HOTEL ZARGO

MEMORIA DEL PROYECTO

Cerdá Sánchez, Ignacio
Clemente Montero, Noel
Jimeno Garrachón, Gorka
Olivera Zaldua, Pablo
Sáez Hernando, Álvaro
Valero Martín, Luis

27 Mayo de 2013

1. Introducción:

Hotel Zargo es un proyecto para la asignatura de ingeniería del software, el cual trata la gestión y organización de un software basado en la gestión de un hotel.

Tendremos dos tipos de usuarios: Administración y servicios. Ambos podrán interaccionar con el software pero con cometidos diferentes:

- Los administradores, tendrán control total del programa.
- Los empleados de servicio, interaccionaran con el software para visualizar contenidos autorizados.

2. Modelo del dominio:

Nuestro modelo del dominio consta de los siguientes módulos:

1 Reservas

1.1: Alta de reserva
1.2: Baja de reserva
1.3: Modificar reserva
1.4: Listar reservas
1.5: Buscar disponibilidad
1.6: Hacer efectiva

2 Habitaciones

2.1: Alta de habitación
2.2: Baja de habitación
2.3: Modificar habitación
2.4: Listar habitaciones

3 Empleado

3.1: Alta de empleado
3.2: Baja de empleado
3.3: Modificar empleado
3.4: Listar empleados

4 Cliente

4.1: Alta de cliente
4.2: Baja de cliente

4.3: Modificar cliente
4.4: Listar clientes

5 Turno

5.1: Alta de turno
5.2: Baja de turno
5.3: Modificar turno
5.4: Listar turnos

6 Servicios del hotel

6.1: Alta de servicio hotel
6.2: Baja de servicio hotel
6.3: Modificar servicio hotel
6.4: Listar servicios hotel

3. Arquitectura:

Para el diseño de la aplicación y su implementación hemos utilizado una arquitectura multicapa dividida en:

Integración: Se ocupa de la gestión y comunicación del código con la base de datos.

Negocio: Consta de los transfer y el servicio de aplicación.

Presentación: Diseña e implementa la interfaz gráfica de la aplicación y gracias a ella la del usuario con el sistema.

4. Patrones:

En la implementación del software hemos utilizado los siguiente patrones:

MVC: El Modelo Vista Controlador es un patrón de arquitectura de software que separa los datos y la lógica de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Singleton: El patrón de diseño singleton está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

Façade: El patrón de diseño Façade conoce qué clases del subsistema son responsables de una determinada petición, y delega esas peticiones de los clientes a los objetos apropiados del subsistema.

Abstract Factory: Es la definición de la interfaces de las factorías. Debe de proveer un método para la obtención de cada objeto que pueda crear.

Command: Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma. Para ello se encapsula la petición como un objeto, con lo que además se facilita la parametrización de los métodos.

