Programación 4

Informe del
Modelo de Casos de Uso

Grupo 24
Integrantes

Alexis Moderana Maitia	**CI:** 4.829.917-6
Eber Manuel Rodriguez Gonzalez	**CI:** 5.097.757-4
Francisco Mauri Canale	**CI:** 5.140.198-4
Ailen Monroy Aiscorbe	**CI:** 5.369.824-2
Federico Mello Gutierrez	**CI:** 4.709.146-4

Docente: Leonel Peña

1 Cambios e impacto de nuevos requerimientos

1.1 Cambios generales

En el caso de uso Alta de Habitación, en la operación de sistema nuevaHabitacion, se cambió para que recuerde un datatype y no la instancia de la habitación, y así lograr guardar datos temporales correctamente. Además se agregó una nueva operación del sistema getConsumiblesRegistrados que retorna un set de enteros, los códigos de los tipos de consumo registrados en el sistema. Luego en la operación confirmarRegistroHabitacion se cambió el tipo del parámetro nombreHostal, pasando a ser de tipo String para no pasar un exceso de información.

En el caso de uso Asignar Empleado a Hostal, en la operación del sistema getEmpleadosDisponibles el tipo del valor devuelto pasa a ser un Set(String) para evitar pasar información innecesaria.

En el caso de uso Consulta de Usuario, se agregó una operación getUsuarioDetalle que retorna la información solicitada por el caso de uso, y la operación getUsuarios se modificó para que solo devuelva los identificadores (emails) de los usuarios para nuevamente evitar pasar más información que la necesaria.

Para el caso de uso Realizar Reserva se agregó una operación para liberar la memoria recordada por el sistema. También se cambió la operación del sistema getHostales por la operación del sistema getHostalesConCalificacion, la misma se adapta más a los requerimientos del caso de uso, esto se debe a que este devuelve un Set(DTHostal), a diferencia de getHostales que retorna un Set(String) y que por lo tanto no devuelve toda la información requerida.

En el caso de uso Consulta de Hostal se agregó la operación del sistema getInfoBasicaHostal que retorna la información básica del hostal.

En el caso de uso Consulta de Estadía fueron eliminadas las operaciones del sistema getHuesped y getHabitacion, fueron consideradas innecesarias para cumplir con los requerimientos del caso ya que es más simple brindar dicha información en la operación getEstadias.

1.2 Impacto de nuevos requerimientos

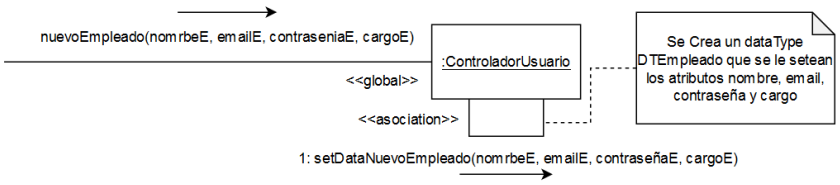
En el caso de uso Calificar Estadía, fue cambiada la operación calificarEstadia, dejamos de pasarle el parámetro nombreHostal, el identificador del hostal, debido a que no es necesario para realizar la operación. Además se cambió el contrato de esta operación porque además de calificar la estadia, ahora la operación se encarga de notificar a los Empleados suscritos que se creo una nueva Calificacion, esto provocó un cambio en el modelo de dominio haciendo que exista una nueva relación entre calificación y empleado que posteriormente fue implementada mediante un patrón de diseño observer.

A su vez se introdujo un nuevo datatype DTCalificacionNotificada como producto de la introducción de las nuevas especificaciones, en donde se almacena la información asociada a cada notificación.

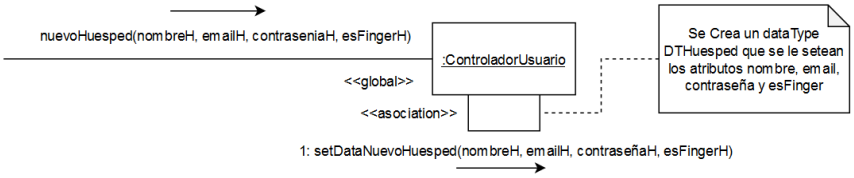
2 Realización de Casos de Uso

2.1 Alta de Usuario

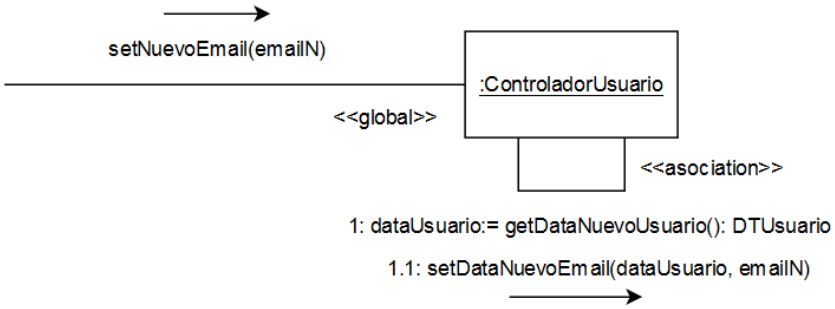
nuevoEmpleado(String, String, TipoCargo)



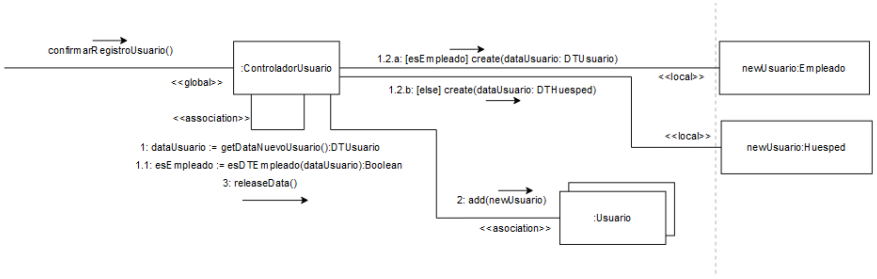
nuevoHuesped(String, String, String, Boolean)



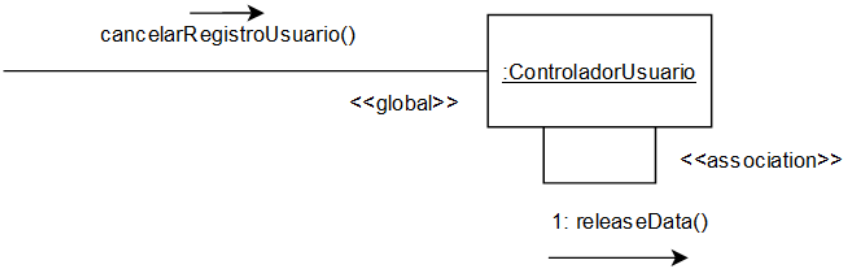
setNuevoEmail(String)



confirmarRegistroUsuario()

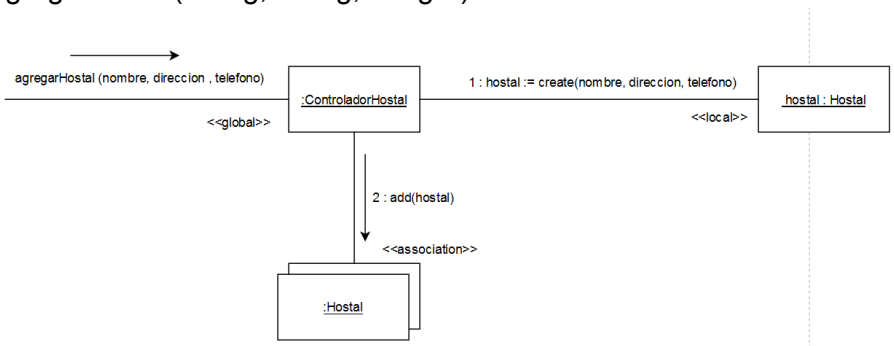


cancelarRegistroUsuario()



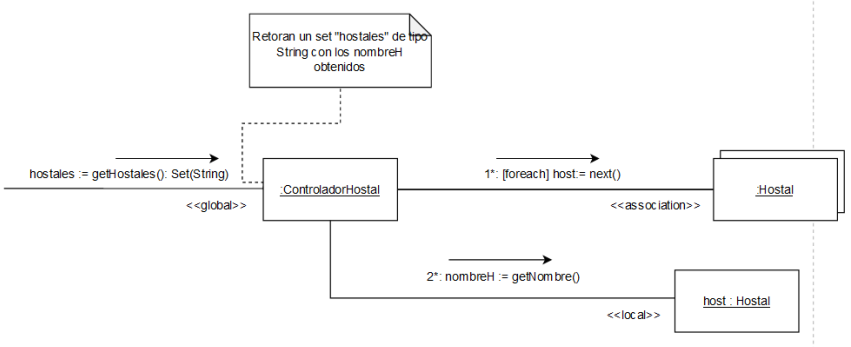
2.2 Alta de Hostal

agregarHostal(String, String, Integer)

