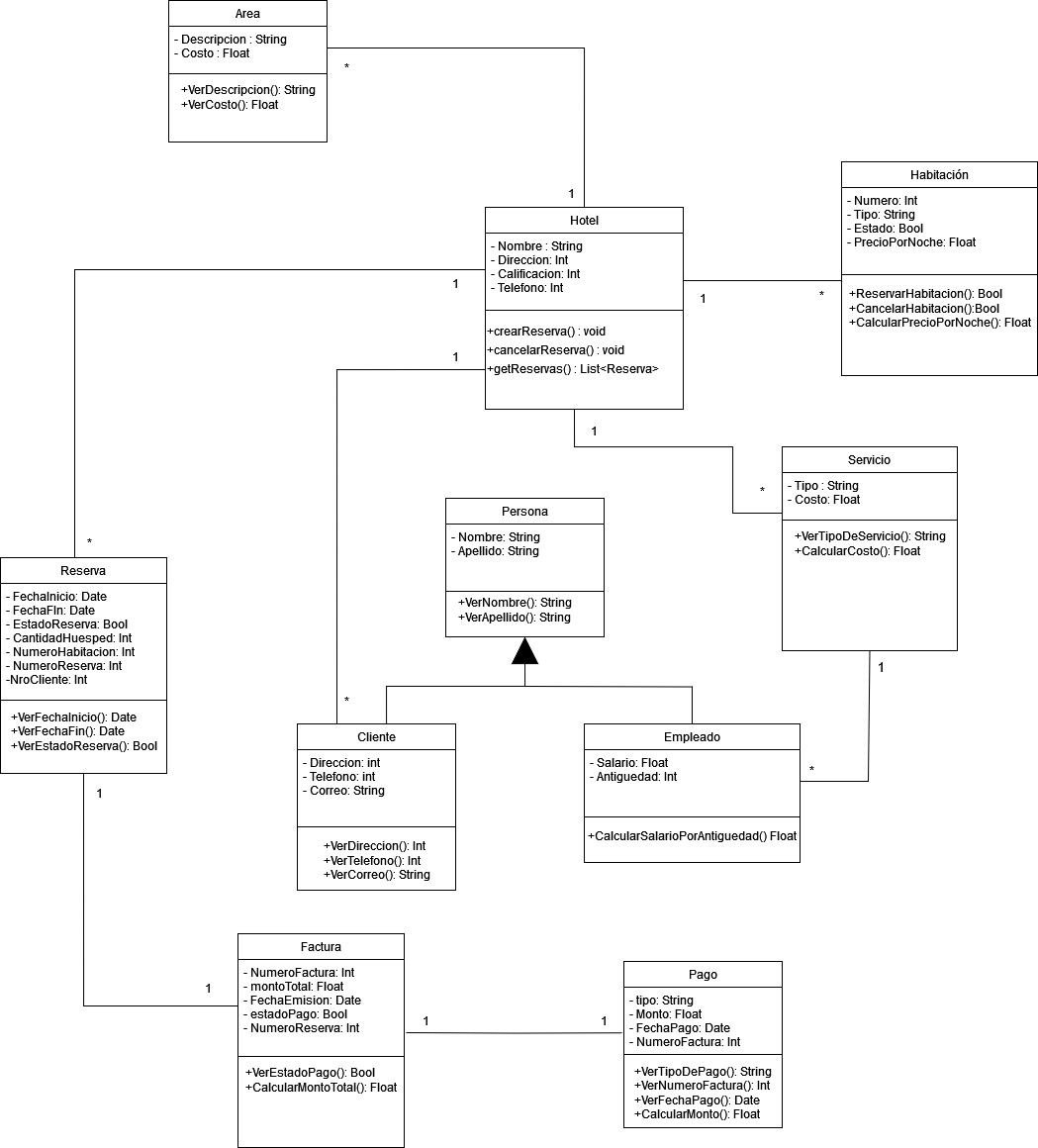
La aplicación desarrollada para la elaboración de este informe se trata de un sistema de reservas de hotel que hace uso de funcionalidades vistas durante la cursada de la materia, para poder llevar adelante la implementación es necesario plantear un problema a resolver.

