Modelado de Software

Titulación: Grado en Ingeniería del Software
Entrega Memoria Aplicación *"HOTEL MANAGER"*
Curso 2013-2014
Profesor: Antonio Navarro Martín

Participantes:
Álvarez Piñeiro, Emilio
Pérez Valbuena, Juan Luis
Serrano Torres, Daniel

INDICE

1. Introducción a Hotel Manager 3
2. Patrones Aplicados 3
   2.1 Negocio 3
   2.2 Presentación: 3
3. Diagramas de aplicación 3
   3.1 Diagramas de Casos de Uso: 4
   3.2 Diagramas de Actividad: 4
   3.3 Diagramas de Secuencia: 5
4. Base de datos relacional (mysql) + persistence manager (jpa) 5
5. Gestión de Concurrencia 7
6. Testing 7
7. Gestión de excepciones 8

1. INTRODUCCIÓN A HOTEL MANAGER

'Hotel Manager' es un Software de gestión hotelera realizada usando el Proceso Unificado de Rational por el grupo ***'BSoD Software'*** de la Universidad Complutense de Madrid.

Se trata de una aplicación multicapa, realizada íntegramente en Java.

Esta entrega es la segunda y última realizada para la asignatura de *'Modelado del Software'* del Grado de Ingeniería del Software.

Se añaden solo los cambios significativos con respecto a la anterior entrega. Para poder consultar cambios de la primera entrega, referirse a la Memoria de dicha entrega.

2. PATRONES APLICADOS

2.1 NEGOCIO

- **Patrón Objeto de Negocio:** *(Business Object*): Empleados, Tarea, Departamento.
- **Patrón Domain Store :** Servicios de Aplicación Empleados , Tarea y Departamento
- **Singleton:** EntityManagerFactoryS.

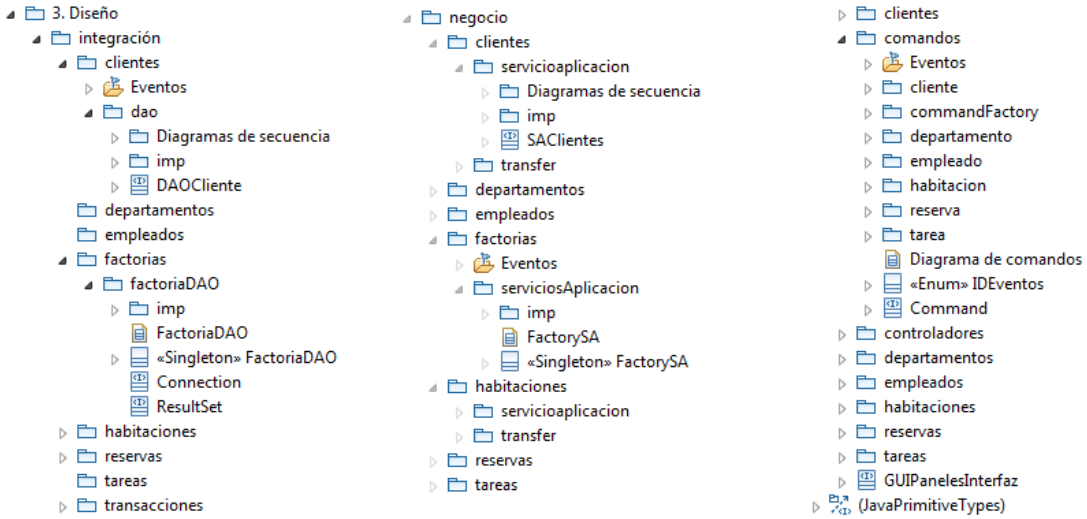
2.2 PRESENTACIÓN:

- **Singleton (objeto único):** ControladorAplicacion, Dispatcher ,GUIClientes, GUIHabitacion, GUIReserva, GUIEmpleados, GUIDepartamentos, GUITareas.
- **Controlador de aplicación (*Application controller*):** ControladorAplicacion.
- **Abstract Factory (Factoría abstracta):** CommandFactory, GUIClientes, GUIReservas, GUIHabitaciones, GUIEmpleados, GUIDepartamentos, GUITareas.

3. DIAGRAMAS DE APLICACIÓN

Se ha utilizado la herramienta CASE '*IBM Rational Software Architect 8.0.3*' para la realización de los diagramas necesarios para el desarrollo de esta aplicación.