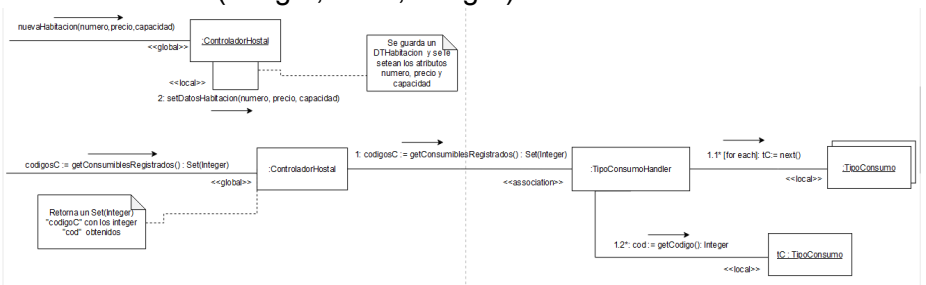


2.3 Alta de Habitación

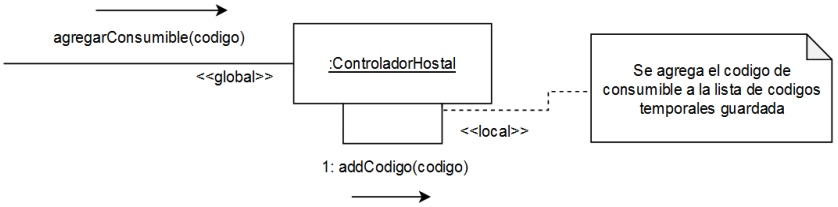
getHostales() : Set(String)



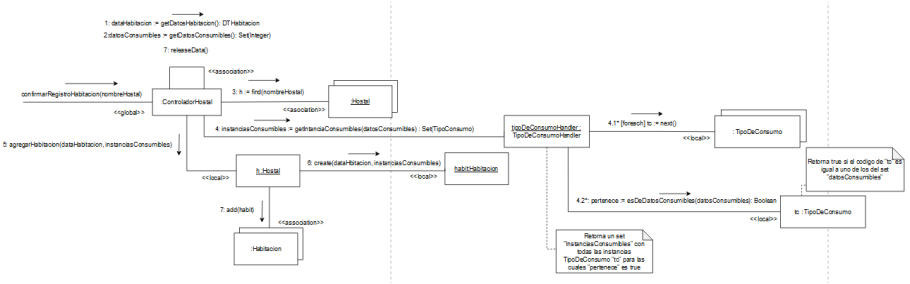
nuevaHabitacion(Integer, Float, Integer)



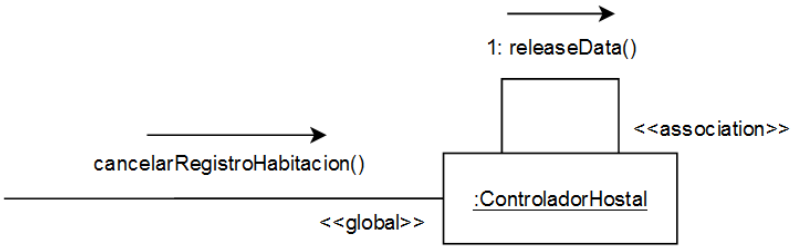
agregarConsumible(Integer)



confirmarRegistroHabitacion(String)

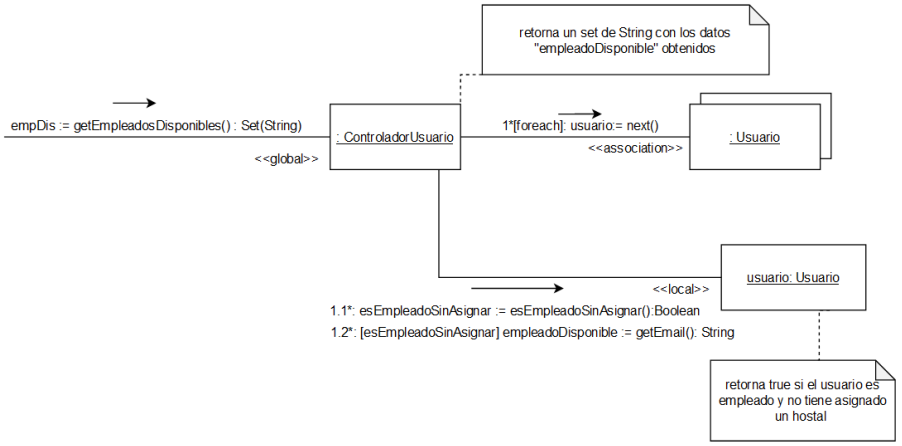


cancelarRegistroHabitacion()

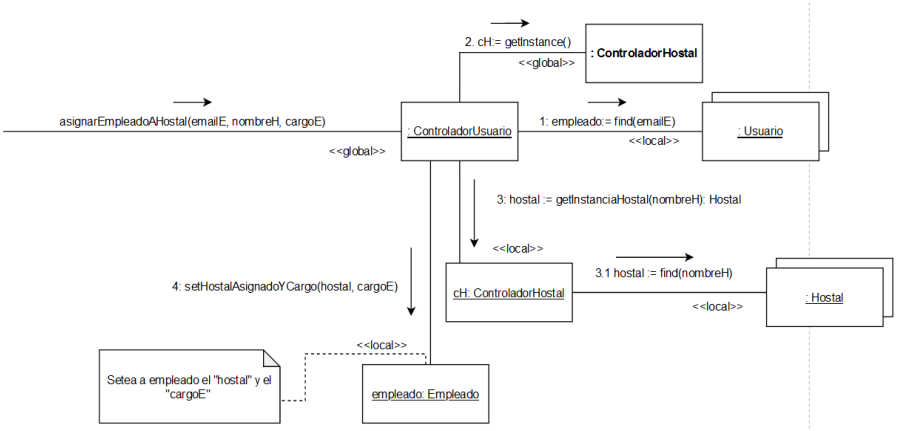


2.4 Asignar Empleado a Hostal

getEmpleadosDisponibles() : Set(String)

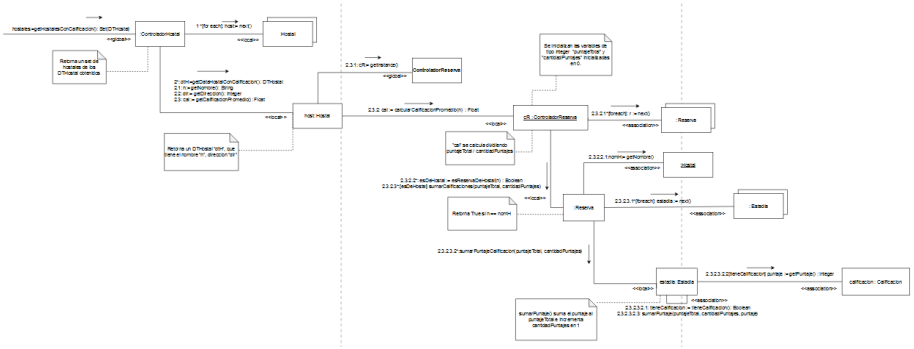


asignarEmpleadoAHostal(String, String, TipoCargo)

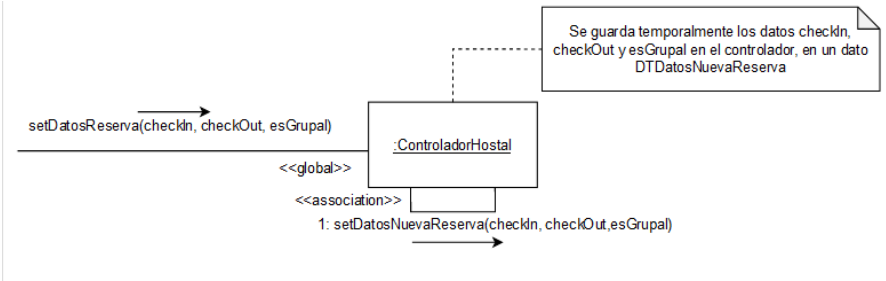


2.6 Realizar Reserva

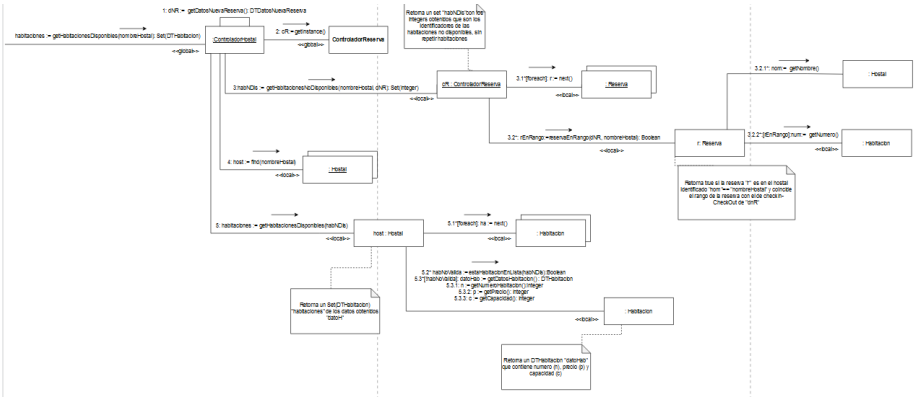
getHostalesConCalificacion(): Set(DTHostal)