**Problema:**

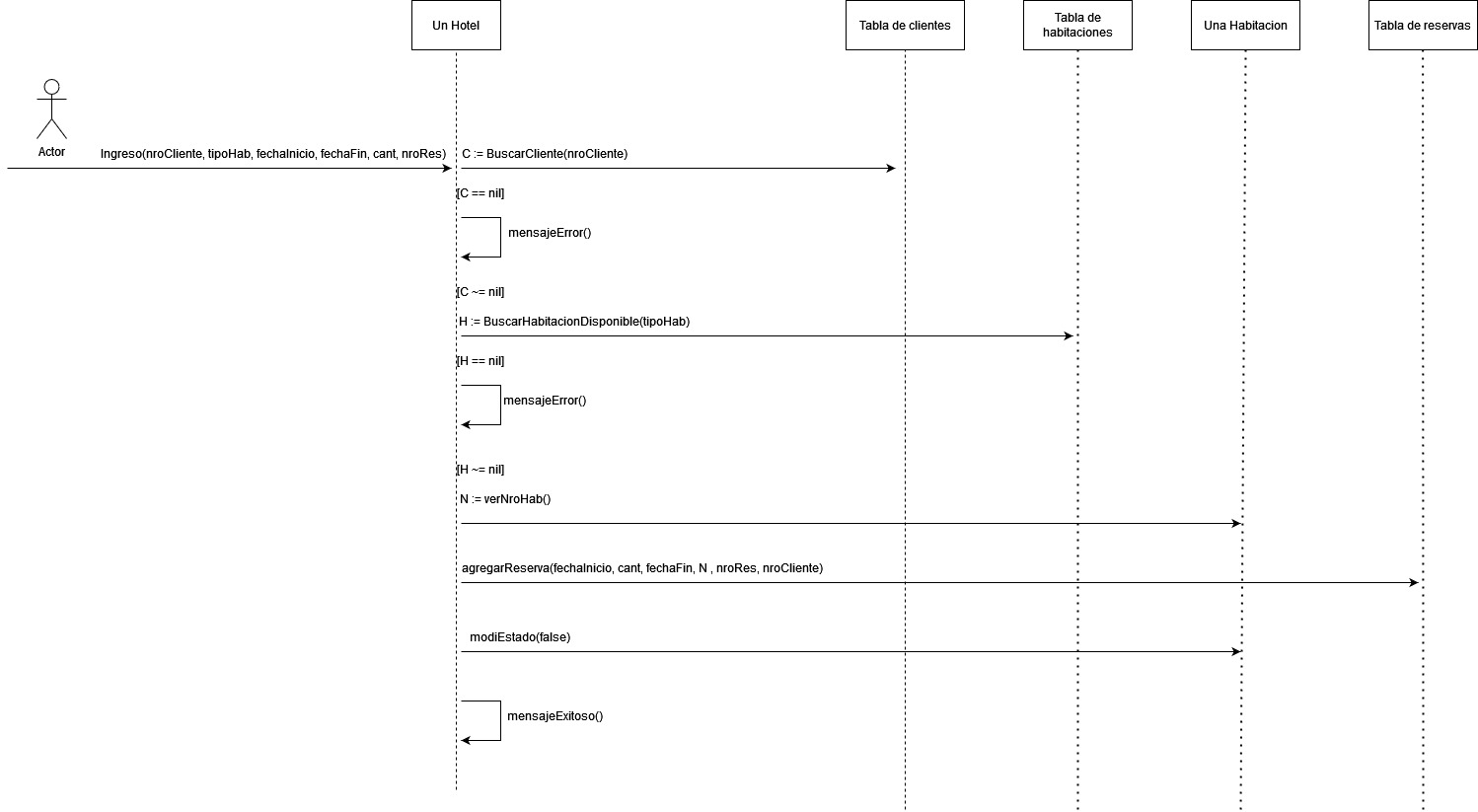
Una persona busca realizar una reserva en la recepción de un hotel, el recepcionista del hotel accede al sistema de reservas del hotel y recibe datos por parte de la persona (número de cliente, tipo de habitación deseada, fecha de inicio y fecha de fin, cantidad de huéspedes)

Con los datos recibidos el recepcionista verifica en el sistema que la persona sea un cliente registrado, en caso de que no esté registrado se notificará en pantalla que no está registrado, si la persona está registrada como cliente se procederá a verificar que haya habitación del tipo solicitado, si hay se verificará que esté disponible para reservarla, en caso de que no haya habitación que cumpla con los requisitos del cliente se notificará en pantalla, de lo contrario se procede a solicitar un numero de reserva, se realiza la reserva, se modifica el estado de la habitación para indicar que está ocupada y se notifica en pantalla que la reserva fue realizada con éxito.

