

Modelado de Software

Titulación: Grado en Ingeniería del Software

Entrega Documentación Aplicación *“HOTEL MANAGER”*

Curso 2013-2014

Profesor: Antonio Navarro Martín

Participantes:

Alejandro Zabala Hidalgo

Álvaro Pérez Latorre

Álvaro Quesada Pimentel

Daniel Serrano Torres

Emilio Álvarez Piñeiro

Juan Luis Pérez Valbuena

ÍNDICE

[1. Introducción 4](#_Toc372129869)

[1.1 Propósito 4](#_Toc372129870)

[1.2 Alcance 4](#_Toc372129871)

[1.3 Glosario 4](#_Toc372129872)

[1.4 Referencias 4](#_Toc372129873)

[1.5 Resumen 4](#_Toc372129874)

[2. Descripción general 5](#_Toc372129875)

[2.1 Perspectiva del producto 5](#_Toc372129876)

[2.2 Funciones del producto 6](#_Toc372129877)

[2.3 Características de usuario 6](#_Toc372129878)

[2.4 Restricciones 6](#_Toc372129879)

[2.5 Supuestos y dependencias 7](#_Toc372129880)

[2.5.1 Requisitos futuros 7](#_Toc372129881)

[2.6 REQUISITOS ESPECÍFICOS 7](#_Toc372129882)

[2.7 Interfaces externas 7](#_Toc372129883)

[3. Funciones 9](#_Toc372129884)

[3.1 - Clientes: 9](#_Toc372129885)

[3.1.1 **Función:**  Nuevo cliente 9](#_Toc372129886)

[3.1.2 **+ Función:** Baja cliente. 9](#_Toc372129887)

[3.1.3 **Función:** Mostrar Lista de Clientes. 10](#_Toc372129888)

[3.1.4 **Función:** Modificar Cliente 10](#_Toc372129889)

[3.2 Habitaciones 10](#_Toc372129890)

[3.2.1 **Función:** Alta de Habitación 10](#_Toc372129891)

[3.2.2 **Función:** Baja de Habitación 11](#_Toc372129892)

[3.2.3 **Función:** Modificar Habitación 11](#_Toc372129893)

[3.2.4 **Función:** Mostrar Habitación 11](#_Toc372129894)

[3.2.5 **Función:** Mostrar Lista de Habitaciones. 12](#_Toc372129895)

[3.3 Reservas 12](#_Toc372129896)

[3.3.1 **Función:** Alta de una reserva 12](#_Toc372129897)

[3.3.2 **Función:** Baja de reservas 12](#_Toc372129898)

[3.3.3 **Función:** Modificar reserva 13](#_Toc372129899)

[3.3.4 **Función:** Consultar reserva 13](#_Toc372129900)

[3.3.5 **Función:** Consultar todas las reservas 13](#_Toc372129901)

[3.3.6 **Función:** Consultar reservas por cliente 14](#_Toc372129902)

[3.4 Departamentos: 14](#_Toc372129903)

[3.4.1 **Función:** Mostrar departamento 14](#_Toc372129904)

[3.4.2 **Función:** Mostrar todos los departamentos 15](#_Toc372129905)

[3.4.3 **Función:** Modificar departamento 15](#_Toc372129906)

[3.4.4 **Función:** Borrar departamento 15](#_Toc372129907)

[3.5 Empleados 16](#_Toc372129908)

[3.5.1 **Función:** Alta empleado 16](#_Toc372129909)

[3.5.2 **Función:** Baja empleado 16](#_Toc372129910)

[3.5.3 **Función:** Modificar empleado 16](#_Toc372129911)

[3.5.4 **Función:** Mostrar empleado 17](#_Toc372129912)

[3.5.5 **Función:** Mostrar empleados de una tarea 17](#_Toc372129913)

[3.5.6 **Función:** Añadir tarea a un empleado 17](#_Toc372129914)

[3.6 Tareas 18](#_Toc372129915)

[3.6.1 **Función:** Añadir tarea 18](#_Toc372129916)

[3.6.2 **Función:** Borrar tarea 18](#_Toc372129917)

[3.6.3 **Función:** Mostrar tarea 18](#_Toc372129918)

[3.6.4 **Función:** Mostrar todas las tareas 19](#_Toc372129919)

[3.6.5 **Función:** Modificar tarea 19](#_Toc372129920)

[3.6.6 **Función:** Tareas asignadas a un empleado. 19](#_Toc372129921)

[4. Requisitos de rendimiento 20](#_Toc372129922)

[4.1 Requisitos lógicos de la base de datos 20](#_Toc372129923)

[4.2 Requisitos funcionales 20](#_Toc372129924)

[4.3 Atributos del sistema de software 21](#_Toc372129925)

# Introducción

## Propósito

* Software destinado a la gestión de un hotel. Se encarga de administrar los clientes, reservas de habitaciones, departamentos, empleados y sus tareas.
* Audiencia:
  + Cliente: **Antonio Navarro**.
  + Desarrollador: **BSoD Software**.
  + Usuario: **Antonio Navarro**.