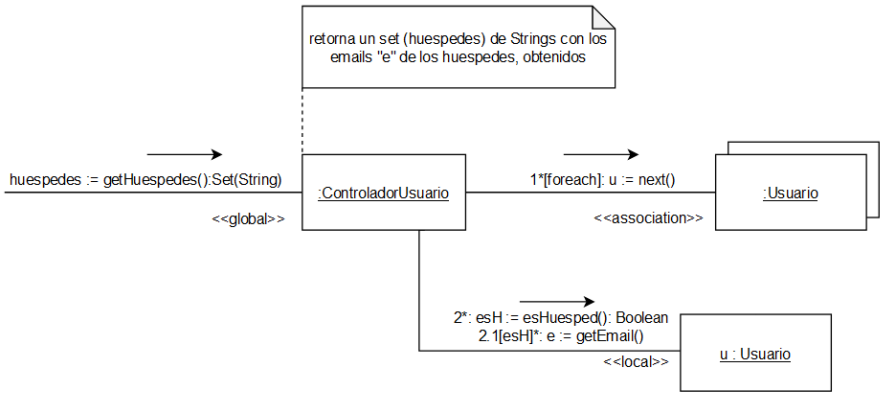
setDatosReserva(DTFecha, DTFecha, Boolean)



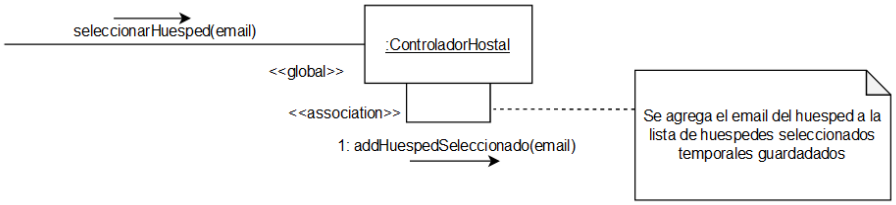
getHabitacionesDisponibles(String): Set(DTHabitacion)



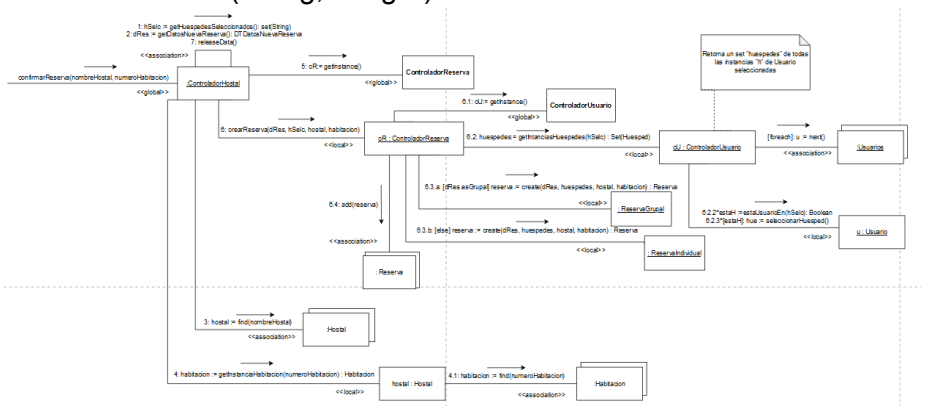
getHuespedes(): Set(String)



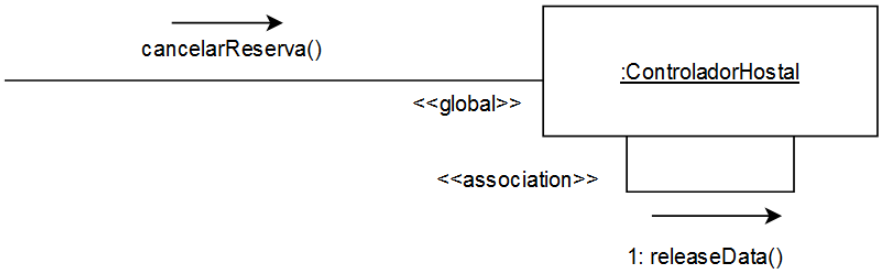
seleccionarHuesped(String)



confirmarReserva(String, Integer)

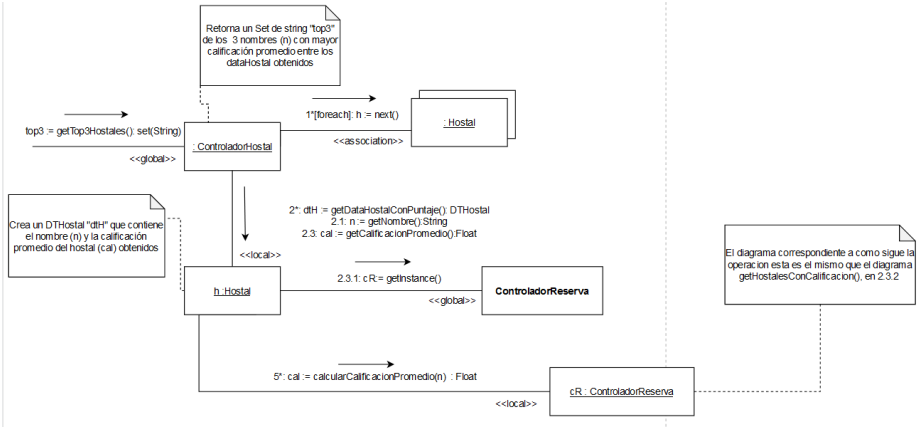


cancelarReserva()

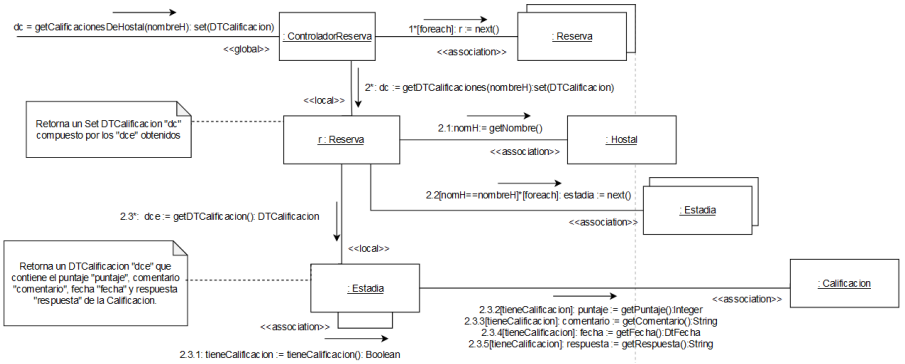


2.7 Consultar Top 3 Hostales

getTop3Hostales(): set(String)

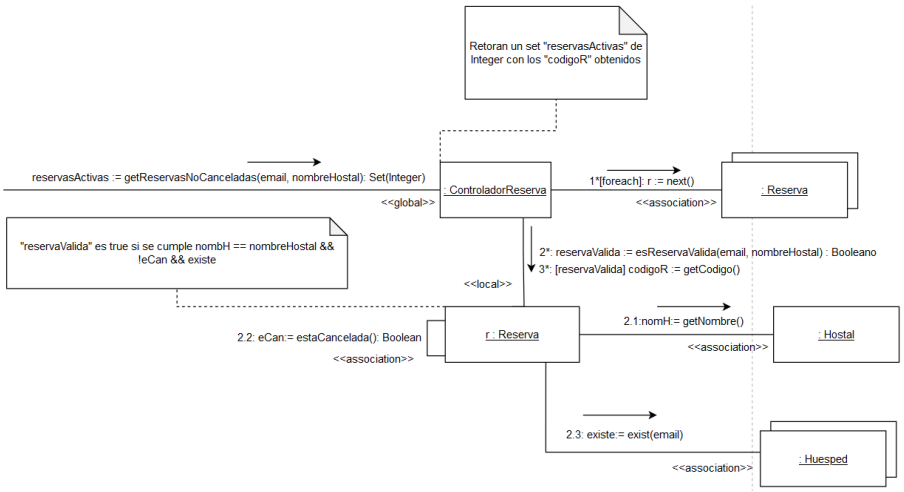


getCalificacionesDeHostal(nombreH): set(DTCalificacion)