El modelo se estructura acorde a la arquitectura multicapa. El primer nivel de paquetes se corresponde con las capas de integración, negocio y presentación en las que se divide la aplicación. Dentro de estos paquetes se encuentran los diagramas y objetos organizados por las entidades de la aplicación (Clientes, habitaciones, reservas, departamentos, empleados, tareas). Aquellos objetos que hacen que hacen referencia a varias entidades, cómo las factorías y los controladores se encuentran incluidos en sus propios paquetes, y en casos especiales, dentro de los mismos se encuentra otra organización por entidades.



En concreto, los diagramas realizados son:

3.1 DIAGRAMAS DE CASOS DE USO:

- Empleado:
  - Alta Empleado
  - Consulta Empleado
  - Modificar Empleado
  - Consulta Todos los Empleados
  - Dar de Baja Empleado
  - Dar de Alta Tarea Empleado
  - Dar de Baja Tarea Empleado
- Tarea:
  - Alta Tarea
  - Modificar Tarea
  - Consultar Tarea
  - Consultar Lista de Tareas
  - Dar de Baja Tarea
- Departamento:
  - Dar de Alta Departamento
  - Dar de Baja Departamento
  - Modificar Departamento
  - Consultar Departamento
  - Consultar todas las Departamento

3.2 DIAGRAMAS DE ACTIVIDAD:

- Empleado:
  - Alta Empleado
  - Consulta Empleado
  - Modificar Empleado
  - Consulta Todos los Empleados
  - Dar de Baja Empleado
  - Dar de Asignar Tareas a Empleados
- Tarea:
  - Alta Tarea

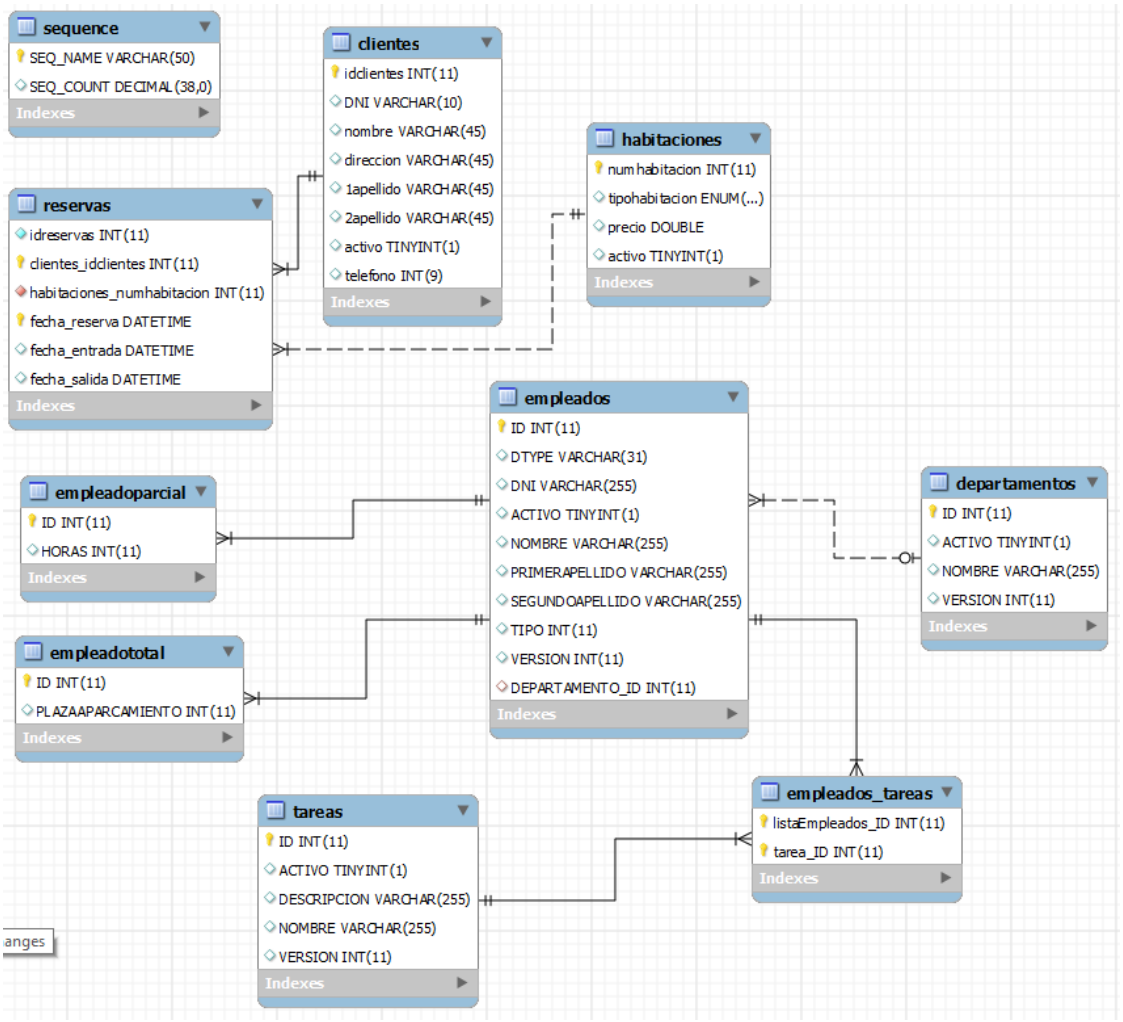
        - Modificar Tarea
        - Consultar Tarea
        - Consultar Lista de Tareas
        - Dar de Baja Tarea
    - Departamento:
        - Dar de Alta Departamento
        - Dar de Baja Departamento
        - Modificar Departamento
        - Consultar Departamento
        - Consultar todas las Departamento