## Alcance

* Nombre del producto software: **Hotel Manager.**
* Gestiona las altas, bajas y modificaciones de los distintos clientes, las reservas y cancelaciones que estos pueden hacer de las habitaciones del hotel. También se gestionan altas, bajas y modificaciones de los empleados y departamentos, así como las tareas de los empleados.
* Uso e interfaz sencilla, con multitud de funcionalidades. Completo manejo de la administración de las habitaciones, los clientes, reservas, empleados, departamentos y tareas.

## Glosario

* Tareas: Trabajos que realiza un empleado en su jornada laboral.
* Clientes: Persona reserva al menos una vez como mínimo una habitación.

## Referencias

* IEEE Std. 830-1998. Recommended Practice for Software Requirements Specification, 1998.
* Alur, D., Crupi. J . Core J2EE Design Patterns: Best Practices and Design Strategies. 2nd Edition. Pretice Hall, 2003.

## Resumen

Este proyecto consiste en la creación de una aplicación que gestiona las reservas de habitaciones de un hotel, y las tareas de los empleados de sus departamentos. Dicha aplicación consta de 6 módulos: habitaciones, clientes, reservas, departamentos, empleados y tareas. Cada módulo consta de sus correspondientes funcionalidades para llevar a cabo una gestión de las mismas (consultar, crear, actualizar y borrar). Las principales funciones de la aplicación son: gestionar las reservas de habitaciones y las tareas de los empleados. El usuario de la aplicación tendrá acceso a todos los módulos sin restricciones. El proyecto está pensado para llevarse a cabo en un entorno donde sólo es necesario disponer de un terminal.

Para los requisitos funcionales, seguiremos reglas impuestas por el IEEE 830 – 1998, sujetos a las normas estipuladas por nuestro modelo unificado de Rational.

# Descripción general

## Perspectiva del producto

* Contexto de implantación.

Nuestro producto está orientado a la gestión de la actividad económica que consiste en proveer estancias en las habitaciones de un hotel, y a la administración de los empleados del mismo.

* Interfaces del sistema.

Tenemos un programa principal y una base de datos. El empleado encargado de la gestión de las reservas gestionará las peticiones de los clientes, y el programa hace sus correspondientes operaciones con la base de datos. Dicha base de datos estará definida en un principio en el lenguaje SQL y el gestor de base de datos MySQL, el software principal en el lenguaje de programación Java.

* Interfaces de usuario.

El usuario se comunica con el programa principal a través de una interfaz gráfica realizada con swing (tecnología de Java para representar interfaces de usuario).

Los requisitos para poder presentar la interfaz de usuario serán los siguientes:

* Resolución de pantalla mínima: 1280x720 pixels.
* Sistema operativo de la plataforma: Windows, Mac OS X, Linux, sistemas basados en Unix.
* JRE ( sistema de ejecución de aplicaciones java).

La interfaz de usuario estará integrada en una única ventana, de tal forma que se dividirá por secciones (en pestañas de navegación u en otro elemento de similar comportamiento…).

Los mensajes de aviso, error, o confirmación se mostrarán en forma de diálogo (ventana independiente asociada a la principal).

* Interfaces hardware.

Se necesitará la capacidad de conexión local al puerto por defecto de MySQL (3306).

* Interfaces Software.

La plataforma que ejecutará nuestra aplicación necesitará tener instalada la máquina virtual de Java (JVM), esta máquina virtual deberá ser la versión 7 o superior.

* Interfaces de comunicación.

Es una aplicación de escritorio que no necesita de ninguna comunicación con ningún otro equipo.

* Memoria.

Trabajaremos con las memorias RAM de al menos 512MB. Solo será necesario persistir los datos mediante el gestor de base de datos en memoria secundaria (disco duro).

* Operaciones.

La aplicación contará con un único nivel de uso (acceso total) y no contará con ningún sistema de backup.

* Instalaciones particulares.

Será necesario tener instalado o instalar el gestor de bases de datos, MySQL 5 o superior.

## Funciones del producto

Los clientes pueden realizar reservas de las habitaciones del hotel, habiendo dos tipos de estas (normal y suite). A su vez si un cliente así lo precisa y no se ha sobrepasado la fecha de una reserva puede darla de baja o modificar la reserva, siempre y cuando esta última no entre en conflicto con otra ya establecida.

Se podrán añadir nuevas habitaciones en el caso por ejemplo de una ampliación del hotel, borrarlas si se necesita hacer una reforma, o modificar sus datos.

El producto gestiona un conjunto de departamentos a los cuales estarán repartidos los empleados. Pudiendo añadir departamentos si se generan nuevos durante la actividad del hotel, modificar los datos de los departamentos si hubiera necesidad de ello, y borrarlos en caso de que ya no existiese tal departamento.