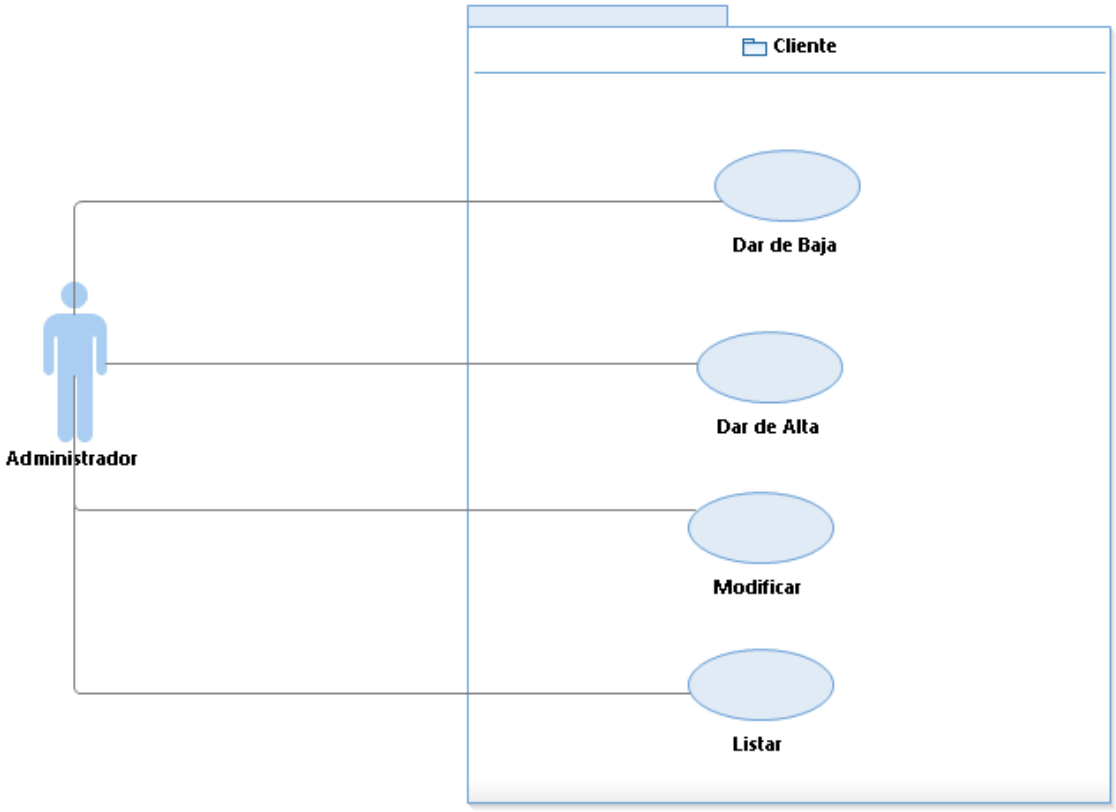
5. Modelos y diagramas:

Para representar la interacción entre las distintas capas hemos utilizado una serie de diagramas.

5.1 Diagramas de casos de uso:

Estos diagramas representan la interacción del usuario con los distintos módulos del programa.

