**Programación 4**

Informe del Modelo de Diseño- Diagramas de Comunicación -

|  |  |  |
| --- | --- | --- |
| **Grupo 22** | | |
|  | | |
|  |  |  |
| Renzo Tissoni | renzo.tissoni@fing.edu.uy | **CI:** 5.460.797-9 |
| Juan Appoloni | juan.appoloni@fing.edu.uy | **CI:** 5.128.657-0 |
| Carolina Martínez | carolina.martinez@fing.edu.uy | **CI:** 5.245.351-8 |
| Rafael Silva | ramon.rafael.silva.ramirez@fing.edu.uy | **CI:** 3.268.024-0 |
| Felipe Miranda | felipe.miranda@fing.edu.uy | **CI:** 5.479.847-7 |

|  |  |
| --- | --- |
| Docente: | Daniel Calegari |

**ÍNDICE**

[Tabla de Figuras 3](#_Toc104759497)

[1 CAMBIOS GENERADOS POR LOS NUEVOS CASOS DE USO 4](#_Toc104759498)

[2 DIAGRAMAS DE COMUNICACIÓN 6](#_Toc104759499)

[Listar Notificaciones de un empleado 6](#_Toc104759500)

[Modificar Fecha del Sistema 7](#_Toc104759501)

[Suscribirse a una Notificación 8](#_Toc104759502)

[Eliminar Suscripción 9](#_Toc104759503)

[Alta de Usuario 10](#_Toc104759504)

[Alta de Hostal 11](#_Toc104759505)

[Alta de Habitación 12](#_Toc104759506)

[Asignar empleado a hostal 13](#_Toc104759507)

[Realizar Reserva 15](#_Toc104759508)

[Consultar top 3 de hostales 17](#_Toc104759509)

[Registrar estadía 18](#_Toc104759510)

[Finalizar Estadía 20](#_Toc104759511)

[Calificar Estadía 21](#_Toc104759512)

[Comentar Calificación 22](#_Toc104759513)

[Consulta de Usuario 23](#_Toc104759514)

[Consulta de Hostal 24](#_Toc104759515)

[Consulta de Reserva 25](#_Toc104759516)

[Consulta de Estadía 26](#_Toc104759517)

[Baja de reserva 28](#_Toc104759518)

[3 DIAGRAMA DE CLASES 30](#_Toc104759519)

[4 EXPLICACIÓN DEL DISEÑO DEL SISTEMA 31](#_Toc104759520)

# Tabla de Figuras

[Figura 1 Cambios en el Modelo de Dominio por agregar Notificación. 4](#_Toc104759474)

[Figura 2 Cambios en el DDS Calificar Estadía al agregar Notificación 5](#_Toc104759475)

[Figura 3 Listar Notificaciones de un empleado 6](#_Toc104759476)

[Figura 4 Listar Modificar fecha del sistema 7](#_Toc104759477)

[Figura 5 Suscribirse a una Notificación 8](#_Toc104759478)

[Figura 6 Eliminar suscripción 9](#_Toc104759479)

[Figura 7 Alta de Usuario 10](#_Toc104759480)

[Figura 8 Alta de Hostal 11](#_Toc104759481)

[Figura 9 Alta de Habitación 12](#_Toc104759482)

[Figura 10 Asignar empleado a hostal 14](#_Toc104759483)

[Figura 11 Realizar Reserva 16](#_Toc104759484)

[Figura 12 Consultar top 3 de hostales 17](#_Toc104759485)

[Figura 13 Registrar estadía 19](#_Toc104759486)

[Figura 14 Finalizar Estadía 20](#_Toc104759487)

[Figura 15 Calificar Estadía 21](#_Toc104759488)

[Figura 16 Comentar Calificación 22](#_Toc104759489)

[Figura 17 Consulta de Usuario 23](#_Toc104759490)

[Figura 18 Consulta de Hostal 24](#_Toc104759491)