Por otra parte, también se pueden gestionar los empleados de los departamentos del hotel, realizando asignaciones de tareas para las jornadas laborales (media o completa), las cuales se le pueden eliminar al empleado o modificar, por ejemplo el número de horas de la tarea, si así se precisa. No se podrá asignar tareas a empleados que no estén en un departamento.

## Características de usuario

El producto está pensado para que pueda ser utilizado por personas con nivel de informática muy básico. Al ser un programa muy sencillo, el tiempo de aprendizaje para que se maneje de una forma óptima será muy bajo.

## Restricciones

* Las políticas de regulación se han seguido de acuerdo con el estándar IEEE.
* Limitaciones hardware: necesario un medio para mostrar el entorno gráfico de la plataforma.
* No existirán interfaces con otras aplicaciones.
* Operaciones en paralelo: tratamiento de archivos de registros.
* Funciones de auditoria: ninguna.

## Supuestos y dependencias

* Factores que afectan al desarrollo del proyecto:
  + Desconocimiento de las tecnologías que se van a usar.
  + Falta de tiempo.
  + Disminución de los miembros activos del equipo.

### Requisitos futuros

* No se esperan requisitos futuros.

## REQUISITOS ESPECÍFICOS

## Interfaces externas

Las entradas al sistema se realizarán a través de una interfaz por teclado, con sus restricciones pertinentes. Las salidas serán mostradas en la interfaz de usuario o reflejadas en un de registro de la aplicación. Independientemente de la función, las entradas por teclado tendrán un determinado formato que se le indicará al usuario mediante la interfaz de usuario para que no haya lugar a errores.

El nombre del elemento será **ManageTel** proporcionado por la empresa **BSoD Software** y dedicado a la administración de un hotel, por tanto este software está dirigido a clientes/empresas que gestionen uno. El origen de la entrada será una acción de los clientes/empresas deberá ser registrada en la base de datos y la salida constará de la interfaz de usuario del sistema y del archivo de registro. El rango en el que actuará el software será el rango aceptable y lógico de las transacciones empresariales comunes, es decir, registro de habitaciones que tiene el hotel, registro de reservas/anulaciones de habitaciones, gestión de los empleados, de los departamentos a los que pertenece y de las tareas de cada empleado. Las unidades de medida serán los estándares. Las habitaciones tendrán un tipo (normal y suite), el precio y costo se medirá en euros, los turnos se medirán en horas (existiendo dos tipos, medio y completo). La interfaz será una ventana amigable con el usuario en la que se podrá acceder fácilmente a cada una de las operaciones necesarias para cada gestión. El formato de las fechas será el estándar europeo, es decir, DD/MM/AAAA.

Las entradas de elementos al software serán validadas en función de los requisitos de la función que se encargue de su aceptación. Concretamente:

* En el módulo clientes:
  + El DNI debe contener 8 cifras y una letra.
  + Teléfono será un campo numérico de 9 dígitos exactos, positivos y sin coma flotante.
  + La dirección será una cadena de caracteres.
  + Los clientes tiene un atributo que indican si están activos o no.
  + A cada cliente se le asignará un ID que lo identificará.
  + No puede existir dos clientes con el mismo DNI
* En el módulo habitaciones:
  + Hay dos tipos de habitaciones: habitaciones normales y suites.
  + A cada habitación se le asignará un ID que la identificará.
* En el módulo reservas:
  + El precio de la reserva dependerá del tipo de habitación y de la duración de la estancia.
  + A cada reserva se le asignará un ID que la identificará.
* En el módulo departamentos:
  + Los departamentos pueden no tener empleados asociados.
  + A cada departamento se le asignará un ID que lo identificará.
* En el módulo empleados:
  + El DNI debe contener 8 cifras y una letra.
  + Teléfono será un campo numérico de 9 dígitos exactos, positivos y sin coma flotante.
  + La dirección será una cadena de caracteres.
  + Los empleados tiene un atributo que indican si están activos o no.
  + A cada empleado se le asignará un ID que lo identificará.
  + Un empleado tiene entre 0 y varias tareas asociadas
  + Hay dos tipos de empleados, los que trabajan a tiempo completo y los que trabajan a tiempo parcial
* Por último, en el módulo de las tareas:
  + Cada tarea tendrá un nombre descriptivo y una descripción extensa de en qué consisten, estos campos tienen un límite de tamaño grande.
  + A cada tarea se le asociará un ID que la identificará.

Las respuestas ante situaciones excepcionales serán controladas. En caso de que escapase al control, la entrada será rechazada y mostrará un mensaje de error y se deberá hacer una nueva petición de entrada.