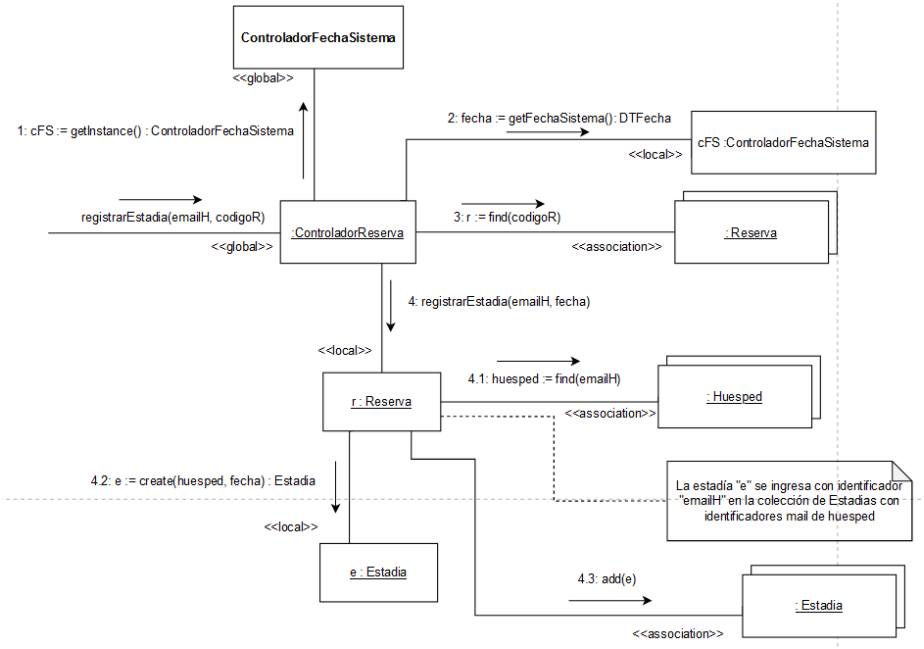


2.8 Registrar Estadia

getReservasNoCanceladas(String, String): Set(Integer)

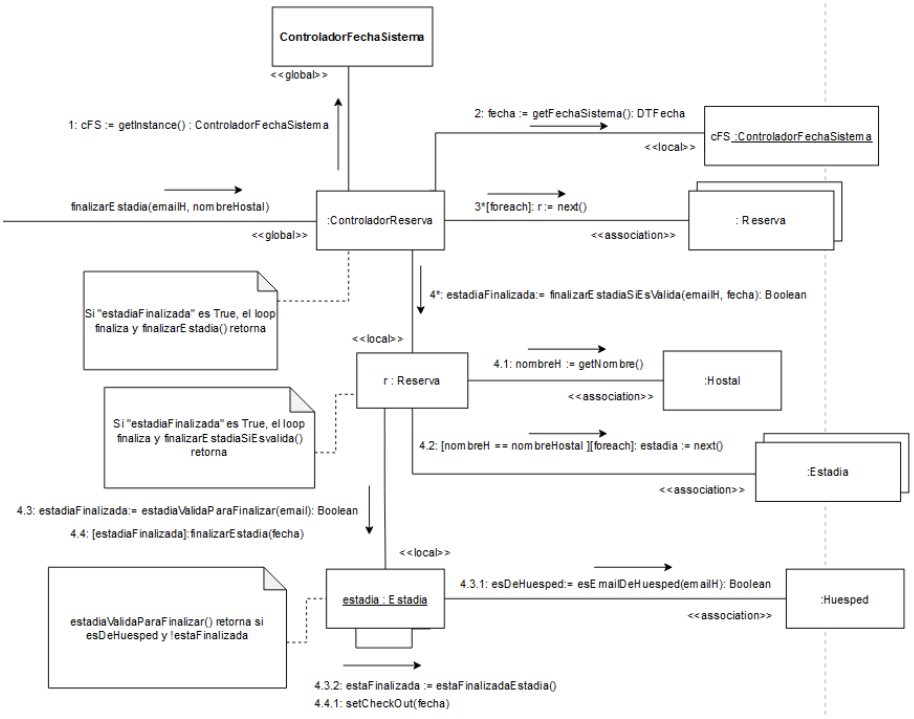


registrarEstadia(String, Integer)



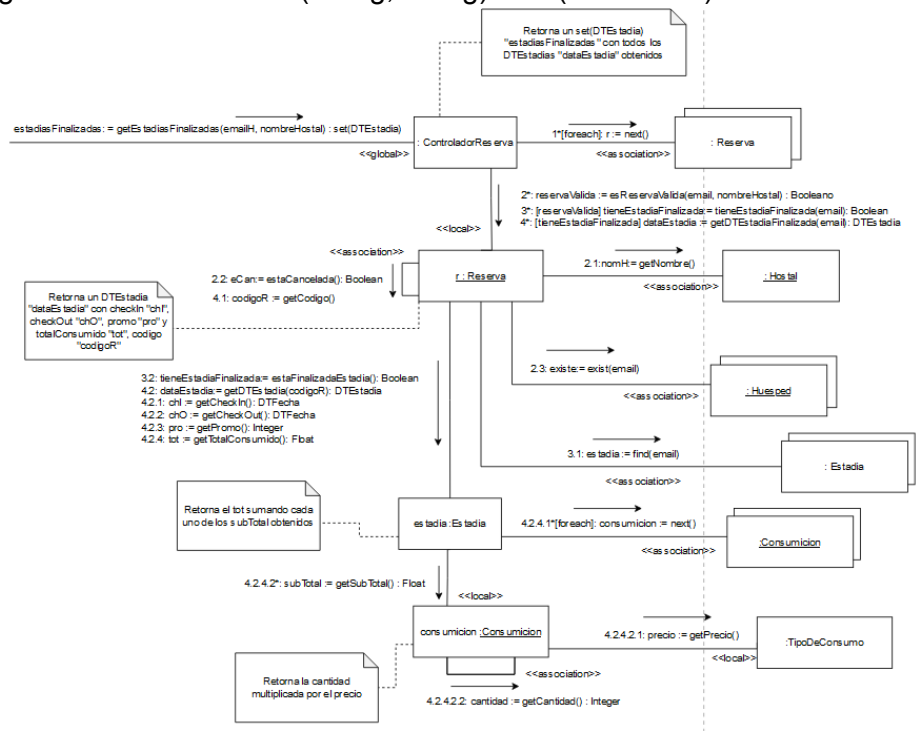
2.10 Finalizar Estadia

finalizarEstadia(String, String)

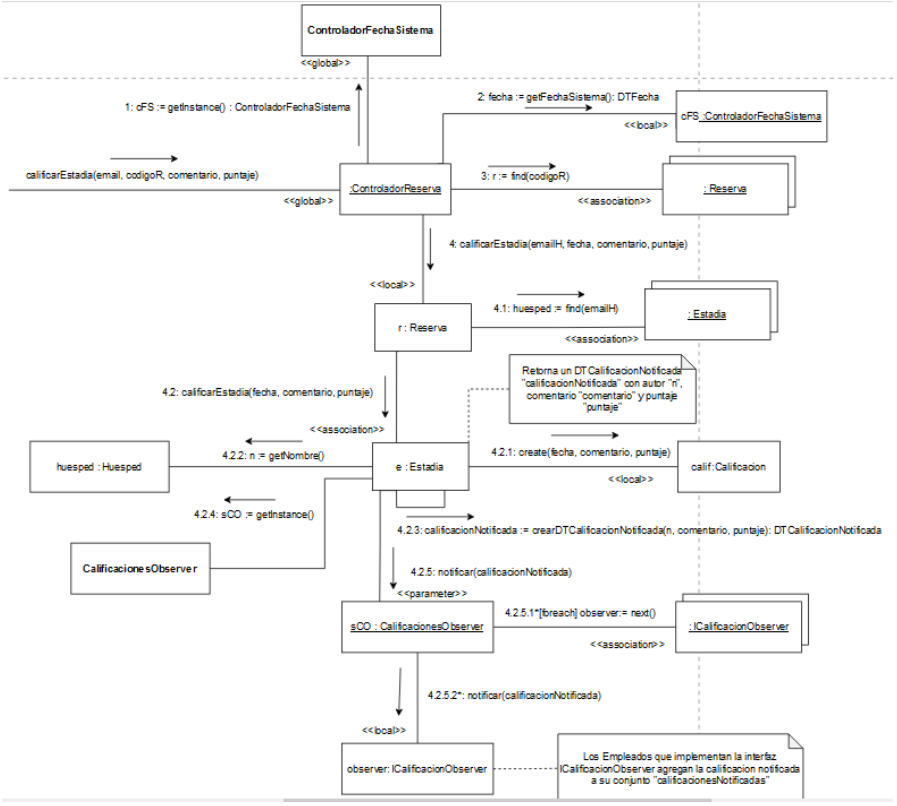


2.11 Calificar Estadia

getEstadiasFinalizadas(String, String) : Set(DTEstadia)