# Diagrama UML



# Diagrama de secuencias



# Implementación y especificación de clases utilizadas

## A continuación se detallarán las clases desarrolladas en nuestro sistema de reservas para hoteles. Especificaciones

|  |  |  |
| --- | --- | --- |
| Clase | Atributos | Métodos principales |
| Hotel | nombre  dirección  calificación  teléfono  áreas  reservas  habitaciones  servicios  clientes | initHotelnom:dir:cal:tel:  agregarReserva: unaRes  agregarHabitacion: unaHab  agregarCliente: unClient  todasLasReservas  todasLasHabitaciones  todosLosClientes  buscarCliente:unNum  buscarHabitacionDisponibleTipo: unTipo  buscarHabitacionPorNumero: unNum  reservasVigentes  verNombre |
| Reserva | fechaInicio  fechaFin  estadoReserva  cantidadHuesped  numeroHabitacion  numeroReserva  numeroCliente | initHabitacionnro:tipo:precioPorNoche:  esVigente: unaFecha  cantHuesped  modiEstadoReserva: unEstado  modiFechaFin: unaFecha  modiFechaInicio: unaFecha  modiNumeroHabitacion: unNum  verEstadoReserva  verFechaFin  verFechaInicio  verNumeroCliente  verNumeroReserva  verNumeroHabitacion |
| Habitación | nroHab  tipo  estado  precioPorNoche | initHabitacionnro:tipo:precioPorNoche:  verEstado  verNroHab  verTipo  verPrecioPorNoche  modiEstado: unEstado  modiprecioPorNoche: unPrecio |
| Cliente | nombre  apellido  dirección  correo  teléfono  nroCliente | initClientDireccion:correo:telefono:nombre:apellido:  verNroCliente  verCorreo  verDireccion  verTelefono |
| Persona | nombre  apellido | initPersonaNombre:apellido:  verApellido  verNombre |
| Empleado | nombre  apellido  salario  antigüedad | initEmpleadoSalario:antigüedad:nombre:apellido:  modiAntiguedad:  verAntiguedad  verSalario |
| Factura | nroFactura  montoTotal  fechaEmision  estadoPago  nroReserva | calcularMontoTotal  verEstadoPago  initFacturanroFactura:montoTotal:fechaEmision:estadoPago:nroReserva: |
| Servicio | tipo  costo | initServiciotipo:costo:  modiCosto: unCosto  modiTipo: unTipo  verCosto  verTipo |

## 

## Implementaciones clases utilizadas

### Hotel

**Método de clase:**

* crearHotelnom:dir:cal:tel

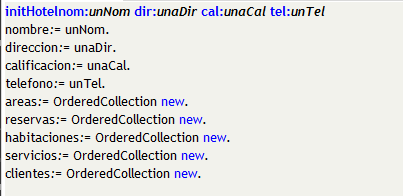
Crea una instancia del objeto Hotel inicializando sus atributos



**Métodos de instancia:**

* initHotelnom:dir:cal:tel:

Inicializa un Hotel dentro del sistema. En el mismo se le debe asignar sus 4 atributos, además el método realiza la creación de las colecciones dentro del hotel.



* agregarHabitacion:

Agrega una instancia de Habitación a la colección de habitaciones del Hotel



* agregarCliente:

Agrega una instancia de Cliente a la colección de clientes del Hotel



* agregarReserva:

Agrega una instancia de Reserva a la colección de reservas del Hotel



* buscarCliente:

Detecta y retorna el primer objeto Cliente en la colección de clientes del Hotel que coincida con el número de cliente proporcionado. Si no se encuentra retorna nil.



* buscarHabitacionDisponibleTipo:

Detecta y retorna el primer objeto Habitación en la colección de habitaciones del Hotel que coincida con el tipo proporcionado y tenga estado true. Si no se encuentra retorna nil.