# Funciones

A continuación exponemos los requisitos funcionales propios de cada función, organizado por módulos:

## - Clientes:

### **Función:** Nuevo cliente

* + - Prioridad: Alta
    - Estabilidad: Alta
    - Descripción: Introducimos en la base de datos de un nuevo cliente. Pide datos del cliente y lo introduce en la base de datos.
    - Entrada: DNI, Nombre, Apellidos, Número de teléfono.
    - Salida: Comprobación de si está correcto y asignación de ID al cliente en caso de que está correcto.
    - Origen: Interfaz de usuario.
    - Destino: Sistema y base de datos.
    - Necesita: Acceso a la base de datos .
    - Acción: Dar de alta a un cliente.
    - Precondición: Cliente no existente en la base de datos (con el mismo ID).
    - Postcondición: Que se haya dado de alta correctamente y figure en la base de datos con un nuevo ID.
    - Efectos colaterales: ninguno.

### **Función:** Baja cliente.

* + - Prioridad: Baja.
    - Estabilidad: Alta.
    - Descripción: Comprueba si cumple los requisitos para dar de baja y se ejecuta en caso positivo
    - Entrada: ID del cliente
    - Salida: Si la operación se ha ejecutado correctamente cambiando el cliente a inactivo.
    - Origen: Interfaz de usuario.
    - Destino: Sistema y base de datos.
    - Necesita: Base de datos y el ID.
    - Acción: Borra un cliente.
    - Precondición: Que el ID exista en la base de datos y que el cliente esté activo.
    - Postcondición: Cliente cambiado a inactivo.
    - Efectos colaterales: ninguno

#### **Función:** Mostrar Cliente.

* + - Prioridad: Media.
    - Estabilidad: Alta.
    - Descripción: Muestra en la interfaz de usuario los datos del cliente.
    - Entrada: ID del cliente.
    - Salida: Datos del cliente en la interfaz de usuario.
    - Origen: Interfaz de usuario.
    - Destino: Interfaz de usuario.
    - Necesita: Acceso a la base de datos.
    - Acción: Mostrar en la interfaz de usuario los datos del cliente.
    - Precondición: Que la ID exista.
    - Postcondición: Que los datos mostrados sean los del cliente elegido.
    - Efectos laterales: Mostrar mensaje de error si el cliente está inactivo.

### **Función:** Mostrar Lista de Clientes.

* + - Prioridad: Media.
    - Estabilidad: Alta.
    - Descripción: Muestra en la interfaz de usuario la Id con el nombre de todos los clientes.
    - Entrada: vacía.
    - Salida: Id y Nombre de los clientes.
    - Origen: Interfaz de usuario.
    - Destino: Interfaz de usuario.
    - Necesita: Acceso a la base de datos.
    - Acción: Mostrar en el interfaz de usuario los Id y nombres de los clientes.
    - Precondición: Ninguna.
    - Postcondición: Que los datos mostrados sean los de los clientes de la base de datos.
    - Efectos laterales: Mostrar mensaje de advertencia si no existen clientes.

### **Función:** Modificar Cliente

* + - Prioridad: Media.
    - Estabilidad: Alta.
    - Descripción: Modificar los datos de un cliente.
    - Entrada: ID del cliente, selección de datos a modificar, nuevos datos.
    - Salida: Si el cliente ha sido modificado correctamente o no.
    - Origen: Interfaz de usuario.
    - Destino: Base datos.
    - Necesita: Acceso a la base de datos.
    - Acción: Modificar un cliente (sus datos).
    - Precondición: Que la ID sea correcta. Que si se cambia el DNI, el nuevo DNI no exista en la base de datos.
    - Postcondición: Que el cliente haya sido modificado con los datos correctos.
    - Efectos colaterales: ninguno.

## Habitaciones

### **Función:** Alta de Habitación

* + - Prioridad: Alta.
    - Estabilidad: Alta.
    - Descripción: Añade una habitación a la base de datos.
    - Entrada: Datos de la habitación (Nº de habitacion, Tipo de habitación).
    - Salida: Si se ha añadido correctamente .
    - Origen: Interfaz de usuario.
    - Destino: Base de datos.
    - Necesita: Base de datos .
    - Acción: Da de alta una habitación y le asigna un Nº de habitación.
    - Precondición: Que el Id de la habitación no exista en la base de datos.
    - Postcondición: Habitación añadida a la base de datos.
    - Efectos colaterales: ninguno.

### **Función:** Baja de Habitación

* + - Prioridad: Baja.
    - Estabilidad: Alta.
    - Descripción: Elimina una habitación de la base de datos.
    - Entrada: Id de la habitación (Nº de habitación).
    - Salida: Si se ha añadido correctamente.
    - Origen: Interfaz de usuario.
    - Destino: Base de datos.
    - Necesita: Acceso a la base de datos .
    - Acción: Dar de baja una habitación.
    - Precondición: Que exista esa habitación en la base de datos, que no tenga reservas pendientes.
    - Postcondición: Habitación cambiado a inactivo.
    - Efectos colaterales: ninguno.