[Figura 19 Consulta de Reserva 25](#_Toc104759492)

[Figura 20 Consulta de Estadía 27](#_Toc104759493)

[Figura 21 Baja de reserva 29](#_Toc104759494)

[Figura 22 Diagramas de Clases 30](#_Toc104759495)

[Figura 23 Estructura del patrón observer 32](#_Toc104759496)

# CAMBIOS GENERADOS POR LOS NUEVOS CASOS DE USO

A partir de los nuevos requerimientos es necesario agregar el concepto “Notificación” al modelo de dominio, el cual se relaciona de esta manera con el ya existente:

Diagram

Description automatically generated

Figura 1 Cambios en el Modelo de Dominio por agregar Notificación.

De esta forma cada empleado tiene su lista de notificaciones cada una correspondiente a una calificación, se nota que por cada calificación hay un conjunto de notificaciones, una por cada empleado suscripto.

También se producen cambios en el caso de uso Calificar estadía, pues cada vez que se agrega una calificación ahora también es necesario notificar a todos los empleados suscriptos, cosa que se logra al agregar al finar del DSS de dicho caso de uso la operación “NotificarNuevaCalificación()”, que se encarga de notificar a los empleados suscriptos creando para cada uno de estos una instancia de Notificación que es agregada a la lista de notificaciones de estos empleados, además de que cada una de estas notificaciones mantiene un link a la nueva calificación. El DSS resulta de la siguiente forma:

Diagram, table

Description automatically generated

Figura 2 Cambios en el DDS Calificar Estadía al agregar Notificación

# DIAGRAMAS DE COMUNICACIÓN

### Listar Notificaciones de un empleado

Graphical user interface

Description automatically generated

Figura 3 Listar Notificaciones de un empleado

### Modificar Fecha del Sistema

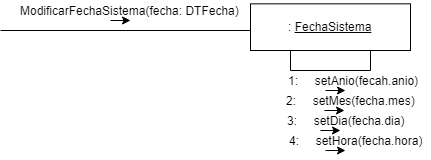


Figura 4 Listar Modificar fecha del sistema

### Suscribirse a una Notificación

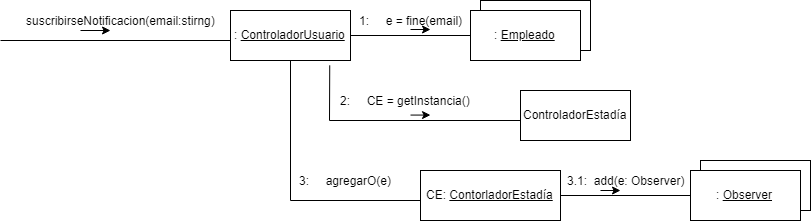


Figura 5 Suscribirse a una Notificación