3.3 DIAGRAMAS DE SECUENCIA:

Se eligió un módulo (empleados) y se realizaron los diagramas correspondientes. Se han realizado además los diagramas de las operaciones de otros módulos que entrañan mayor dificultad o complejidad.

4. BASE DE DATOS RELACIONAL (MYSQL) + PERSISTENCE MANAGER (JPA)

Para la persistencia de los datos, se utilizado una base de datos relacional *'MySQL Server 5.5'* en remoto y/o local, lo cual permite el uso de esta aplicación desde cualquier computador con acceso a Internet o con un gestor propio de bases de datos. Se ha realizado el siguiente diseño usando *'MySQL WorkBench 6.0'*.
La base de datos es gestionada por EclipseLink 2.1.3, una implementación del Framework JPA (Almacén del dominio) para Java. EclipseLink realiza la carga dinámica de datos y gestiona el acceso a tablas a través de entidades, además de permitir un amplio rango de configuraciones.


sequence
SEQ_NAME VARCHAR(50)
SEQ_COUNT DECIMAL(38,0)
Indexes
clientes
idclientes INT(11)
DNI VARCHAR(10)
nombre VARCHAR(45)
direccion VARCHAR(45)
1apellido VARCHAR(45)
2apellido VARCHAR(45)
activo TINYINT(1)
telefono INT(9)
Indexes
habitaciones
numhabitacion INT(11)
tipohabitacion ENUM(...)
precio DOUBLE
activo TINYINT(1)
Indexes
reservas
idreservas INT(11)
clientes_idclientes INT(11)
habitaciones_numhabitacion INT(11)
fecha_reserva DATETIME
fecha_entrada DATETIME
fecha_salida DATETIME
Indexes
empleados
ID INT(11)
DTYPE VARCHAR(31)
DNI VARCHAR(255)
ACTIVO TINYINT(1)
NOMBRE VARCHAR(255)
PRIMERAPELLIDO VARCHAR(255)
SEGUNDOAPELLIDO VARCHAR(255)
TIPO INT(11)
VERSION INT(11)
DEPARTAMENTO_ID INT(11)
Indexes
empleadoparcial
ID INT(11)
HORAS INT(11)
Indexes
departamentos
ID INT(11)
ACTIVO TINYINT(1)
NOMBRE VARCHAR(255)
VERSION INT(11)
Indexes
empleadototal
ID INT(11)
PLAZAAPARCAMIENTO INT(11)
Indexes
empleados_tareas
listaEmpleados_ID INT(11)
tarea_ID INT(11)
Indexes
tareas
ID INT(11)
ACTIVO TINYINT(1)
DESCRIPCION VARCHAR(255)
NOMBRE VARCHAR(255)
VERSION INT(11)
Indexes
anges