### **Función:** Modificar Habitación

* + - Prioridad: Media.
    - Estabilidad: Alta.
    - Descripción: Modifica los datos de una habitación.
    - Entrada: Nº de habitación y los datos a modificar.
    - Salida: Si se ha añadido correctamente.
    - Origen: Interfaz de usuario.
    - Destino: Sistema y base de datos.
    - Necesita: Acceso a la base de datos.
    - Acción: Modificar algún dato de una habitación.
    - Precondición: Que exista el Id de la habitación en la base de datos.
    - Postcondición: Los datos se hayan modificado correctamente.
    - Efectos colaterales: ninguno.

### **Función:** Mostrar Habitación

* + - Prioridad: Media.
    - Estabilidad: Alta.
    - Descripción: Muestra en la interfaz de usuario los datos de la habitación.
    - Entrada: Nº de habitación.
    - Salida: Datos de la habitación.
    - Origen: Interfaz de usuario.
    - Destino: Interfaz de usuario.
    - Necesita: Acceso a la base de datos.
    - Acción: Mostrar en la interfaz de usuario los datos de la habitación.
    - Precondición: Que el Nº de habitación exista. en la base de datos
    - Postcondición: Que los datos mostrados sean los de la habitación elegido.
    - Efectos colaterales: Mostrar mensaje de error si la habitación está inactivo.

### **Función:** Mostrar Lista de Habitaciones.

* + - Prioridad: Media.
    - Estabilidad: Alta.
    - Descripción: Muestra en la interfaz de usuario la lista de habitaciones.
    - Entrada: ninguna.
    - Salida: Nº de habitaciones y Tipo.
    - Origen: Interfaz de usuario.
    - Destino: Interfaz de usuario.
    - Necesita: Acceso a la base de datos.
    - Acción: Mostrar en la interfaz de usuario los Nº de habitaciones y tipo de habitaciones.
    - Precondición: Ninguna.
    - Postcondición: Que los datos mostrados sean los de las habitaciones de la base de datos.
    - Efectos colaterales: Mostrar mensaje de advertencia si no existen habitaciones.

## Reservas

### **Función:** Alta de una reserva

* + - Prioridad: Alta.
    - Estabilidad: Alta.
    - Descripción: Se da de alta una determinada reserva.
    - Entrada: Fecha de la reserva, Fecha de entrada, fecha de salida, DNI del cliente, número de habitación.
    - Salida: si se ha incorporado a la base de datos.
    - Origen: Interfaz de usuario.
    - Destino: Sistema y base de datos.
    - Necesita: Acceso a la base de datos.
    - Acción: Añadir una nueva reserva.
    - Precondición: Que el DNI del cliente exista en la base de datos, que la habitación exista en la base de datos, que día de entrada sea menor que el día de salida y que no exista una reserva de la misma habitación para esas fechas.
    - Postcondición: Nueva reserva añadida a la base de datos.
    - Efectos colaterales: ninguno.

### **Función:** Baja de reservas

* + - Prioridad: Alta.
    - Estabilidad: Alta.
    - Descripción: Se da de baja una determinada reserva.
    - Entrada: ID de la reserva.
    - Salida: Si se ha eliminado correctamente de la base de datos.
    - Origen: Interfaz de usuario.
    - Destino: Sistema y base de datos.
    - Necesita: Acceso a la base de datos.
    - Acción: Eliminar una reserva.
    - Precondición: Que la ID que exista en la base de datos.
    - Postcondición: Reserva eliminada correctamente de la base de datos.
    - Efectos colaterales: ninguno.

### **Función:** Modificar reserva

* + - Prioridad: Alta.
    - Estabilidad: Alta.
    - Descripción: Se modifica una determinada reserva.
    - Entrada: ID de la reserva.
    - Salida: Datos de la reserva.
    - Origen: Interfaz de usuario.
    - Destino: Sistema y base de datos.
    - Necesita: Acceso a la base de datos.
    - Acción: Modificar la reserva especificada.
    - Precondición: que la ID sea válida, es decir, que exista. Que los parámetros a introducir para modificar los atributos sean válidos. (No se considera válido cambiar la ID de la reserva).
    - Postcondición: Reserva modificada correctamente en la base de datos.
    - Efectos colaterales: ninguno.

### **Función:** Consultar reserva

* Prioridad: Media.
* Estabilidad: Alta.
* Descripción: Se muestran los datos de una determinada reserva.
* Entrada: ID de la reserva.
* Salida: Datos de la reserva.
* Origen: Interfaz de usuario.
* Destino: Interfaz de usuario.
* Necesita: Acceso a la base de datos.
* Acción: Mostrar los datos de la reserva especificada.
* Precondición: que la ID sea válida, es decir, que exista.
* Postcondición: Reserva mostrada en la interfaz de usuario.
* Efectos colaterales: ninguno.