* buscarHabitacionPorNumero:

Detecta y retorna el primer objeto Habitación en la colección de habitaciones del Hotel que coincida con el número de habitación ingresado. Si no se encuentra retorna nil.

****

* todasLasHabitaciones

Retorna la colección de habitaciones del Hotel.



* todasLasReservas

Retorna la colección de reservas del Hotel.



### Habitación

**Método de clase:**

* crearHabitacionnro:tipo:precioPorNoche:

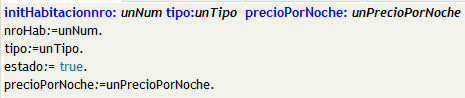
Crea una instancia objeto Habitación inicializando sus atributos



**Métodos de instancia:**

* initHabitacionnro:tipo:precioPorNoche:

Inicializa una habitación dentro del sistema. En el mismo se le debe asignar sus 3 atributos, además el método asigna ‘true’ al atributo estado.



* modiEstado:

Modifica el atributo estado de la Habitación



* modiPrecioPorNoche:

Modifica el atributo precioPorNoche de la Habitación



* verEstado

Retorna el atributo estado de la Habitación



* verNroHab

Retorna el atributo nroHab de la Habitación



* verTipo

Retorna el atributo tipo de la Habitación.

****

* verPrecioPorNoche

Retorna el atributo precioPorNoche de la Habitación.



### Reserva

**Método de clase:**

* crearReservaIni:Fin:cantHuesp:numHab:nroRes:nroCliente:

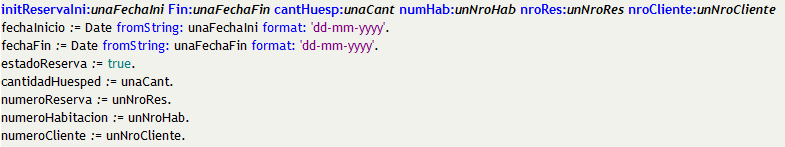
Crea una instancia del objeto Reserva inicializando sus atributos.



**Métodos de instancia:**

* initReservaIni:Fin:cantHuesp:numHab:nroRes:nroCliente:

Inicializa una reserva dentro del sistema. En el mismo se le debe asignar sus 6 atributos, además el método asigna ‘true’ al atributo estadoReserva



* verFechaInicio

Retorna la fecha de inicio de la Reserva



* verFechaFin

Retorna la fecha de fin de la Reserva



* verEstadoReserva

Retorna el estado de la Reserva



* cantHuesped

Retorna la cantidad de huéspedes de la Reserva



* verNumeroHabitacion

Retorna el número de habitación de la Reserva



* verNumeroCliente

Retorna el número de cliente de la Reserva



* verNumeroReserva

Retorna el número propio de la Reserva



### Cliente ← Persona

**Método de clase:**

* crearClienteNombre:apellido:dirección:teléfono:correo:nroCliente:

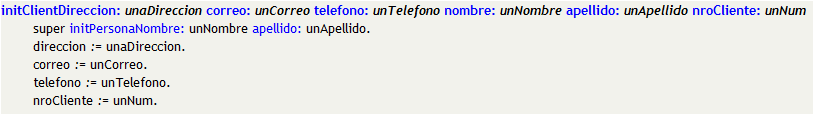
Crea una instancia del objeto Cliente inicializando sus atributos.



**Métodos de instancia:**

* initClientDireccion:correo:teléfono:nombre:apellido:nroCliente:

Inicializa un Cliente dentro del sistema. En el mismo se le debe asignar valores para sus 6 atributos, 2 de los cuales pertenecen a su superclase (Persona).



* verNroCliente

Retorna el atributo nroCliente del Cliente.



* verDireccion

Retorna el atributo dirección del Cliente

****

* verTelefono

Retorna el atributo teléfono del Cliente



* verCorreo

Retorna el atributo correo del Cliente



### Persona

**Métodos de instancia:**

* verNombre

Retorna el atributo nombre de la Persona.

****

* verApellido

Retorna el atributo apellido de la Persona.



# Implementación de la aplicación en Smalltalk

# Metodología utilizada en el proyecto

# Frameworks para desarrollar el proyecto en POO

# Casos de uso de Patrones

# Repositorios utilizados en el desarrollo del software

# 

# Bibliografía