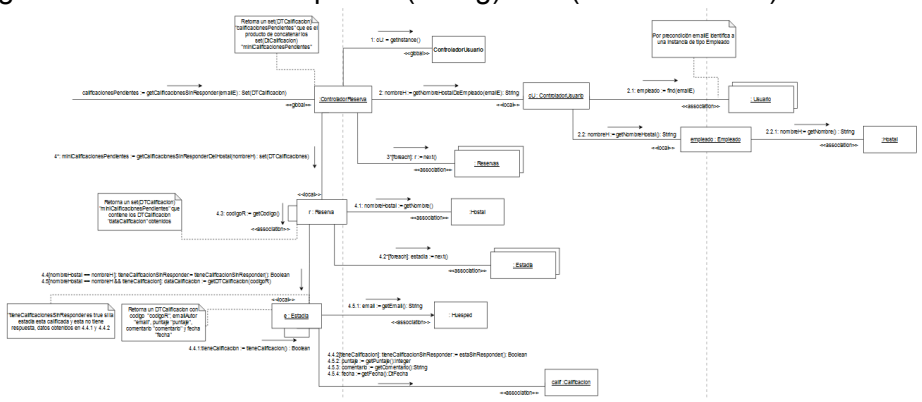


calificarEstadia(String, Integer, String, Integer)

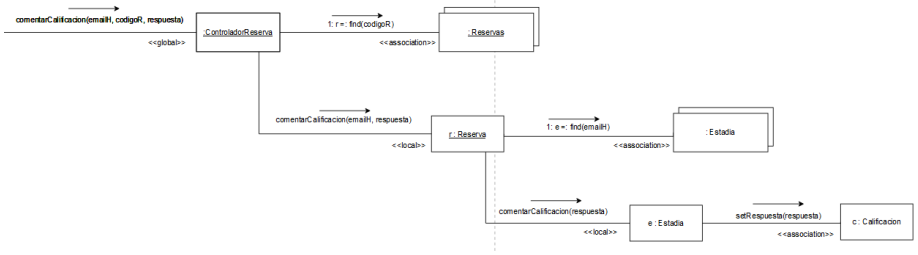


2.12 Comentar Calificación

getCalificacionesSinResponder(String) : Set(DTCalificacion)

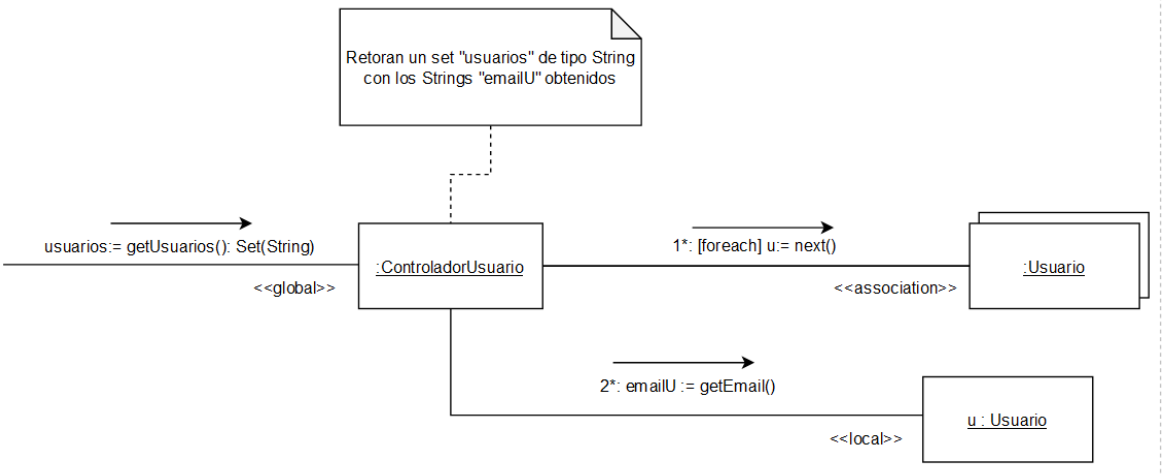


comentarCalificacion(String, Integer, String)

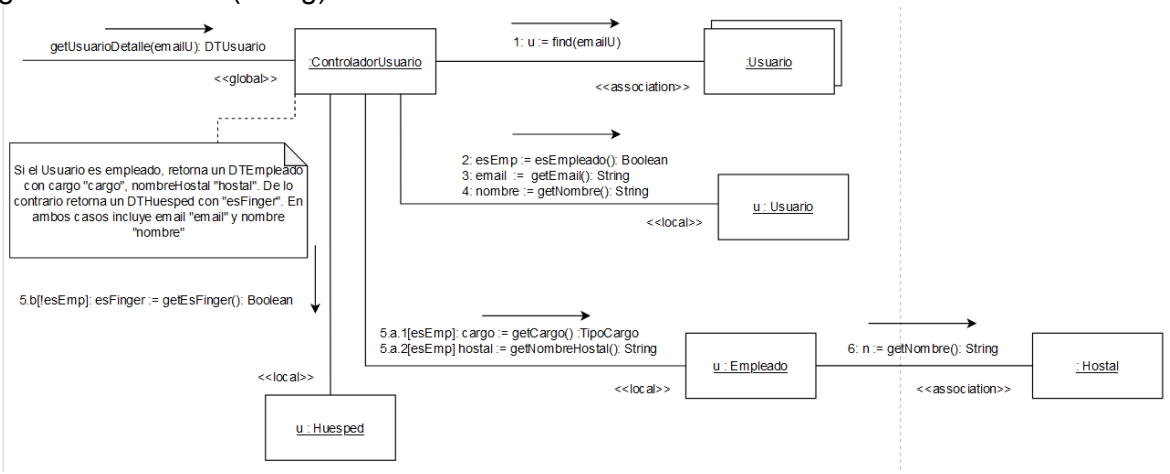


2.13 Consulta de Usuario

getUsuarios(): Set(String)