### **Función:** Consultar todas las reservas

* Prioridad: Media.
* Estabilidad: Alta.
* Descripción: Se muestran todas las reservas existentes.
* Entrada: ninguna.
* Salida: Lista de todas las reservas existentes.
* Origen: Interfaz de usuario.
* Destino: Interfaz de usuario.
* Necesita: Acceso a la base de datos.
* Acción: Mostrar las reservas existentes.
* Precondición: Ninguna.
* Postcondición: Reservas mostradas en la interfaz de usuario.
* Efectos colaterales: ninguno.

### **Función:** Consultar reservas por cliente

* Prioridad: Media.
* Estabilidad: Alta.
* Descripción: Se muestran todas las reservas realizadas por un determinado cliente.
* Entrada: DNI del cliente.
* Salida: Reservas realizadas por el cliente determinado.
* Origen: Interfaz de usuario.
* Destino: Interfaz de usuario.
* Necesita: Acceso a la base de datos.
* Acción: Mostrar reservas de un cliente.
* Precondición: Que el Id de cliente exista en la base de datos.
* Postcondición: Reservas del cliente mostradas por la interfaz de usuario.
* Efectos colaterales: ninguno.

## Departamentos:

**+ Función:** Añadir departamento

* + - Prioridad: Alta.
    - Estabilidad: Alta.
    - Descripción: Se crea un departamento al cual pertenecen un grupo de empleados.
    - Entrada: Nombre del departamento.
    - Salida: Ninguna.
    - Origen: Interfaz de usuario.
    - Destino: Base de datos.
    - Necesita: Acceso a la base de datos.
    - Acción: Crear un nuevo departamento.
    - Precondición: No existe otro departamento con el mismo nombre.
    - Postcondición: Nuevo departamento creado y con ID asignado.
    - Efectos colaterales: Ninguno.

### **Función:** Mostrar departamento

* + - Prioridad: Baja.
    - Estabilidad: Alta.
    - Descripción: Muestra en la interfaz de usuario los departamentos existentes.
    - Entrada: Nombre del departamento.
    - Salida: Datos del departamento.
    - Origen: Interfaz de usuario.
    - Destino: Interfaz de usuario.
    - Necesita: Acceso a la base de datos.
    - Acción: Mostrar los datos del departamento.
    - Precondición: Un nombre de departamento existente en la base de datos.
    - Postcondición: Ninguna.
    - Efectos colaterales: Ninguno.

### **Función:** Mostrar todos los departamentos

* + - Prioridad: Baja.
    - Estabilidad: Alta.
    - Descripción: Muestra en la interfaz de usuario todos los departamentos del sistema.
    - Entrada: Ninguna.
    - Salida: Departamentos almacenados en la base de datos.
    - Origen: Interfaz de usuario.
    - Destino: Interfaz de usuario.
    - Necesita: Acceso a la base de datos.
    - Acción: Mostrar en la interfaz de usuario los departamentos del sistema.
    - Precondición: Ninguna.
    - Postcondición: Ninguna.
    - Efectos colaterales: Si no hay departamentos la lista estará vacía.

### **Función:** Modificar departamento

* Prioridad: Media.
* Estabilidad: Media.
* Descripción: Se pueden modificar los datos de un departamento.
* Salida: Si la operación se ha llevado a cabo.
* Origen: Interfaz de usuario.
* Destino: Base de datos.
* Necesita: Acceso a la base de datos.
* Acción: Modificar datos de un departamento.
* Precondición: Que el nombre del departamento no exista.
* Postcondición: Ninguna.
* Efectos laterales: A todos los empleados les actualiza los datos del departamento.

### **Función:** Borrar departamento

* Prioridad: Baja.
* Estabilidad: Media.
* Descripción: Permite eliminar del sistema un departamento que no tenga empleados asignados.
* Salida: Si la operación se ha llevado a cabo.
* Origen: Interfaz de usuario.
* Destino: Base de datos.
* Necesita: Acceso a la base de datos
* Acción: Modificar datos de un departamento.
* Precondición: Que el departamento exista y no tenga empleados.
* Postcondición: Ninguna.
* Efectos laterales: Ninguna.

## Empleados

### **Función:** Alta empleado

* + Prioridad: Media.
  + Estabilidad: Alta.
  + Descripción: Añade un empleado en la base de datos.
  + Entrada: Datos de empleado(Nombre, Apellido, DNI , idDepartamento, tipo de empleado).
  + Salida: Se ha añadido correctamente.
  + Origen: Interfaz de usuario.
  + Destino: Base de datos.
  + Necesita: Acceso a la base de datos.
  + Acción: Añadir un empleado a la base de datos.
  + Precondición: Que el empleado no exista ya en la base de datos.
  + Postcondición: Que el empleado se haya insertado correctamente.
  + Efectos colaterales: ninguno.