### Eliminar Suscripción

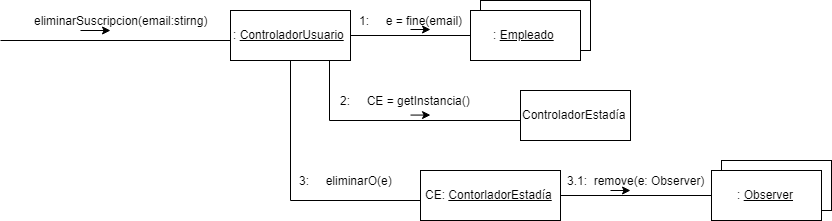


Figura 6 Eliminar suscripción

### Alta de Usuario

Diagram

Description automatically generated

Figura 7 Alta de Usuario

### Alta de Hostal

Diagram

Description automatically generated

Figura 8 Alta de Hostal

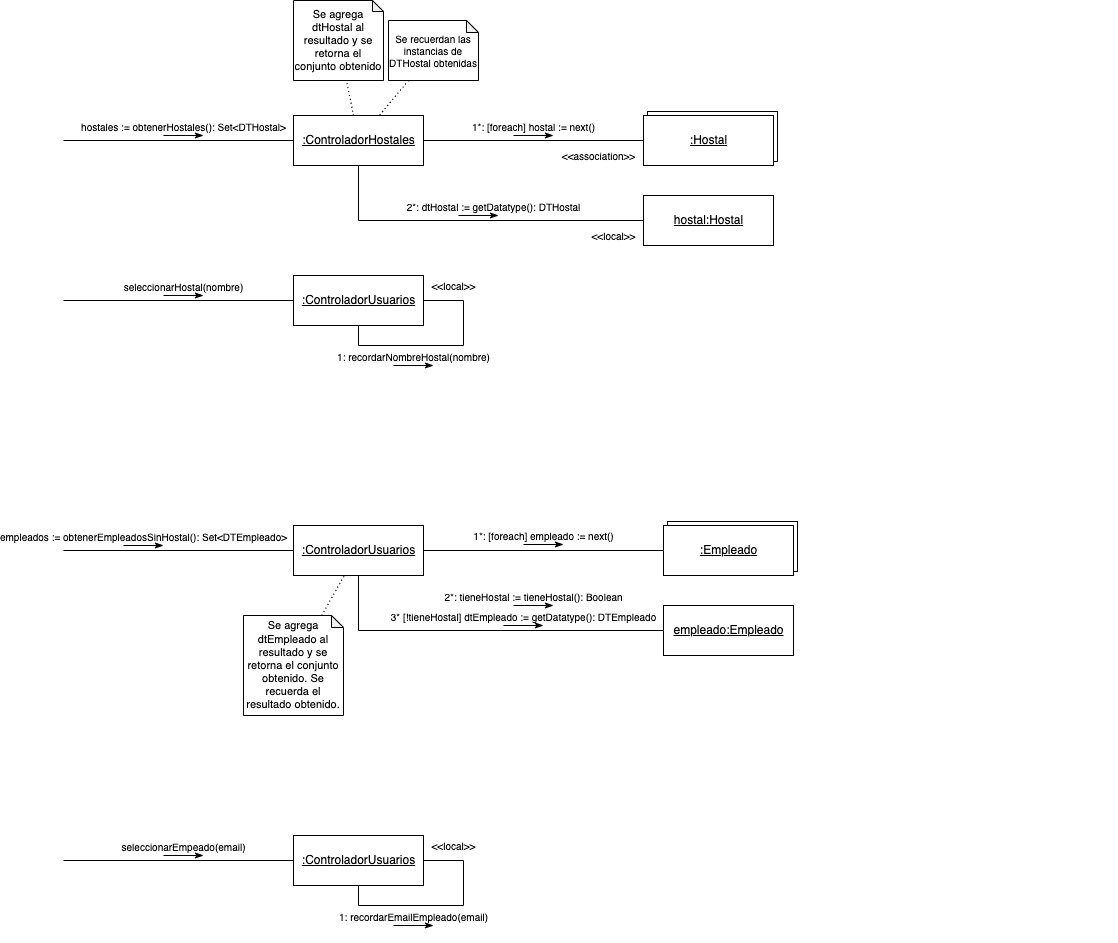
### Alta de Habitación

Diagram

Description automatically generated

Figura 9 Alta de Habitación

### Asignar empleado a hostal

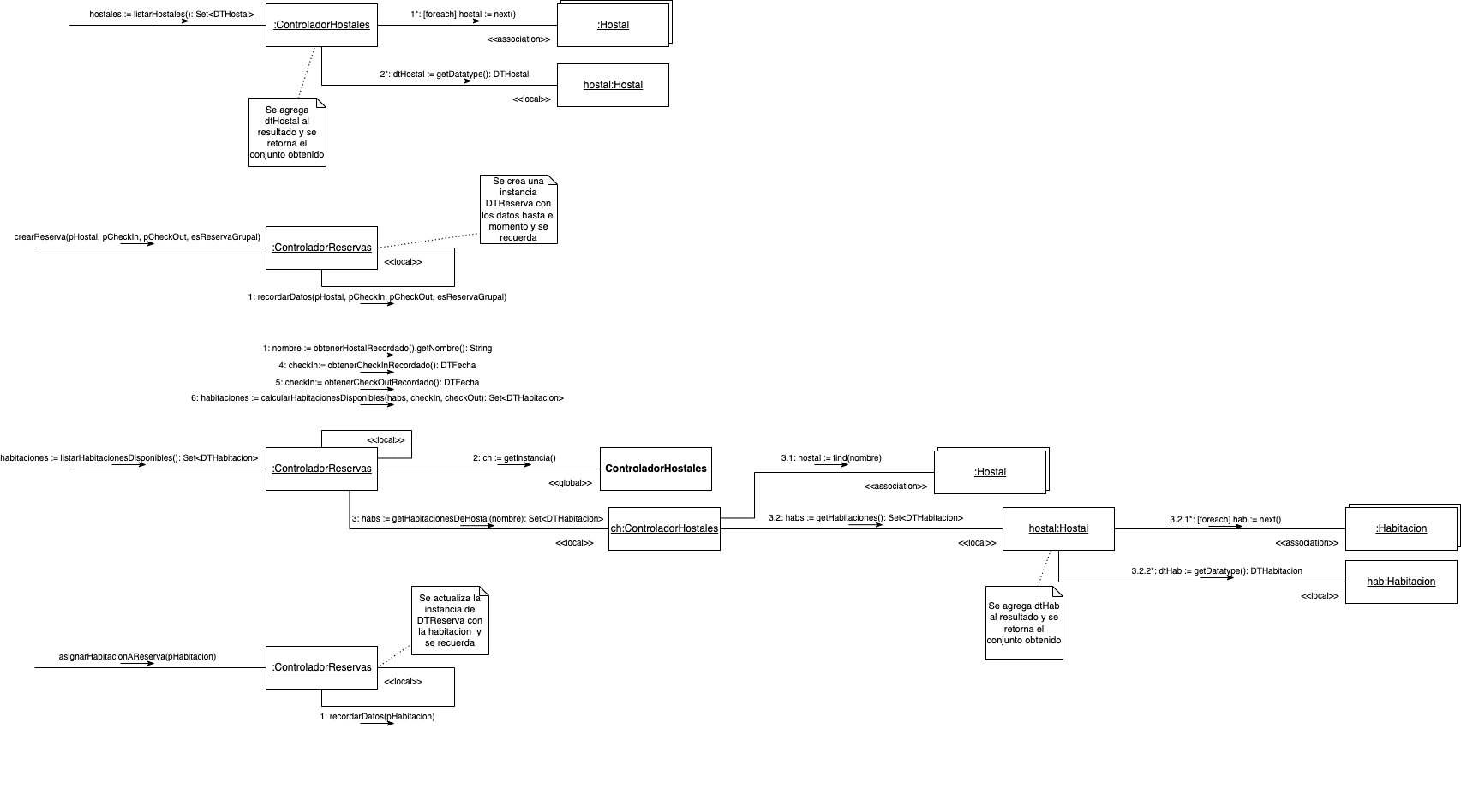


Diagram

Description automatically generated