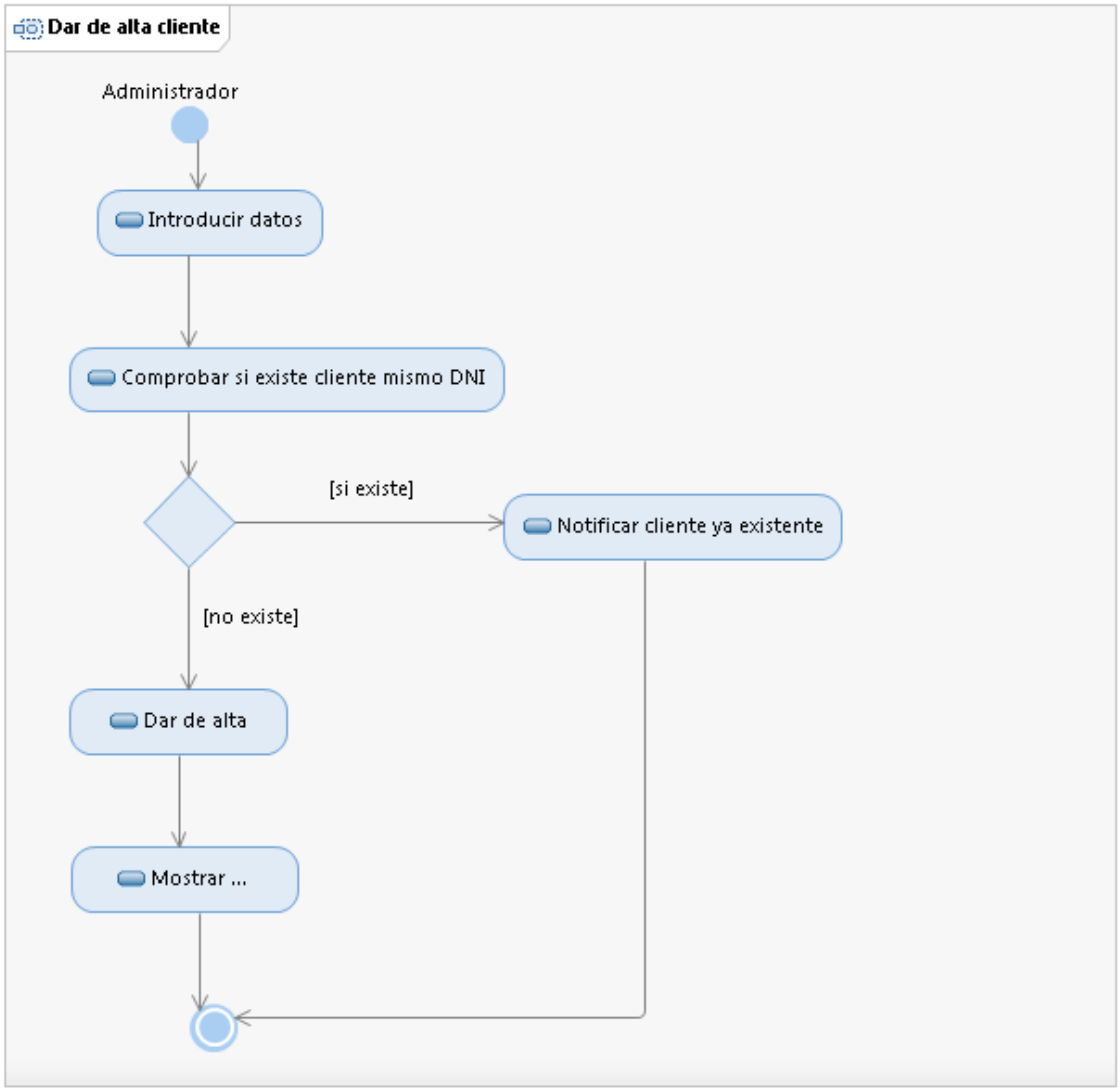
Adjuntamos el ejemplo del módulo cliente.



5.2 Diagramas de actividad:

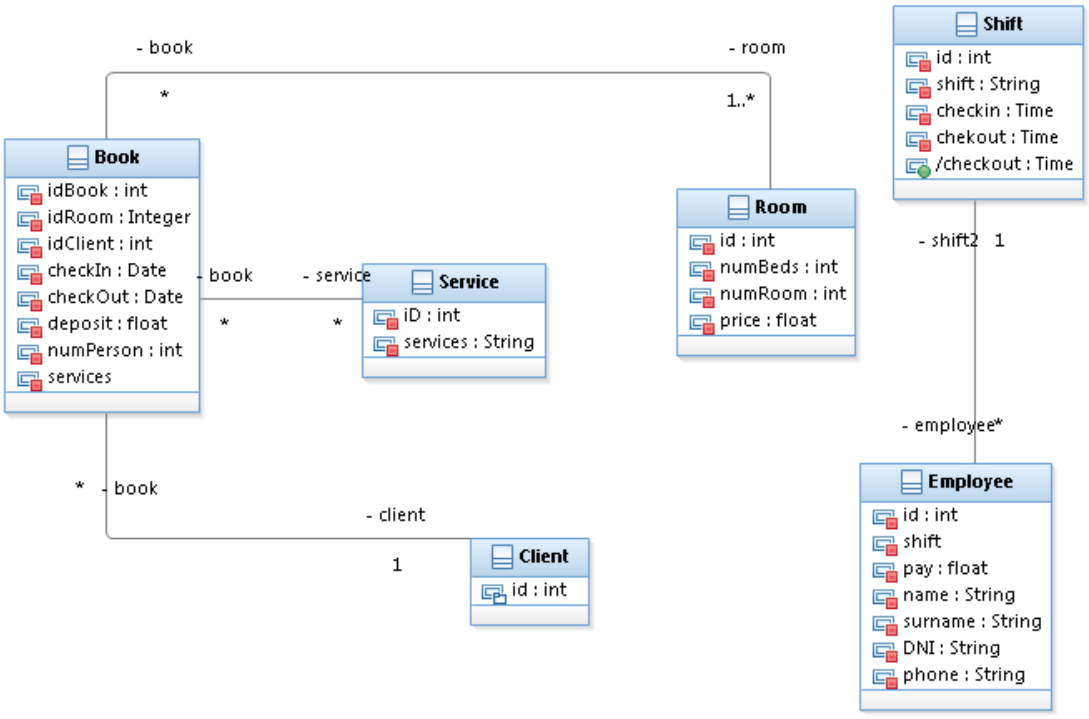
Representan las distintas funciones que desarrolla el software, desde que el usuario realiza una acción hasta que el sistema le devuelve le responde.

