

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](#_Toc378798609)

[2. Patrones Aplicados 3](#_Toc378798610)

[2.1 Negocio 3](#_Toc378798611)

[2.2 Presentación: 3](#_Toc378798612)

[3. Diagramas de aplicación 3](#_Toc378798613)

[3.1 Diagramas de Casos de Uso: 4](#_Toc378798614)

[3.2 Diagramas de Actividad: 4](#_Toc378798615)

[3.3 Diagramas de Secuencia: 5](#_Toc378798616)

[4. Base de datos relacional (MYSQL) + Persistence Manager (JPA) 5](#_Toc378798617)

[5. Gestión de Concurrencia 7](#_Toc378798618)

[6. Testing 7](#_Toc378798619)

[7. Gestión de excepciones 8](#_Toc378798620)

# 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.

# Patrones Aplicados

## Negocio

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

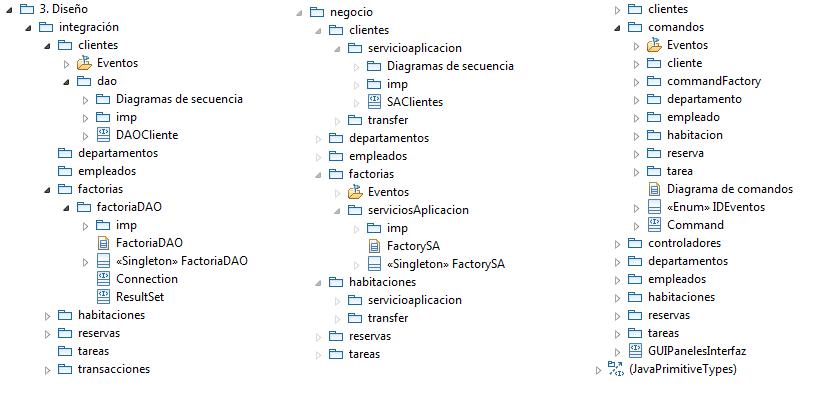
## Presentación:

* + **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.

# 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:

## 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

## 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

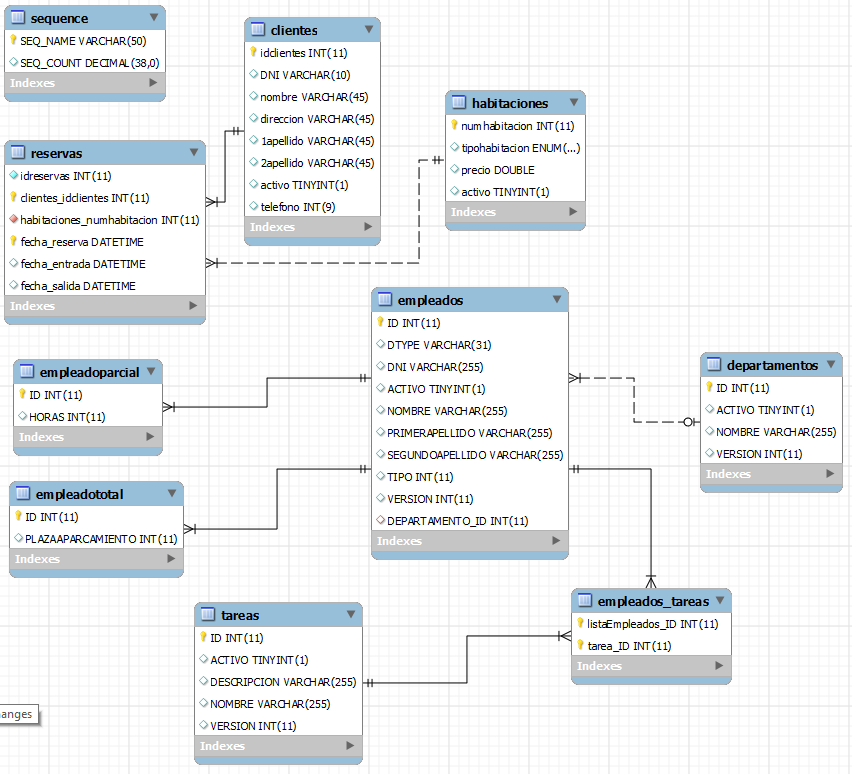
## Diagramas de Secuencia:

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.

# 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 cualpermite 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.



# Gestión de Concurrencia

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

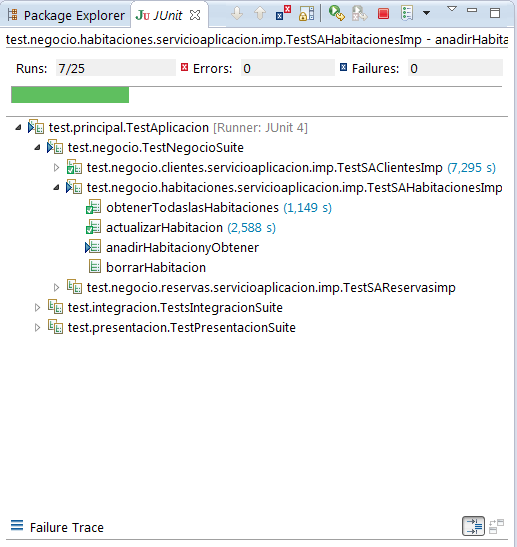
* Desactivar el *‘AUTOCOMMIT’* de nuestra base de datos. Con ello solo persistirán los cambios si se ejecuta posteriormente un *‘COMMIT’*. En caso contrario si queremos descartar dichos cambios se ejecutará un *‘ROLLBACK’.*
* Se han implementado los mecanismos de gestión de concurrencia de JPA, como las etiquetas.

@Version (en cada entidad persistible).

entityManager.lock(entity, LockModeType.OPTIMISTIC\_FORCE\_INCREMENT)

# 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.



# 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.