Figura 10 Asignar empleado a hostal

### Realizar Reserva



Diagram, schematic

Description automatically generated

Figura 11 Realizar Reserva

### Consultar top 3 de hostales

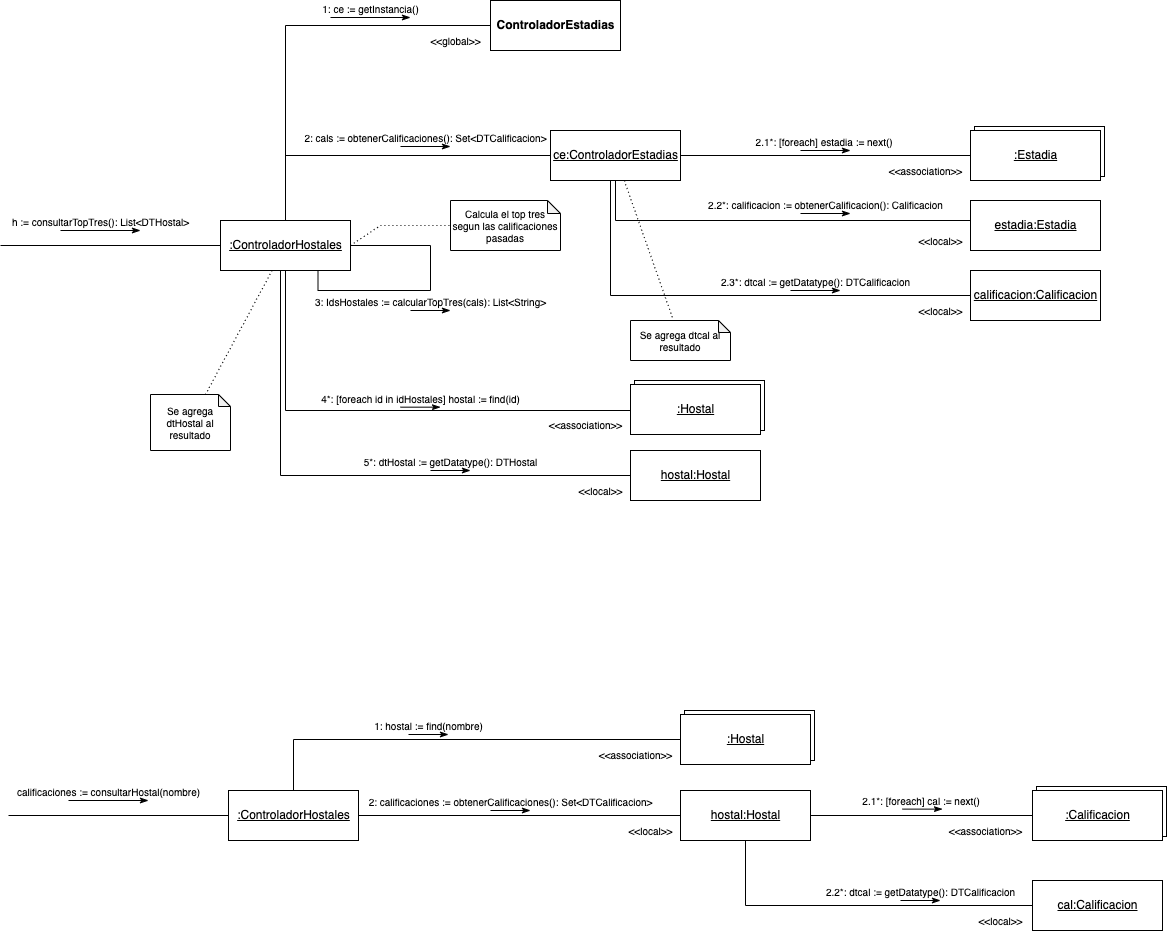


Figura 12 Consultar top 3 de hostales

### Registrar estadía

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Figura 13 Registrar estadía

### Finalizar Estadía

Diagram

Description automatically generated

Figura 14 Finalizar Estadía

### Calificar Estadía

Diagram, engineering drawing

Description automatically generated