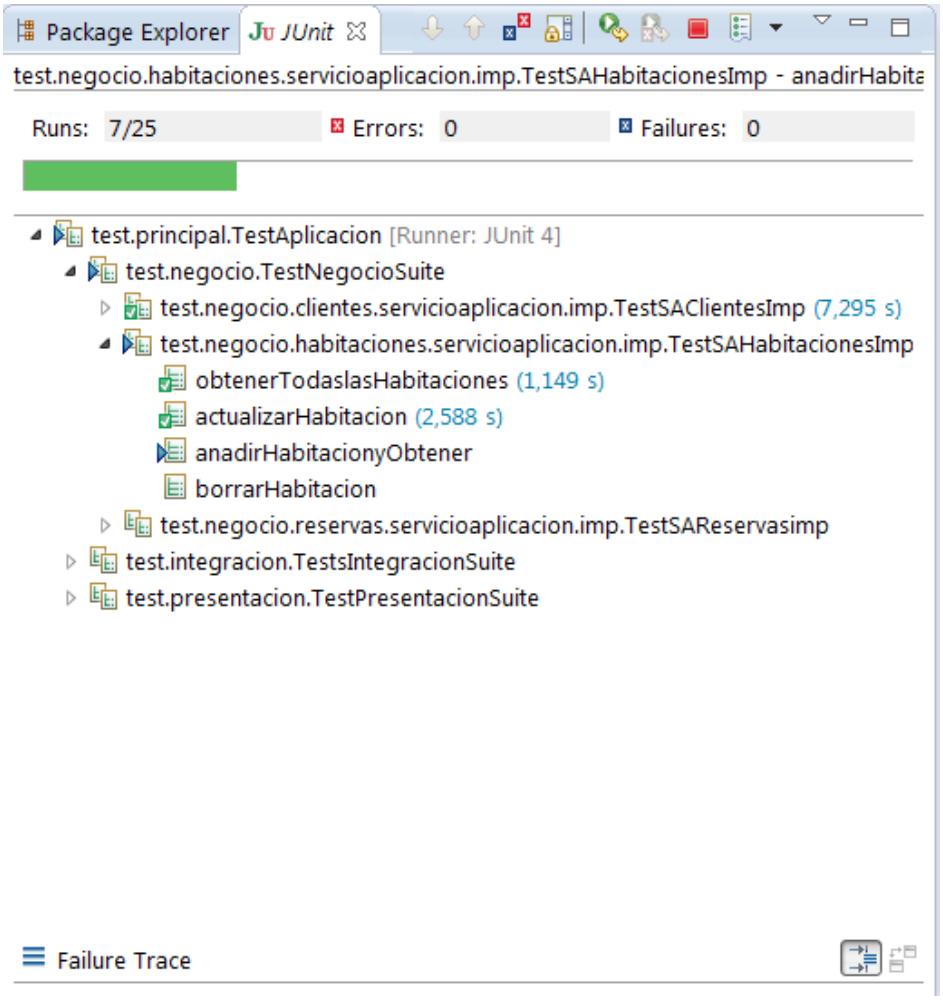
5. GESTIÓN DE CONCURRENCIA

Se ha decidido utilizar una gestión de **concurrencia optimista**. Se ha utilizado lo siguiente:

- Desactivar el *'AUTOCOMMIT'* de nuestra base de datos. Con ello solo persistirán los cambios si se ejecuta posteriormente un *'COMMIT'*. En caso contrario si queremos descartar dichos cambios se ejecutará un *'ROLLBACK'*.
- Se han implementado los mecanismos de gestión de concurrencia de JPA, como las etiquetas.
  @Version (en cada entidad persistible).
  entityManager.lock(entity, LockModeType.OPTIMISTIC_FORCE_INCREMENT)

6. TESTING

Se ha implementado una suite de test de JUnit que prueba los casos de uso de forma general, probando que las salidas son iguales a las esperadas y que no se producen errores a la hora de inicializar los datos.



7. GESTIÓN DE EXCEPCIONES

Se ha realizado una gestión de excepciones personalizada, es decir, se han encapsulado las excepciones del sistema y errores de la aplicación en una excepción propia (*BSoDException*) para poder escalarla por la arquitectura multicapa, y que de esta manera se tenga un control absoluto de los errores.