Adjuntamos el ejemplo de dar de alta cliente.



5.3 Diagrama de modelo del dominio:

Muestra los módulos principales de la aplicación y su relación entre ellos.

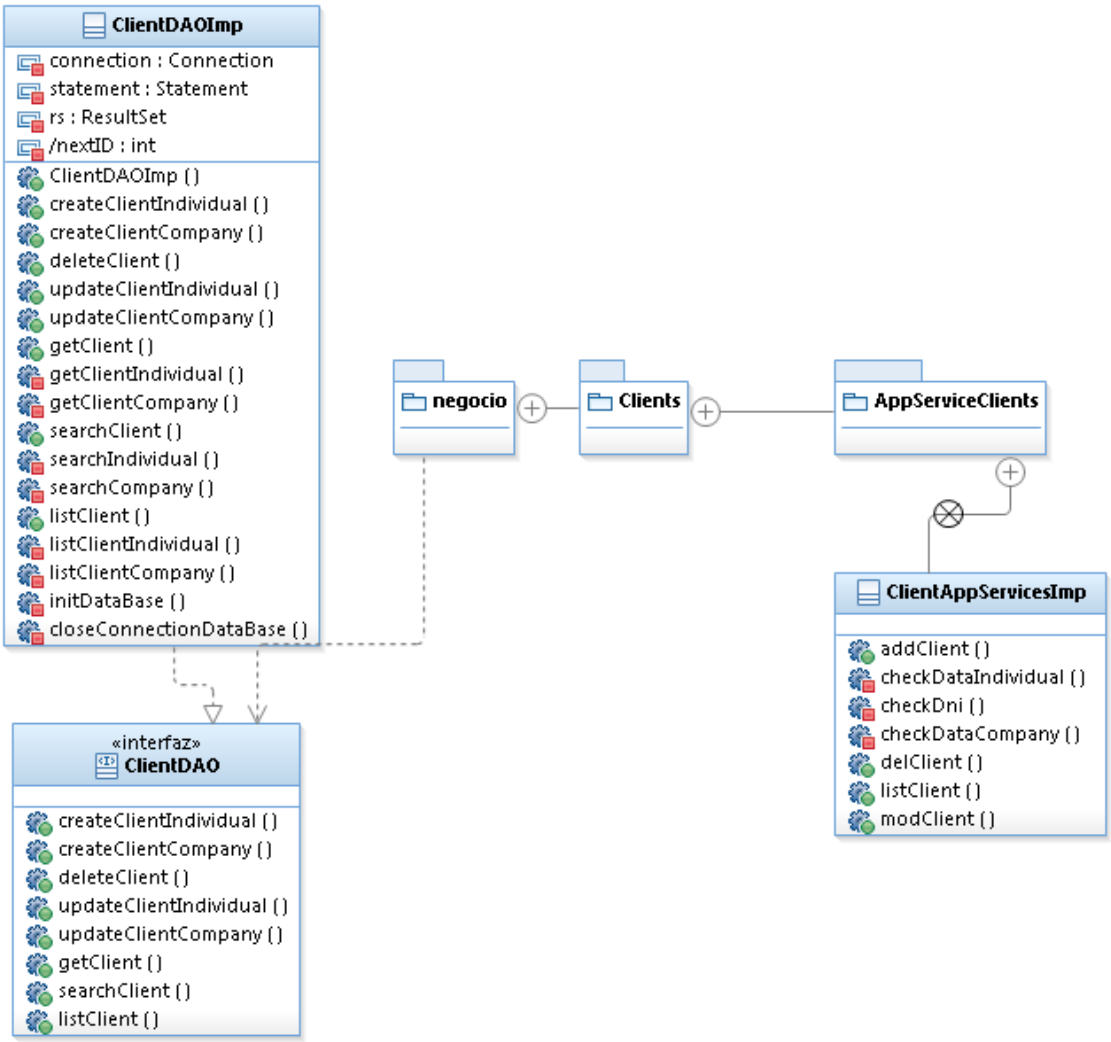