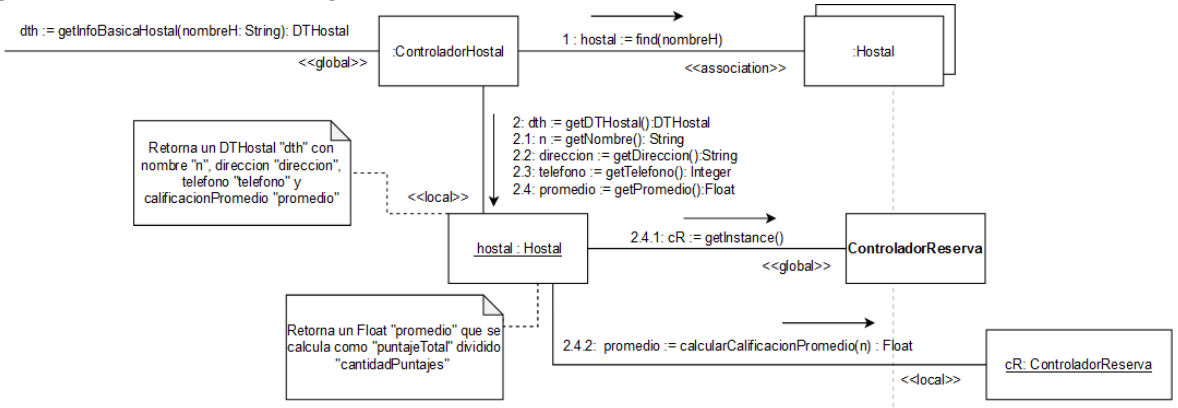


getUsuarioDetalle(String): DTUsuario

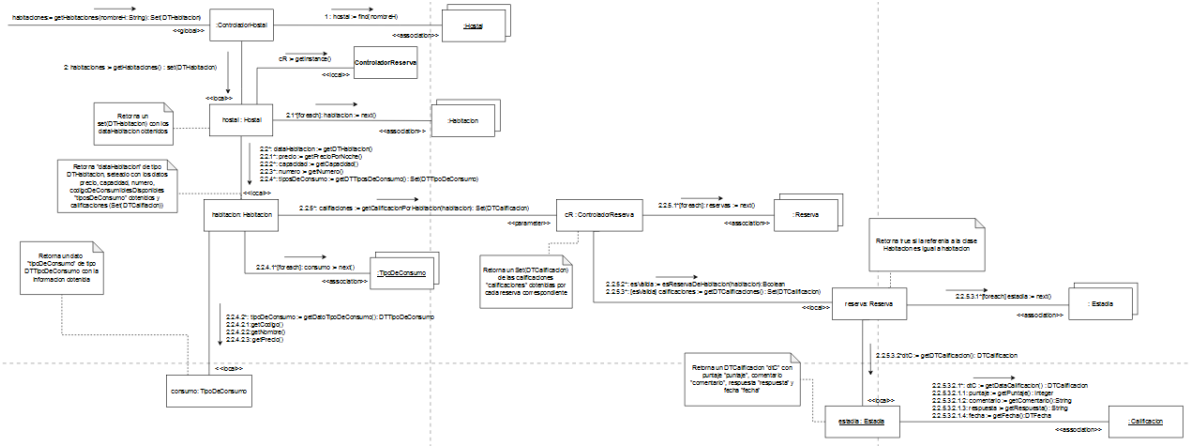


2.14 Consulta de Hostal

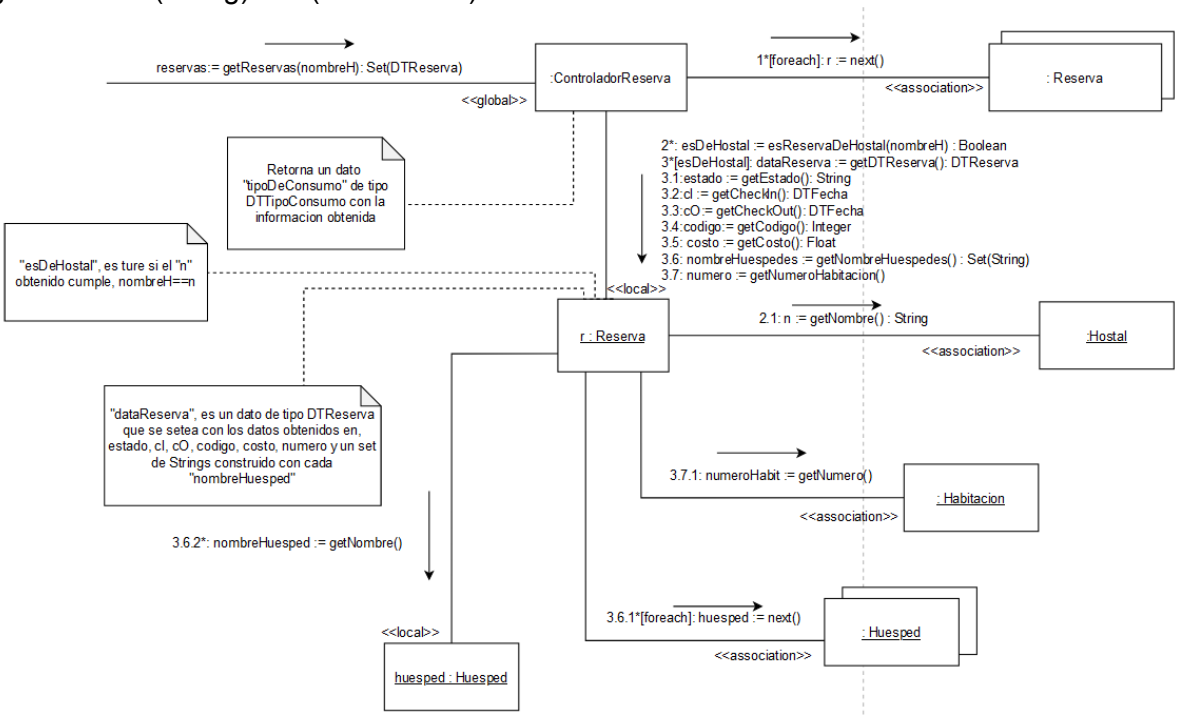
getInfoBasicaHostal(String): DTHostal



getHabitaciones(String): Set(DTHabitacion)



getReservas(String): Set(DTReserva)

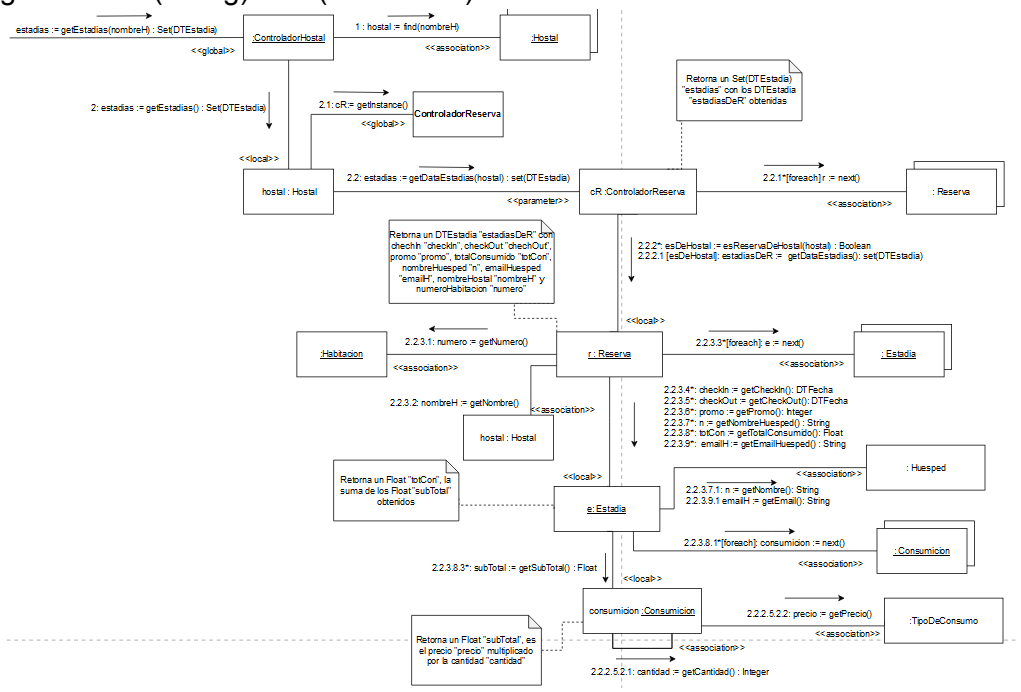


2.14 Consulta de Reserva

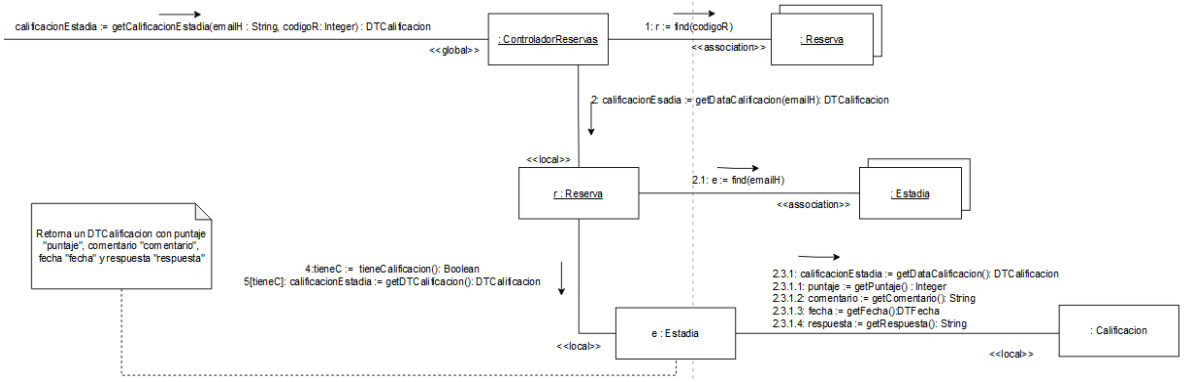
Los métodos usados en este caso de uso ya fueron planteados anteriormente en otros diagramas de comunicación. Estos son getHostales() : Set(String) que se puede encontrar en el caso de uso Alta de Habitación, y getReservas(nombreH): Set(DTReserva) que se puede observar en el caso de uso Consulta de Hostal.

2.16 Consulta de Estadia

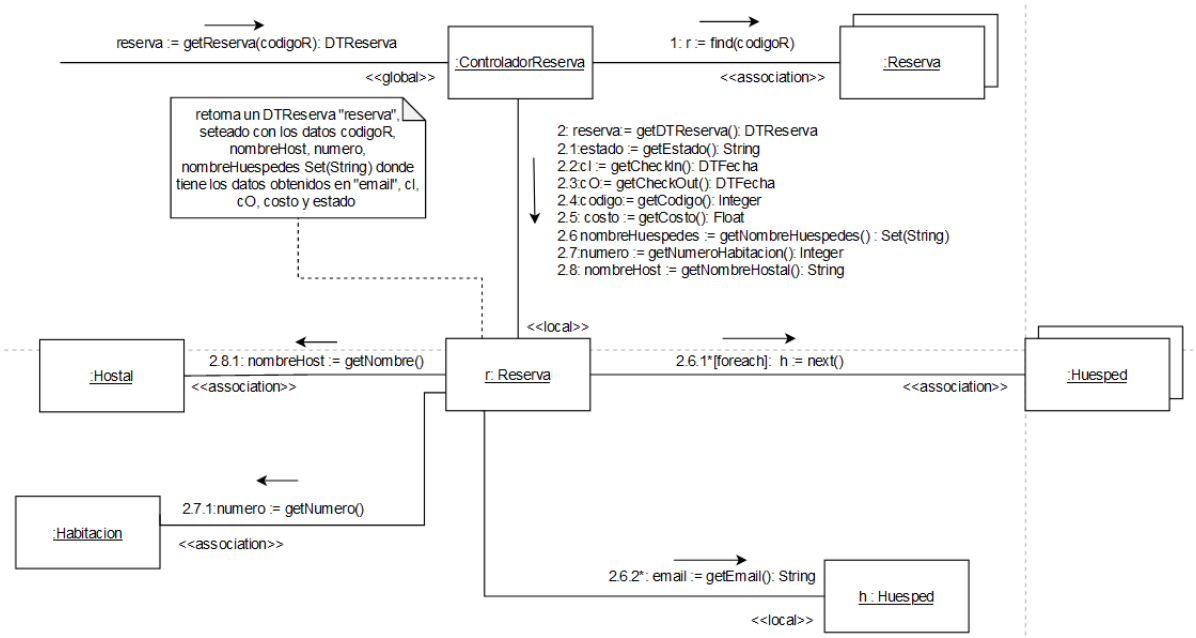
getEstadias(String) : Set(DTEstadia)



