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 Latorre, Álvaro
Pérez Valbuena, Juan Luis
Quesada Pimentel, Álvaro,
Serrano Torres, Daniel
Zabala Hidalgo, Alejandro

1. INTRODUCCIÓN A HOTEL MANAGER

Esta aplicación denominada *'Hotel Manager'* de gestión hotelera realizada íntegramente en Java es una aplicación multicapa que ha sido desarrollada usando el proceso unificado de Rational.

2. PATRONES APLICADOS

En esta aplicación están bien definidas las tres capas: integración, negocio y presentación.

- **Integración:**
  - Singleton (objecto único) : TransactionFactory , TransactionManager , FactoriaDAO
  - Data Object Access (DAO) : DAOCliente, DAOHabitacion , DAOReserva
- **Negocio**:
  - Singleton (objecto único): FactorySA
  - Servicio de Aplicación (*Application Service*): SACliente , SAHabitacion , SAReservas
  - Tranfer Object : TransferCliente, TransferHabitacion , TransferReservas
- **Presentación:**
  - Singleton (objeto único): ControladorAplicacion, Dispatcher ,GUIClientes, GUIHabitacion, GUIReserva.
  - Controlador de aplicación (*Application controller*) : ControladorAplicacion

En esta aplicación está bien definida la arquitectura multicapa (integración, negocio, presentación)

3. DIAGRAMAS DE APLICACIÓN

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

En concreto:

- Diagramas de Casos de Uso:
  - Clientes:
    - Alta Cliente
    - Consulta Cliente
    - Modificar Cliente
    - Consulta Todos los Clientes
    - Dar de Baja Cliente
  - Habitaciones:
    - Alta Habitación

        - Modificar Habitación
        - Consultar Habitación
        - Consultar Lista de Habitaciones
        - Dar de Baja Habitación
    - Reservas:
        - Dar de Alta Reserva
        - Dar de Baja Reserva
        - Modificar Reserva
        - Consultar Reserva
        - Consultar todas las reservas
        - Consultar reservas de un cliente

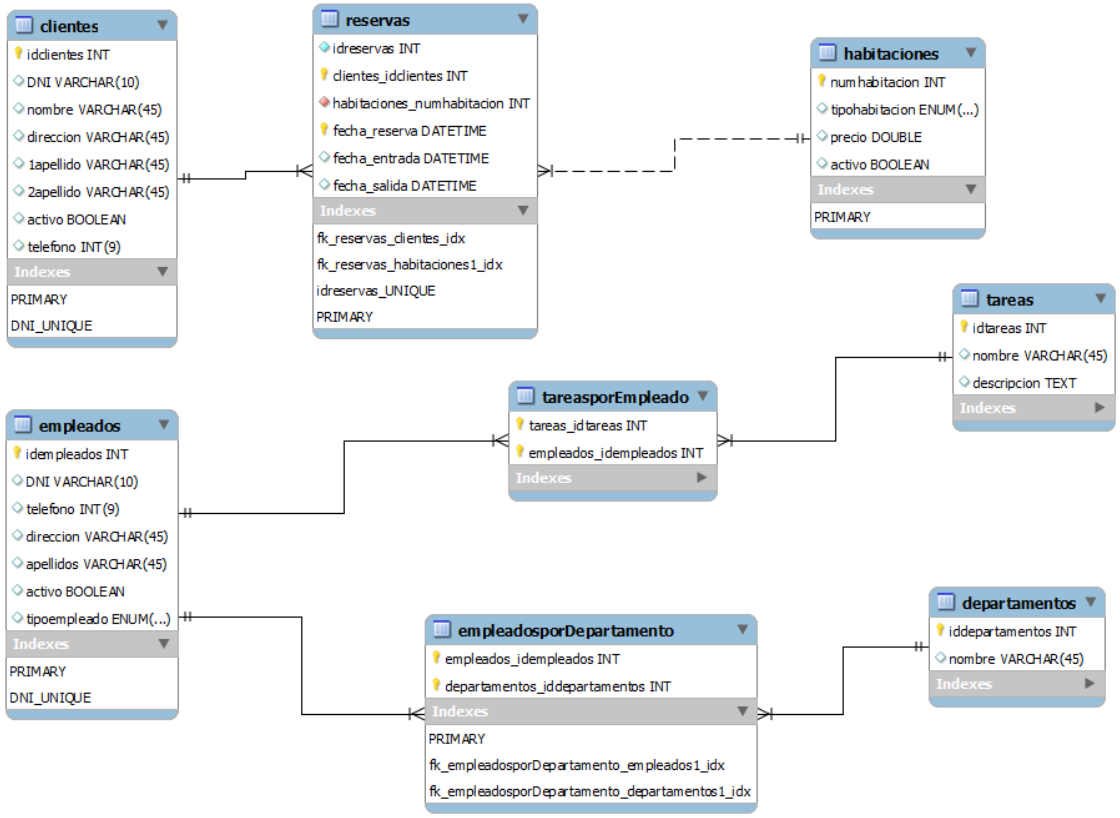
- Diagramas de Actividad:
    - Clientes:
        - Alta Cliente
        - Consulta Cliente
        - Modificar Cliente
        - Consulta Todos los Clientes
        - Dar de Baja Cliente
    - Habitaciones:
        - Alta Habitación
        - Modificar Habitación
        - Consultar Habitación
        - Consultar Lista de Habitaciones
        - Dar de Baja Habitación
    - Reservas:
        - Dar de Alta Reserva
        - Dar de Baja Reserva
        - Modificar Reserva
        - Consultar Reserva
        - Consultar todas las reservas
        - Consultar reservas de un cliente
- Diagramas de Secuencia: Se eligió un módulo y se realizaron los casos de uso de dicho modulo, el resto son análogos. Si hubo algún caso excepcional , se añadió a los diagramas.
    - DAOClientes
        - Alta Cliente
        - Consulta Cliente
        - Modificar Cliente
        - Consulta Todos los Clientes
        - Dar de Baja Cliente
    - FactoriaDAO
        - GeneraDAOCliente
        - GeneraDAOHabitacion
        - GeneraDAOReservas
    - TransactionMySQL
    - TransactionFactory
        - Nueva Transacción
    - TransactionManager
        - Nueva Transacción
        - GetTransaccion
        - Eliminar Transaccion

- SAClientes
  - Alta Cliente
  - Consulta Cliente
  - Modificar Cliente
  - Consulta Todos los Clientes
  - Dar de Baja Cliente
- Comandos
  - Comando Alta Cliente
  - Comando Consulta Cliente
  - Comando Modificar Cliente
  - Comando Consulta Todos los Clientes
  - Comando Dar de Baja Cliente
- Application Controller
  - Handle Request
- Dispatcher
  - Redirect

- Diagrama de Despliegue: …

4. BASE DE DATOS

Para la persistencia de los datos, se utilizado una base de datos relacional *'MySQL Server 5.5'* en remoto, lo cual permite el uso de esta aplicación desde cualquier computador con acceso a Internet.
Se ha realizado el siguiente diseño usando *'MySQL WorkBench 6.0'*



5. GESTIÓN DE CONCURRENCIA

Se ha decidido utilizar una gestión de **concurrencia pesimista**. 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 ha añadido un nivel aislamiento de la base de datos 'REPETABLE_READ'
- Se han añadido a las sentencias SQL de tipo SELECT la cláusula FOR UPDATE para leer el dato más actualizado disponible, estableciendo bloqueos exclusivos sobre cada fila leída. Es decir, el mismo bloqueo que haría UPDATE