Figura 15 Calificar Estadía

### Comentar Calificación

Diagram

Description automatically generated

Figura 16 Comentar Calificación

### Consulta de Usuario

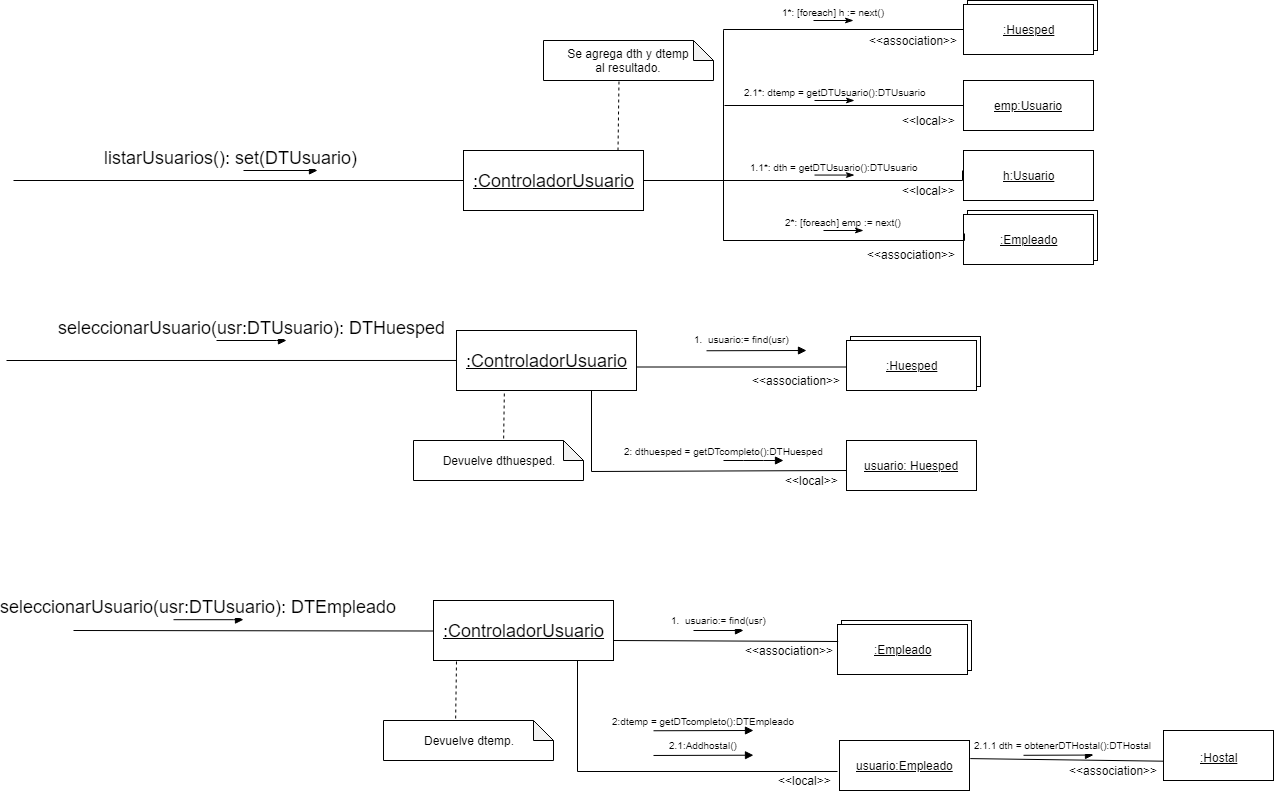


Figura 17 Consulta de Usuario

### Consulta de Hostal

Text, letter

Description automatically generated

Figura 18 Consulta de Hostal

### Consulta de Reserva

Diagram

Description automatically generated

Figura 19 Consulta de Reserva

### Consulta de Estadía

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Figura 20 Consulta de Estadía