5.4 Diagrama de clases:

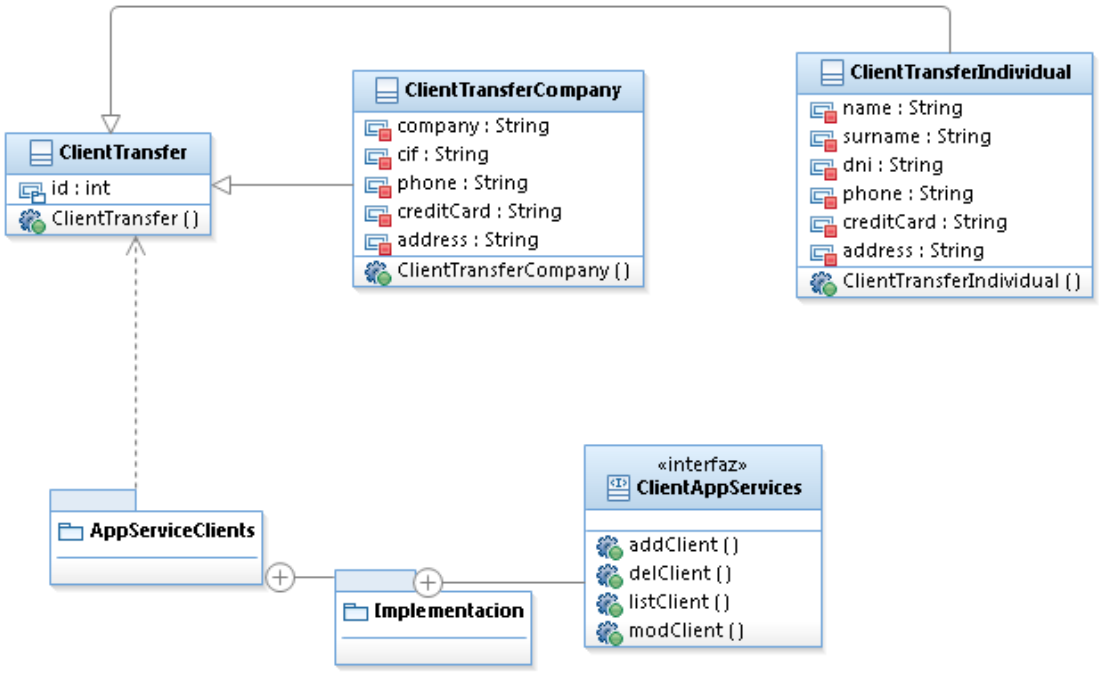
Establece la estructura del sistema mostrando sus clases, atributos y las relaciones entre ellos.

Adjuntamos un ejemplo de cada una de las capas.