getCalificacionEstadia(String, Integer) : DTCalificacion

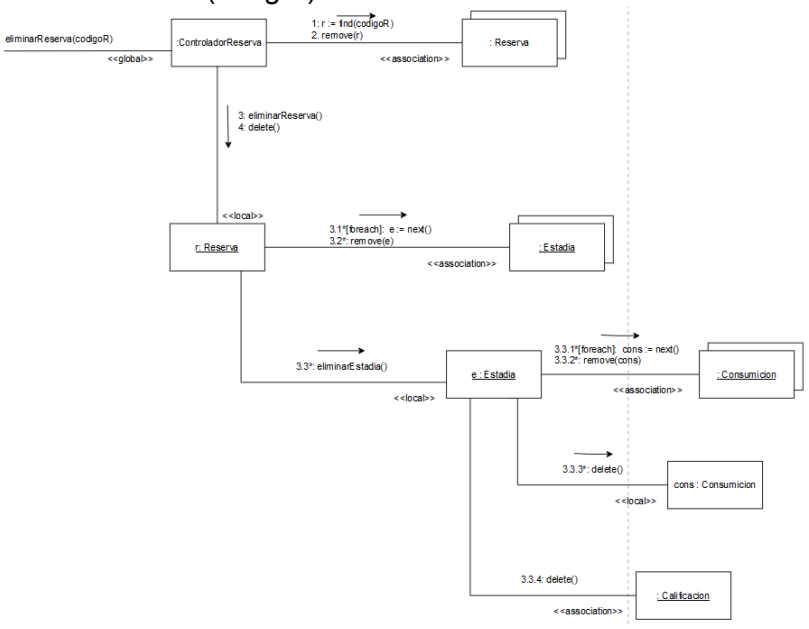


getReserva(Integer): DTReserva



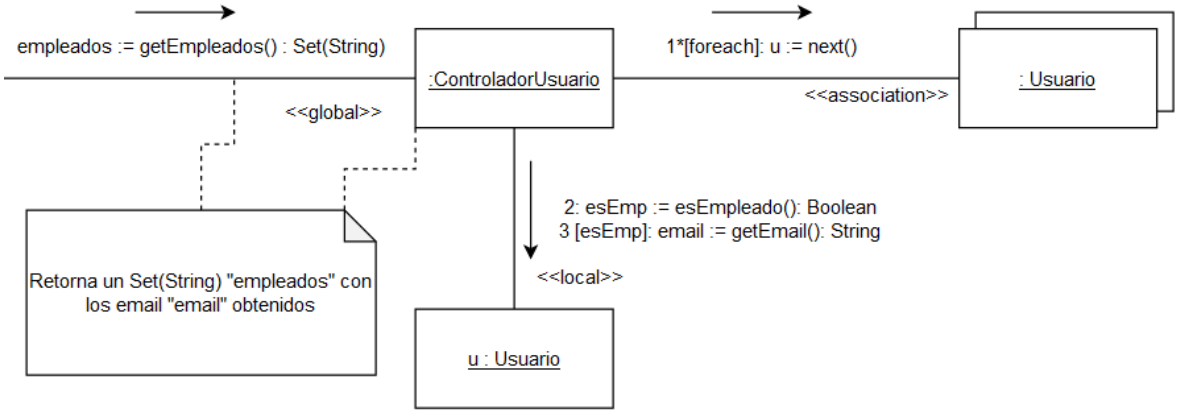
2.17 Baja de Reserva

eliminarReserva(Integer)

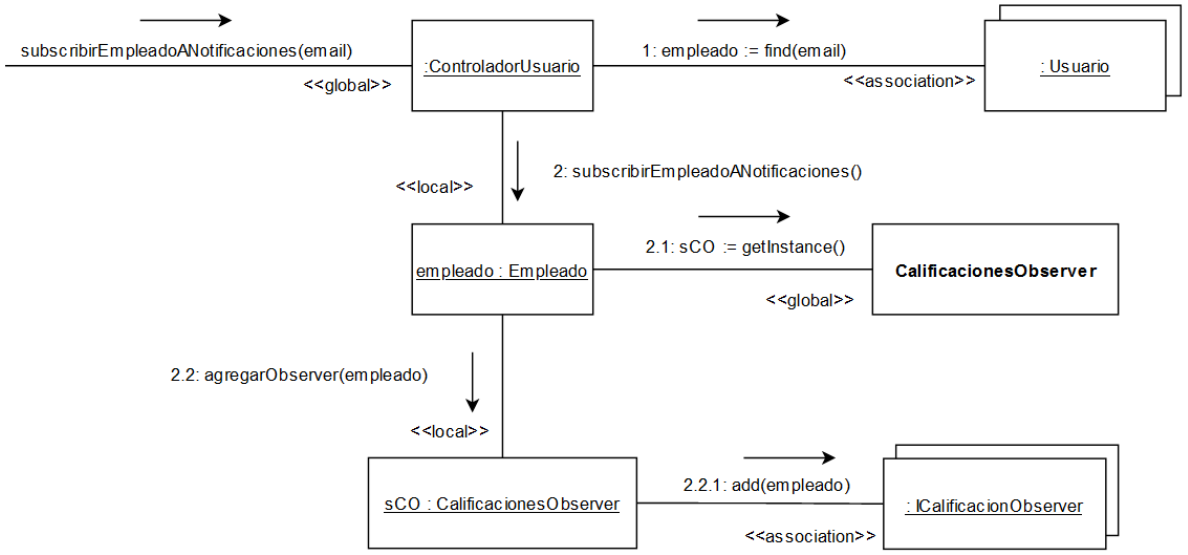


2.18 Suscribirse a Notificaciones

getEmpleados() : Set(String)

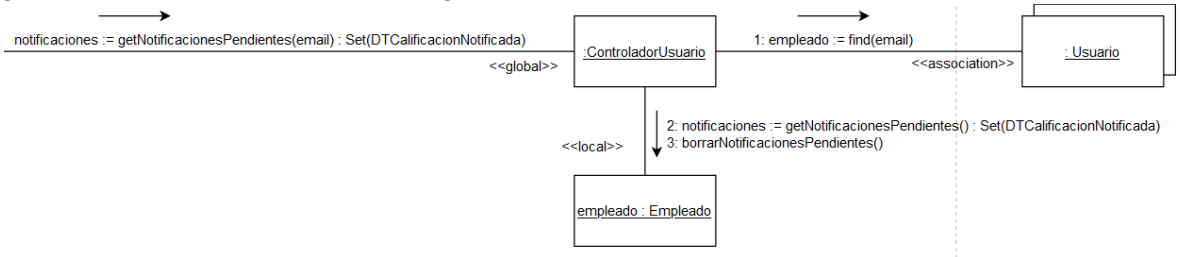


subscribirEmpleadoANotificaciones(String)



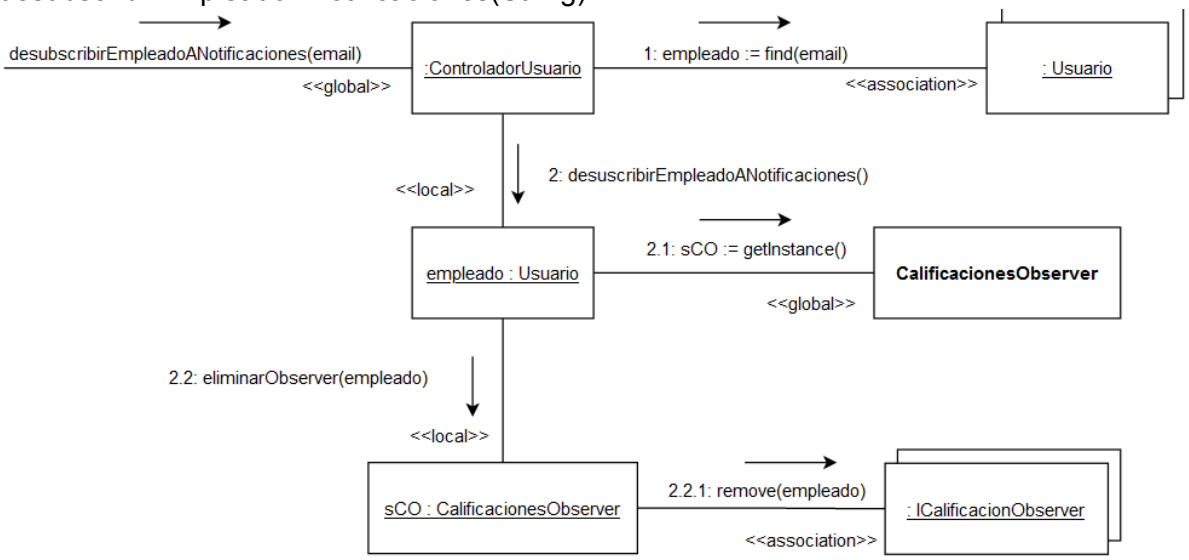
2.19 Consulta de Notificaciones

getNotificacionesPendientes(String) : Set(DTCalificacionNotificada)