### **Función:** Baja empleado

* + Prioridad: Media.
  + Estabilidad: Alta.
  + Descripción: Elimina un empleado de la bases de datos.
  + Entrada: DNI.
  + Salida: Si se ha eliminado correctamente.
  + Origen: Interfaz de usuario.
  + Destino: Base de datos.
  + Necesita: Acceso a la base de datos.
  + Acción: Dar de baja un usuario.
  + Precondición: Que exista el empleado en la bases de datos.
  + Postcondición: El empleado ha sido eliminado de la base de datos y todas sus tareas asignadas.
  + Efectos colaterales: ninguno.

### **Función:** Modificar empleado

* + Prioridad: Media.
  + Estabilidad: Alta.
  + Descripción: Modifica algún dato del empleado.
  + Entrada: DNI del empleado y datos a modificar.
  + Salida: Se ha eliminado correctamente.
  + Origen: Interfaz de usuario.
  + Destino: Bases de datos.
  + Necesita: Acceso a la bases de datos.
  + Acción: Cambia los datos de un empleado.
  + Precondición: Que el empleado exista en la base de datos.
  + Postcondición: Los datos se han modificado correctamente.
  + Efectos colaterales: ninguno.

### **Función:** Mostrar empleado

* + Prioridad: Media.
  + Estabilidad: Alta.
  + Descripción: Muestra en la interfaz de usuario los datos del empleado.
  + Entrada: DNI.
  + Salida: Datos de empleado.
  + Origen: Interfaz de usuario.
  + Destino: Interfaz de usuario.
  + Necesita: Acceso a la base de datos.
  + Acción: Mostrar en la interfaz de usuario los datos del empleado.
  + Precondición: Que el empleado exista.
  + Postcondición: Que los datos mostrados sean los del empleado elegido.
  + Efectos colaterales: ninguno.

### **Función:** Mostrar empleados de una tarea

* + Prioridad: Media.
  + Estabilidad: Alta.
  + Descripción: Muestra todos los empleados asignados a cierta tarea.
  + Entrada: Id de la tarea.
  + Salida: Empleados asignados a la tarea determinada.
  + Origen: Interfaz de usuario.
  + Destino: Interfaz de usuario.
  + Necesita: Acceso a la base de datos.
  + Acción: Muestra datos de los empleados asignados a cierta tarea.
  + Precondición: Que la tarea exista.
  + Postcondición: Que se muestran los datos correctos de los empleados a la tarea elegida.
  + Efectos colaterales: ninguno.

### **Función:** Asignar tareas a un empleado

* + Prioridad: Alta.
  + Estabilidad: Media.
  + Descripción: Asigna tareas a un empleado.
  + Entrada: DNI del empleado, Id de la tarea a asignar.
  + Salida: Si la operación se ha llevado a cabo.
  + Origen: Interfaz de usuario.
  + Destino: Base de datos.
  + Necesita: Acceso a la base de datos.
  + Acción: Asignar una nueva tarea a un empleado.
  + Precondición: Que el empleado y la tarea existan en la base de datos., el empleado no debe tener asignada la tarea anteriormente.
  + Postcondición: El empleado tiene asignada la tarea.
  + Efectos colaterales: ninguno.

### **Función:** Mostrar todos los empleados

* + Prioridad: Media.
  + Estabilidad: Alta.
  + Descripción: Muestra todos los empleados.
  + Entrada:
  + Salida: Lista de empleados.
  + Origen: Interfaz de usuario.
  + Destino: Base de datos.
  + Necesita: Acceso a la base de datos.
  + Acción: Consulta todos los empleados.
  + Precondición:
  + Postcondición:
  + Efectos colaterales: ninguno.

## Tareas

### **Función:** Añadir tarea

* Prioridad: Media.
* Estabilidad: Alta.
* Descripción: Se da de alta una tarea.
* Entrada: Nombre de la tarea y la descripción.
* Salida: El Id asignado a la tarea.
* Origen: Interfaz de usuario.
* Destino: Base de datos.
* Necesita: Acceso a la base de datos.
* Acción: Añade una nueva tarea a la base de datos.
* Precondición: Que no exista una tarea con el mismo nombre.
* Postcondición: La tarea existe en la base de datos.
* Efectos colaterales: ninguno.

### **Función:** Borrar tarea

* Prioridad: Media.
* Estabilidad: Media.
* Descripción: Se elimina una determinada tarea.
* Entrada: Id de la tarea a eliminar.
* Salida: Si se ha podido eliminar la tarea.
* Origen: Interfaz de usuario.
* Destino: Base de datos.
* Necesita: Acceso a la base de datos.
* Acción: Borrar la tarea.
* Precondición: que la tarea no tenga empleados asignados.
* Postcondición: La tarea ya no existe en la base de datos.
* Efectos colaterales: ninguno.

### **Función:** Mostrar tarea

* Prioridad: Media.
* Estabilidad: Alta.
* Descripción: Se consultan los datos de una tarea.
* Entrada: Id de la tarea a consultar.
* Salida: Datos de la tarea.
* Origen: Interfaz de usuario.
* Destino: Interfaz de usuario.
* Necesita: Acceso a la base de datos.
* Acción: Consulta los datos de una tarea existente en la base de datos.
* Precondición: Id de la tarea existe en la base de datos.
* Postcondición: ninguna.
* Efectos colaterales: ninguno.