### Baja de reserva

Diagram

Description automatically generated

Diagram

Description automatically generated

Figura 21 Baja de reserva

# DIAGRAMA DE CLASES

Diagram, schematic

Description automatically generated

Figura 22 Diagramas de Clases

# EXPLICACIÓN DEL DISEÑO DEL SISTEMA

En la solución descrita utilizamos varios controladores y cada controlador tiene una lista de objetos como Hostales, Estadías, Usuarios (Empleados y Huéspedes), etc., es necesario que estas no se dupliquen, es decir que tenga dos listas independientes en donde se podrían duplicar o habría inconsistencia de datos es decir que en una aparece un elemento y en la otra ese no existe. Por esto se decidió aplicar el patrón de diseño *Singleton* soluciona el problema planteado mediante la implementación de un método constructor que consiste en que si la controladora existe en el sistema nos la devuelve (con todas sus colecciones) y si no las crea (eso quiere decir que no se ha registrado nada en el sistema). Cuando surgió que el sistema tenía que manejar una fecha se decidió utilizar este patrón por lo mencionado anteriormente ya que cumple con todas las características mencionadas.

También se utilizan fabricas para romper la dependencia entre el cliente de una interfaz y quien la implementa. Para esto se cuenta con una fábrica por cada uno de los controladores del sistema que cumplen el rol de proveedor de la interfaz que cada uno implementa, dentro de la cual está la operación getControlador() que retorna la instancia del controlador en concreto.

Para resolver el nuevo requerimiento al envío de notificaciones a los empleados al agregar una calificación en el sistema, se implementa el patrón *Observer* en donde los Empleados cumplen el rol de observadores concretos, que dependen del controlador de estadía que cumple el rol de sujeto observado, el cual deberá notificar a sus observadores al agregar una nueva calificación. A estos efectos, los Empleados implementan la operación de la interfaz *Observer agregarNotificacion* que es la respuesta de los observados frente al evento descrito anteriormente. El controlador de Estadía mantiene una colección de observadores, los cuales serán avisados cuando un cliente agregue una nueva calificación. A pesar de que existen otras soluciones que no usan este patrón, esta es la más estable ante los posibles cambios que se realicen en el sistema, como agregar un nuevo tipo de observador, además de que se rompe la dependencia entre el controlador de Estadía y Empleados que se generaría en caso de que no interactuasen a través de la interfaz *observer* reduciendo el tiempo requerido para implementar esta funcionalidad. cambios del cliente. Para ilustrar la estructura de este patrón véase Figura 23 Estructura del patrón observer

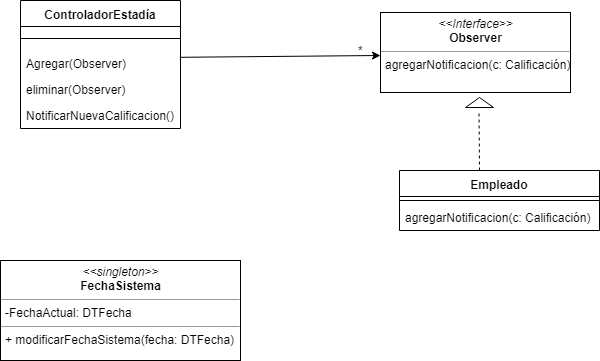


Figura 23 Estructura del patrón observer

Ahora puntualizaremos los criterios GRASP que utilizamos como solución del problema planteado.

* *CONTROLLER*

La estrategia que utilizamos para diseñar los Controladores es un híbrido entre Fachada y un controlador por CU. Agrupamos varios CU por Controlador tratando de que tengan una Alta Cohesión y de no asignarles demasiadas responsabilidades. Diseñamos 4 controladores que agrupan ciertos CU detallados a continuación:

|  |  |  |  |
| --- | --- | --- | --- |
| ControladorEstadía | ControladorUsuario | ControladorReserva | ControladorHostal |
| FinalizarEstadía  CalificarEstadía  ConsultaEstadía  RegistrarEstadía | **AltaUsuario**  **ConsultaUsuario**  **ComentarCalificación** | **BajaReserva**  **ConsultaReserva**  **RealizarReserva** | **AltaHostal**  **AltaHabitacion**  **AsignarEmpleadoHostal**  **ConsultaHostal**  **ConsultarTop3** |

Tabla 1 Controladores del Sistema

Cada controlador es responsable de mantener las colecciones de cada clase asignada. Por ejemplo, ControladorEstadía es responsable de mantener la colección Estadía ; ControladorReserva es responsable de mantener la colección Reserva. El caso del ControladorUsuario es un poco particular porque se mantiene una colección de Empleados y una colección de Huéspedes en el mismo controlador.