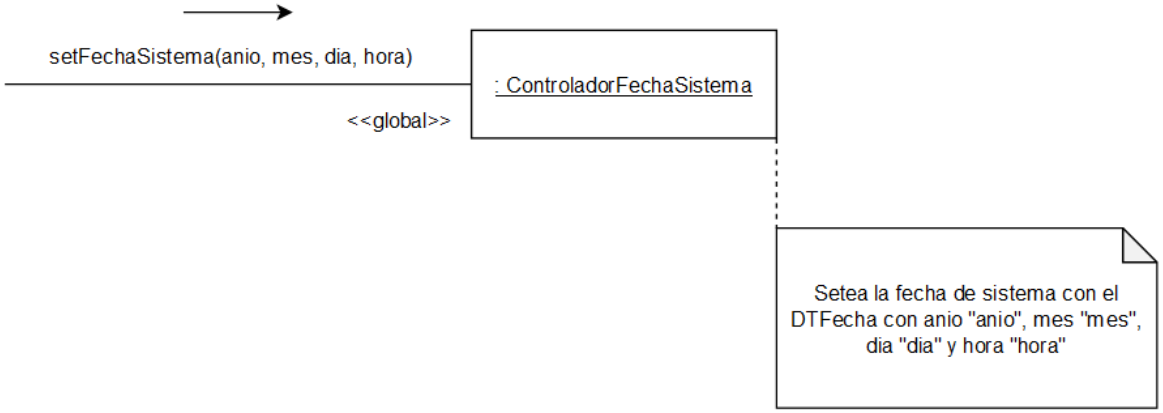
2.20 Eliminar Suscripción

desubscribirEmpleadoANotificaciones(String)

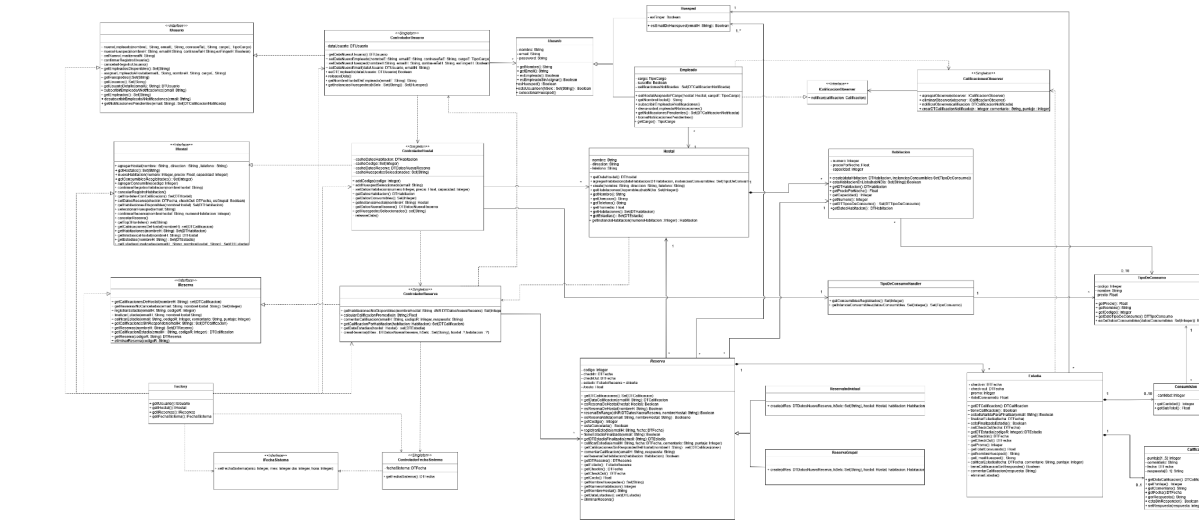


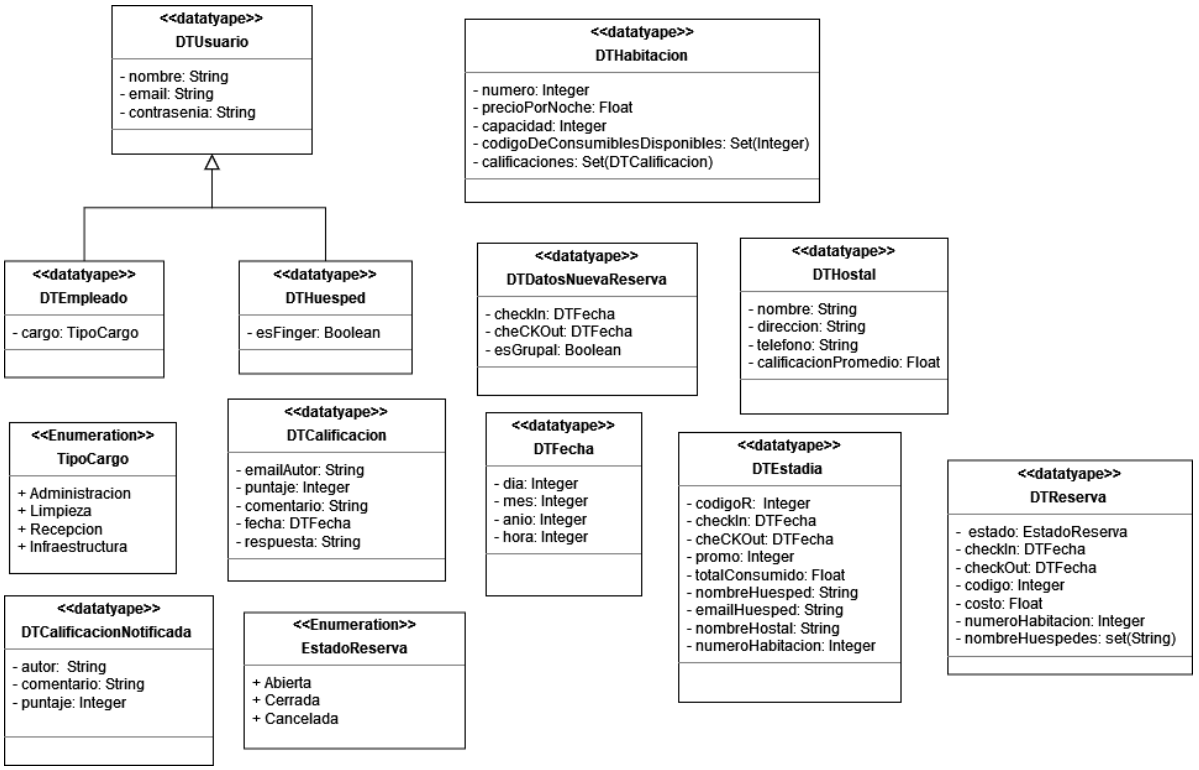
2.21 Modificar Fecha del Sistema

setFechaSistema(Integer, Integer, Integer, Integer)



3 Diagrama de Clases de Diseño





4 Criterios Generales

Para el diseño del sistema se utilizaron 4 interfaces, que especificaban las operaciones necesarias para que dicho sistema satisfaga los requerimientos impuestos por los casos de uso. Dichas interfaces fueron diseñadas contemplando los tres grandes aspectos de nuestro Sistema que son los hostales, los usuarios, las reservas y la fecha del sistema.

Para la implementación de dichas interfaces se diseñaron 4 controladores, respetando el criterio GRASP de controller. Guardamos la información pertinente necesaria para la implementación de las interfaces y derivamos las responsabilidades entre las demás clases del sistema para poder implementar las operaciones.

A su vez respetamos el criterio de expert al derivar dichas responsabilidades a las clases que conocen la información necesaria para su implementación.

Utilizamos el patrón de diseño Singleton para implementar dichos controladores para garantizar la existencia de una sola instancia. A su vez esto facilitó el acceso a dichos controladores desde todas las partes del sistema.

Se empleó el criterio de GRASP de fábrica para el desacoplamiento de las interfaces con los controladores que las implementan, haciendo de esta manera más mantenible el sistema en el tiempo, a su vez ayuda a desacoplar la capa visual de la lógica mediante las interfaces y la fábrica. Consideramos como mejora a futuro implementar una fábrica para la creación de los Usuarios.
El diseño también fue contemplado asegurándose de que las clases tengan bajo acoplamiento, tratando también de evitar relaciones dobles entre ellas.

Otro criterio considerado al momento de diseñar las comunicaciones fue aprovechar la transitividad de los objetos con el fin de mantener una baja cohesión. Al evitar relaciones directas entre objetos que no tienen visibilidad directa, cumplimos el criterio de no hables con extraños.
Para el requerimiento de que el sistema notifique a los empleados, se implementó el patrón de diseño observer. La clase Empleado participa como observador e implementa la interfaz “ICalificacionObserver”, mientras la estadía notifica cuando es calificada. Creamos la clase CalificacionesObserver que se encarga de agregar observadores, eliminarlos y realizar notificaciones (sujeto en el patrón observer).