### **Función:** Mostrar todas las tareas

* Prioridad: Media.
* Estabilidad: Alta.
* Descripción: Se consultan todas las tareas de la base de datos.
* Entrada: ninguna.
* Salida: Tareas registradas en la base de datos.
* Origen: Interfaz de usuario.
* Destino: Interfaz de usuario.
* Necesita: Acceso a la base de datos.
* Acción: Consulta todas las tareas existentes en la base de datos.
* Precondición: ninguna
* Postcondición: ninguna.
* Efectos colaterales: ninguno.

### **Función:** Modificar tarea

* Prioridad: Media.
* Estabilidad: Media.
* Descripción: Se modifican los datos de una determinada tarea.
* Entrada: Id de la tarea y los datos a modificar.
* Salida: Si se ha podido realizar la modificación de los datos.
* Origen: Interfaz de usuario.
* Destino: Base de datos.
* Necesita: Acceso a la base de datos.
* Acción: Modificar una determinada tarea.
* Precondición: Id de tarea existente en la base de datos.
* Postcondición: Tarea modificada con los nuevos datos.
* Efectos colaterales: ninguno.

### **Función:** Tareas asignadas a un empleado.

* Prioridad: Alta.
* Estabilidad: Media.
* Descripción: Se muestran las tareas asignadas a un empleado.
* Entrada: Id del empleado.
* Salida: Tareas asignadas al empleado.
* Origen: Interfaz de usuario.
* Destino: Interfaz de usuario.
* Necesita: Acceso a la base de datos.
* Acción: Consulta las tareas asignadas a un empleado determinado.
* Precondición: Id del empleado existente en la base de datos.
* Postcondición: ninguna.
* Efectos colaterales: ninguno.

# Requisitos de rendimiento

El número de terminales en los que funcionará el software será únicamente de uno, y habrá un único usuario. El número de transacciones que se llevarán a cabo serán acorde con el uso de la aplicación, por lo que en principio es incalculable.

## Requisitos lógicos de la base de datos

Las funciones usarán números del tipo entero y real, además de cadenas de caracteres. Restringiremos en las entradas los números negativos, dado que están fuera del dominio del producto.

Nuestra política de retención de datos específica que guardaremos durante el periodo de 1 año los datos relacionados con nuestros clientes y reservas .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Clientes | Reservas | Habita  ciones | Departamentos | Empleados | Tareas |
| Clientes | X | 1:N | X | X | X | X |
| Reservas | N:1 | X | M:N | X | X | X |
| Habitaciones | X | N:M | X | X | X | X |
| Departamentos | X | X | X | X | 1:N | X |
| Empleados | X | X | X | N:1 | X | M:N |
| Tareas | X | X | X | X | N:M | X |

Para leerlo, hay que tener en cuenta que se lee por filas, por ejemplo, en la fila Clientes, un cliente puede hacer varias reservas.

## Requisitos funcionales

Nos atenemos a las reglas impuestas por el IEEE 830 – 1998, sujetos a las normas estipuladas por nuestro modelo unificado de Rational, es decir, llevaremos a cabo todas las fases del modelo en cada iteración de la realización del proyecto.

## Atributos del sistema de software

Este software se caracterizará por cumplir con los factores de calidad del software:

* ROBUSTEZ: capacidad del software para manejar situaciones excepcionales.
* CORRECCIÓN: capacidad para realizar las tareas descritas en la especificación.
* PORTABILIDAD: capacidad de transportar el código en vivo (serializable) e independiente de la plataforma. También de poder ejecutar la aplicación en distintas plataformas. Nuestro sistema al estar basado en Java tiene la portabilidad intrínseca de este lenguaje de programación.
* EFICIENCIA: capacidad para utilizar de forma correcta los recursos disponibles.
* INTEGRIDAD: característica del sistema para protegerse de elementos o hechos que no tengas acceso a este.
* FACILIDAD DE USO: facilidad de uso para cualquier usuario.
* VERIFICABILIDAD: facilidad para verificar el programa.
* COMPATIBILIDAD: capacidad de poder cambiar los diferentes elementos del software, frente a cambios de la especificación.
* EXTENSIBILIDAD: capacidad de ser escalable.
* REUTILIZACIÓN: posibilidad de usar el sistema o parte de él cuantas veces se quiera.
* FACILIDAD DE MANTENIMIENTO: capacidad del software de ser modificado o cambiado internamente.

En las primeras versiones del software es posible que existan algunos fallos menores que serán resueltas en cuanto se detecten. Nada que afecte gravemente al buen funcionamiento del programa. Su disponibilidad va a ser permanente, es decir, no habrá momentos de “no funcionamiento” en el software.