* *EXPERT*

En gran medida aplicamos *expert* ya que es un criterio bastante genérico, así conservamos el encapsulamiento, debido a que los objetos se valen de información propia para hacer lo que se les pide. Esto soporta un Bajo Acoplamiento y una Alta Cohesión.

Por ejemplo, en la operación buscarComentario():DTComentario del CU ConsultaDeEstadía. El ControladorEstadía delega la responsabilidad a Estadía (aplico *Expert*), ya que este es el experto en información y conoce el comentario de la calificación por estar vinculado con Calificación. Ahora delego la responsabilidad a Calificación (aplico *Expert*) porque este es el experto en información y conoce la respuesta del comentario ya que está vinculado con Comentario. Ahora delego la responsabilidad a Comentario (aplico *Expert*), ya que este es el experto en la información que necesito para los requerimientos de la operación.

* *CREATOR*

Para aplicar este criterio asumimos que la clase creadora tiene una responsabilidad muy fuerte para instanciar a la otra clase, ya sea que está agregada, contenida, registra instancias de ella, utiliza objetos en forma exclusiva o es el Experto en crear instancias. Por ejemplo, en el diagrama de comunicación Comentar Calificación, la operación de responder Calificación.

Es razonable que por *Creator* se decida que la clase Empleado es responsable de crear instancias de Comentario, ya que la clase Empleado mantiene una asociación con la clase Comentario, un empleado puede hacer varios comentarios. Y también Empleado tiene los datos de inicialización que serán transmitidos a Comentario cuando este objeto sea creado, así que Empleado es un Experto respecto a la creación de Comentario.

* NHCE

En líneas generales aplicamos NHCE asignando responsabilidades y tratando de no ganar visibilidad sobre un objeto “indirecto” para no quedar acoplado a este. Este criterio se viola cuando un Controlador necesita acceder a otro Controlador, de esa manera ganamos visibilidad sobre un elemento que no conocemos. El motivo de dicha violación es que necesitamos información en particular que la puede brindar otro Controlador. Un ejemplo de NHCE en el Diag. de comunicación de RegistrarEstadía es la operación de obtenerHostales cuando el ControladorEstadía le pide información al ControladorHostal.

* BAJO ACOPLAMIENTO

En su mayoría aplicamos este criterio, tratando de que una clase dependa de pocas clases, y que así sea fácil de comprender y reutilizar. A partir de lo anterior, nos guiamos por las asociaciones que ya estaban en el Modelo de Dominio para evitar agregar nuevas relaciones y dependencias.

Los Controladores son una excepción a este criterio y mantienen un Alto Acoplamiento. Como se refleja en el DCD:

* ControladorHostales está acoplado con ControladorEstadías.
* ControladorEstadías está acoplado con ControladorHostales, con ControladorUsuarios y con ControladorReservas.
* ControladorUsuarios está acoplado con ControladorEstadías.
* ControladorReservas está acoplado con ControladorHostales y con ControladorUsuarios.

Por ejemplo, ControladorReservas está acoplado con ControladorHostales porque en la operación listarHostales() el ControladorReservas le delega la responsabilidad a ControladorHostales de dicha operación.

* ALTA COHESION

En el diseño de Controladores aplicamos este criterio, buscamos que tengan un número relativamente pequeño de operaciones y que sus funcionalidades estén muy relacionadas.