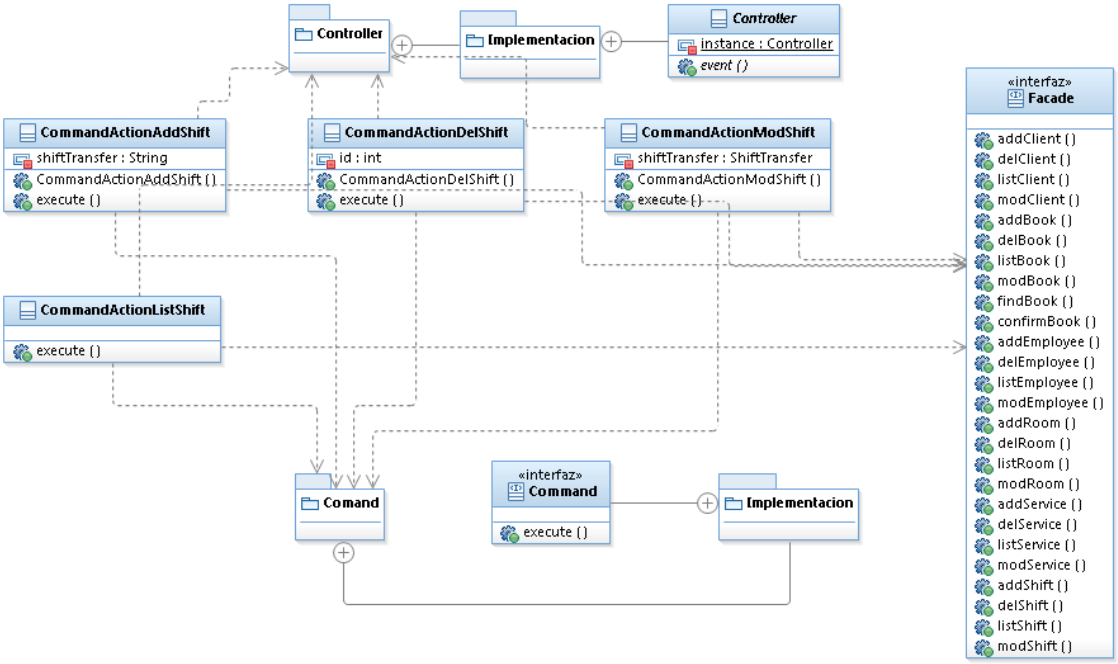
Integración:



Negocio:


ClientTransfer
id : int
ClientTransfer ( )
ClientTransferCompany
company : String
cif : String
phone : String
creditCard : String
address : String
ClientTransferCompany ( )
ClientTransferIndividual
name : String
surname : String
dni : String
phone : String
creditCard : String
address : String
ClientTransferIndividual ( )
AppServiceClients
Implementacion
«interfaz»
ClientAppServices
addClient ( )
delClient ( )
listClient ( )
modClient ( )

Presentación:

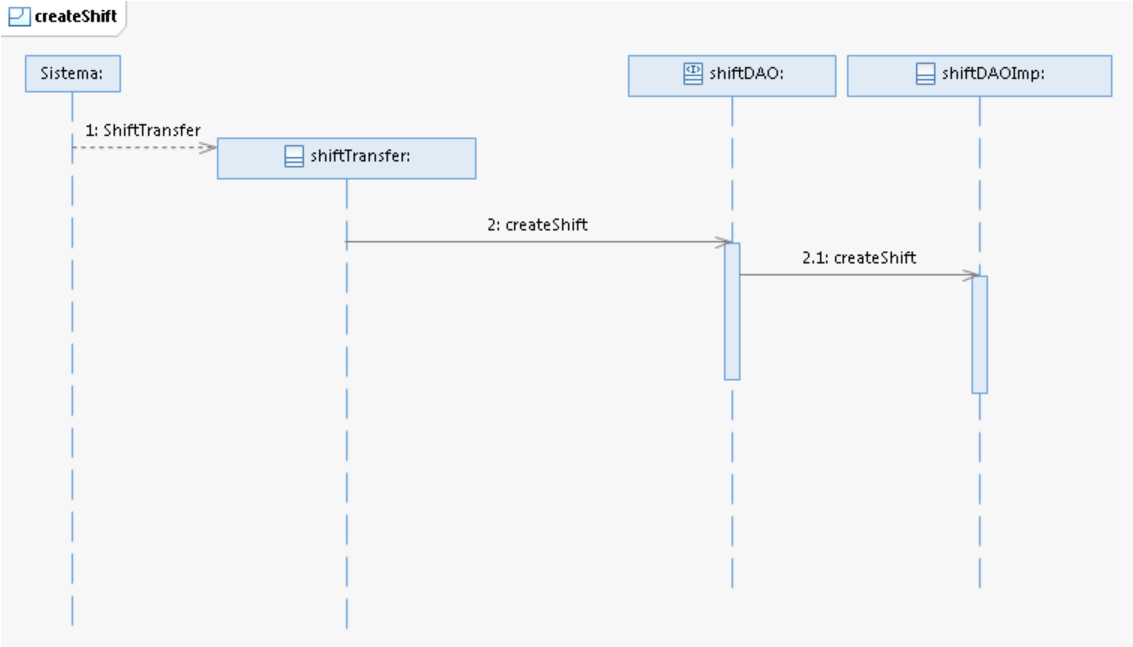


5.5 Diagramas de secuencia:

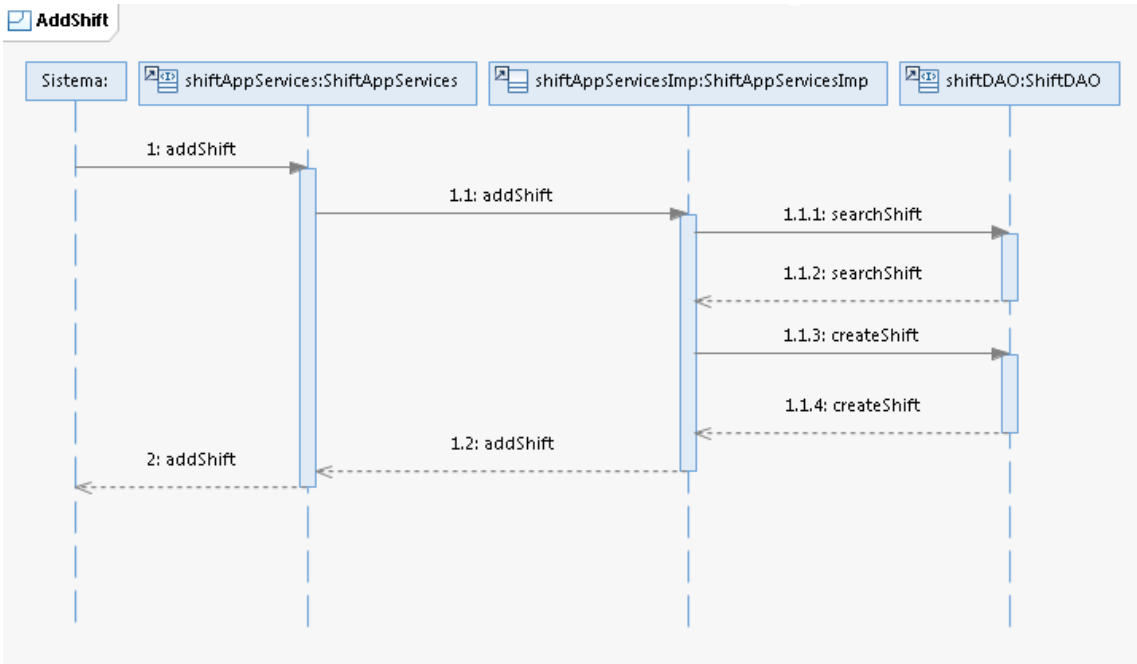
Establece la interacción de un conjunto de objetos en una aplicación a través del tiempo.

Adjuntamos un ejemplo de cada una de las capas.

Integración:



Negocio:



Presentación: