Desarrollo para Internet con tecnología Java

Sistema de reservas para hoteles

Cesar Ruiz Gorrochategui

ETIS

Consultor: Albert Grau Perisé

27 de Junio de 2004

1. Resumen

Desarrollo para Internet con tecnología Java

Sistema de reservas para hoteles

En este proyecto, se presenta la creación de un sistema de reservas para hoteles orientado a Internet utilizando tecnología Java.

En primer término se abordan los objetivos del proyecto, basándose en la exposición de las funcionalidades que son necesarias para la creación de reservas desde tres puntos de vista diferentes, el de un cliente, el de un empleado de hotel y el de un administrador del sistema.

Teniendo en cuenta los objetivos planteados, se pasa a estudiar el estado de la ciencia y las tecnologías aplicables a un proyecto basado en tecnología Java.

En especial, se exponen las tecnologías J2EE que se pueden aplicar desde dos vertientes diferentes. En primer lugar desde la vertiente de implementación con tecnologías J2EE y en segundo lugar desde la vertiente de arquitectura J2EE con las distintas soluciones aplicables para cada capa de negocio.

Además se hace una introducción a las diferentes tecnologías web que complementan o que dan soporte a un proyecto basado en tecnología Java.

Tras la exposición teórica de las distintas tecnologías aplicables, se desarrolla el proyecto llevado a cabo. En el apartado de arquitectura se explica el modelo seguido para crear el sistema de reservas, basado en patrones de diseño J2EE y utilizando el modelo “Model View Controller” adaptado a las necesidades del proyecto, al igual que la obtención de la información de la base de datos, para la cual se han utilizado clases ayudantes o manejadores.

En cuanto al diseño del proyecto se ha dividido en varias fases, empezando por la definición de los actores y sus funciones, seguido del análisis de los casos de uso para llegar a la obtención de las clases entidades y poder finalmente establecer los fundamentos para obtener un diagrama entidad relación, a partir del cual se ha creado la base de datos.

Se han definido como ayuda al desarrollo las interficies de usuario teniendo en cuenta los atributos obtenidos a lo largo de la fase de diseño. Finalmente y como resultado del proyecto se presentan dos vistas reales obtenidas en la fase de desarrollo.

Por último comentar que la planificación inicial ha contribuido a alcanzar los hitos principales del proyecto: *Sistema de reservas para hoteles*. La mencionada planificación consta de la explicación de las tareas fundamentales, de un grafo del algoritmo de tareas y de un diagrama de Gantt.

2. Objetivos

Los objetivos de este proyecto son implementar las funciones básicas para la creación, mantenimiento y consulta de reservas de habitaciones en una base de datos.

El sistema esta pensado para responder a las necesidades propias de un hotel y aspira a recoger los datos de la manera mas realista posible, desde los suplementos utilizados en las tarifas hoteleras, hasta las tarjetas de crédito del cliente.

El sistema constara de una serie de pantallas, bien estructuradas para facilitar la navegación por la interficie.

Además se tiene en cuenta la usabilidad del sistema, para poder crear un producto lo más amigable posible sin dejar de lado detalles de diseño.

El aplicativo dispondrá de tres vertientes:

- La primera de ellas ira dirigida al cliente externo, su interficie estará dedicada a la creación de una reserva, sin exceder de tres pasos o pantallas:
    1. Consulta de disponibilidad,
    2. Introducción de los datos del cliente,
    3. Confirmación de la reserva.

- La segunda vertiente va dirigida al cliente interno, con una interficie dedicada a la creación, modificación y consulta de reservas. El usuario dispondrá de un menú de selección de estas tres categorías mencionadas, siendo sus respectivos pasos o pantallas los siguientes:
    - Creación.
        1. Consulta de disponibilidad,
        2. Introducción de los datos del cliente,
        3. Confirmación de la reserva.
    - Modificación.
        1. Introducción de datos para la búsqueda,
        2. Consulta de la reserva,
        3. Modificación de datos,
        4. Confirmación de la reserva.
    - Consulta.
        1. Introducción de datos para la búsqueda,
        2. Consulta de la reserva.

- La tercera vertiente va dirigida al administrador del sistema, su interficie esta dedicada entonces a la administración del sistema. Creación, modificación y consulta de hoteles, habitaciones, tipos de habitación, tarifas y consulta de tablas en general. Su interficie constara de un menú donde podrá escoger sobre que entidad de la base de datos quiere actuar y que operación quiere realizar. Los pasos o pantallas a realizar son similares a los del cliente interno con cada una de las entidades.

3. Estado de la ciencia: Tecnologías aplicables

La metodología seguida para crear el proyecto ha supuesto un estudio de cada una de las tecnologías aplicables, esta es una parte importante del trabajo, por ser fundamental para la obtención de resultados satisfactorios y tecnológicamente correctos. Esta es la razón que da sentido a los tres siguientes apartados, dedicados íntegramente al estudio de las tecnologías aplicables, tecnologías J2EE, arquitectura J2EE y otras tecnologías aplicables.

3.1 Tecnologías J2EE Aplicables

3.1.1. Introducción a J2EE

Java 2 Enterprise Edition (J2EE) es una arquitectura multicapa, que contiene varias extensiones API que sirven de complemento a la edición estándar de Java (J2SE), de tal manera que se complementan las dos tecnologías para poder crear aplicaciones empresariales y orientadas a la web, y cubrir necesidades de un servidor. Las aplicaciones J2EE costan de varios componentes. Los componentes J2EE son unidades de software que se ensamblan en una aplicación J2EE, con la capacidad de comunicarse con otros componentes de la aplicación. Los componentes J2EE son los siguientes:

- Java Servlet la tecnología JavaServer Pages (JSP): componentes Web que se ejecutan en servidores.
- Enterprise JavaBeans (beans enterprise): componentes de negocio que se ejecutan en el servidor de aplicacion.
- Java Database Connectivity (JDBC), tecnología de acceso a sistemas de bases de datos relacionales.
- Java Transaction API (JTA) o Java Transaction Service (JTS) dan soporte en las transacciones a los componentes J2EE.
- Java Messaging Service (JMS) supone una comunicación con mensajes entre componentes J2EE.
- Java Naming y Directory Interface (JNDI) facilitan el acceso a nombres y directorios.

En los siguientes subapartados se extiende la información de las tecnologías J2EE más importantes para el desarrollo del proyecto, en concreto: Servlets y JSP, EJB, JDBC y Struct.

3.1.2. Servlets y JSP

Los servlets y JSP también son llamados componentes Web. Los servlets son clases Java con capacidad de procesar dinámicamente peticiones y construir respuestas. A su vez los JSP son documentos de texto que se ejecutan como servlets en un servidor de aplicaciones, pero a diferencia de los servlets permiten la creación de contenidos tanto estáticos como dinamicos. De esta forma se puede crear mediante una combinación de tecnologías como HTML, applets, JavaScript, XML, etc. y otras

tecnologías orientadas a diseño grafico, contenidos web mediante el ensamble de la aplicación J2EE. De forma análoga, las clases ubicadas en el servidor también se unen a los componentes web, pero estas no se han de considerar componentes J2EE.

3.1.3.JavaBean y EJB

J2EE dispone de contenedores, pensados especialmente para acceder a servicios del entorno del Servidor J2EE mediante estos contenedores podemos utilizar los diferentes componentes. De tal manera que no hay que desarrollar códigos encargados de manejar transacciones, estado, tratamiento de varios hilos de ejecución, etc. Con la tecnología de contenedor J2EE se gestionan estas necesidades, de tal manera que actualmente hay que resolver los problemas concretos de las aplicaciones, puesto que J2EE resuelve los servicios subyacentes.

Llegados a este punto tenemos que diferenciar los JavaBeans de los JavaBeans Enterprise (EJBs). A pesar de la similitud en sus nombres, son componentes bien distintos. Por una parte los JavaBeans se utilizan para comunicarse con otros JavaBeans dentro de una aplicación, además de con otros objetos. Por otro lado, los EJBs se comunican mediante espacios de direcciones multiples. Una gran ventaja de los EJBs es que permiten la reutilización, porque encapsulan lógica de negocio en componentes para este fin.

Finalmente cabe distinguir entre los tres diferentes tipos de EJBs:

- Beans de mensajes: pensados para la comunicación asíncrona mediante comunicaciones con JMS.

- Beans de sesión, si son de estado almacenan información de una sesión de llamadas entre métodos. Si no son de estado, no disponen de memoria y por tanto no pueden almacenar ninguna información sobre la sesión.

- Beans de entidad: el contenedor EJB (comentado al principio del apartado) da soporte para la administración de estos beans, que no son otra cosa sino objetos de base de datos en memoria, pudiendo gestionar la persistencia de estos beans o delegar esta tarea al propio controlador.

3.1.4.JDBC

Esta API proporciona las clases necesarias para la conexión de una aplicación a una base de datos de negocio. Para realizar esta conexión se utiliza una arquitectura de tres capas, la primera la aplicación Java con un formulario HTML en el cliente, el servidor web con el controlador JDBC y finalmente el controlador de base de datos en el servidor de base de datos. Con esta estructura la librería JDBC consigue que el cliente no tenga que disponer del driver JDBC, no obliga a tener separados los servidores web del correspondiente a la base de datos, aún siendo la mejor solución.

3.1.5.Struct

La API Struct, viene a relacionar las diferentes tecnologías mencionadas en este apartado, HTML, Servlets, JSP y JavaBeans. La manera combinar estas tecnologías es mediante el Modelo Vista Controlador, este modelo consiste en el uso de un

controlador central de flujo de navegación, de tal manera que el modelo engloba la lógica de negocio, el control de flujo influye en la generación de los vistas que tienen soporte de los manejadores.

Como vemos los Struct son un modelo de arquitectura completo que da solución a las diferentes problemáticas que tienen las tecnologías comentadas anteriormente por separado, combinándolas hasta la obtención de un modelo capaz de gestionar una web separando la lógica de negocio de los usuarios finales. Con esto se consigue modular la aplicación web de tal manera que los distintos componentes no tienen porque saber como realizan sus funciones el resto de componentes.

En el siguiente apartado se detalla cada uno de los modelos de arquitectura J2EE, que están estrechamente relacionados con Structs.

3.2. Arquitectura de proyecto J2EE

El diseño de la aplicación web basada en patrones J2EE utiliza varios elementos de diseño que se exponen a continuació.

- *Front Controller* / Controlador frontal
- *Dispatchers*
- Vistas, *JSPs*
- *Composite Views* / Vistas compuestas,
- *View helper* / helpers de las vistas, *JavaBeans*

3.2.1 Front Controller

Con este elemento se consigue centralizar la gestión de peticiones web, recibe las peticiones de clientes y las remite a los dispatchers correspondientes, encargados de la construcción de la respuesta para el cliente.

En este punto se crean los servicios de tratamiento de errores, gestión de control de generación de contenidos y seguridad.

En un contexto de gestión de peticiones distribuida en páginas JSP se produce duplicidad y distribución de código. La solución es el uso de controlador de gestión de peticiones. Se consigue reducir la cantidad de código Java de los JSPs (Scriptlets), porque se centraliza el control en el controlador y se reduce la lógica de negocio en las vistas. La implementación adecuada puede ser un Servlet.

Las principales consecuencias son la centralización del control de peticiones, mejora en el control de seguridad y aumento de la reusabilidad de código.

3.2.2 Dispatchers

En el diseño puede haber varios elementos de este tipo, se codificará la respuesta al cliente, realizando una composición de las vistas y configurando éstas para que muestren la información contestada a la petición de usuario.

El contexto en el que actuán los dispatchers es el control de flujo, acceso a procesamiento de presentación responsable de generación dinámica de contenidos.

Teniendo en cuenta el problema que suponen el no haber centralización de control de acceso, la recuperación de contenidos o gestión de vistas y que puede haber código duplicado en varias vistas. La lógica de negocio y formato de presentación se mezclan en las vistas, se pierde flexibilidad, reusabilidad, modularidad y se crea resistencias al cambio, y perdida de roles entre la producción web y el desarrollo de software.

La solución puede ser la combinación del Front Controller y el Dispatchers con vistas para manejar las peticiones y preparar la presentación dinámica. Un dispatchers, responsable de gestión de vistas, puede estar dentro de un controlador, en una vista o separado.

Un dispatcher puede ser un servlet que comprueba los parámetros que pasa el usuario y llama a la página JSP correspondiente con los parámetros necesarios para que se ejecute de retorno una respuesta.

Las principales consecuencias del uso de dispatchers es la centralización del control, mejora la reusabilidad, el mantenimiento, modula la aplicación separando roles.

3.2.3 Vistas

Generan información visual que responden a las peticiones de los usuarios, son JSP que estarán parametrizados para mostrar diferente información en base a los parámetros enviados. Los JSPs producen un parte de página web que ve el usuario.

Con el uso de vistas se consigue separar la visualización de la lógica de negocio, separando las plataformas de gestión de datos y el servidor web.

Los JSPs suelen estar compuestos de código HTML estático, etiquetas propias JSP, y en ocasiones código Java (Scriptlets). Estos se traducen una sola vez, excepto que éste cambie, generando un fichero de clase, que implementa la funcionalidad del JSP dentro de un Servlet.

3.2.4 *Composite View*

Este patrón es utilizado para que la presentación de vistas sea más manejable, puesto que se gestionan los elementos de una página mediante plantillas. Las páginas web son una combinación de elementos estáticos y dinámicos, cabeceras, pies, imágenes, etc. Los componentes estáticos son genéricos para todas las páginas no así los dinámicos, por tanto las plantillas de vista compuesta estandarizan los elementos estáticos comunes. Este patrón ha de ser un diseño de esquema que compone la página.

Hay que tener en cuenta que en general una web tiene una estructura similar para toda la web. Por tanto una aplicación web presenta contenidos de diferentes fuentes de datos, por esta razón hay que usar subvistas para la generación de una página.

La solución propuesta es utilizar vistas compuestas, que son creadas a partir de varias subvistas, de tal manera que cada componente de la plantilla es incluido dinámicamente en el total de la página, pudiendo gestionar el contenido con independencia a la presentación.

El uso de este patrón es necesario en webs con gran números de subvistas independientes, siendo sus principales consecuencias la mejora de la reutilización, la flexibilidad, modularidad, el mantenimiento y manejabilidad. Sin embargo provoca como inconveniente, que reduce el rendimiento, pero en una cantidad muy pequeña.

Para implementar esta solución hay que basarse en el uso de la etiqueta `<jsp:include>`, que permite incluir tanto contenidos estáticos como dinámicos.

3.2.5 View Helper

Encapsula la lógica Java correspondiente a la presentación y acceso a base de datos y componentes que necesitan una vista, consiguiendo vistas mas simples, reutilizables y fáciles de mantener. La lógica de presentación da formato a datos para su visualización, el acceso a datos implica la obtención de éstos.

Los View Helpers pueden ser JavaBeans, que son objetos Java bien definidos, poseen métodos de acceso para leer y modificar los valores de determinadas propiedades.

Para acceder a un Bean desde un JSP, hay que identificarlo y crear una referencia con la etiqueta:

```
<jsp:useBean ...>
```

Para obtener una propiedad de un Bean se usa la etiqueta:

```
<jsp:getProperty name=”nombre_bean” property=”variable_bean” />
```

para modificarla:

```
<jsp:setProperty name=”nombre_bean” property=”variable_bean” <%=expresion%>” />
```

los JavaBeans hacen uso de JDBC para el acceso a bases de datos o delegan esta labor llamando a otro componente.

En un contexto donde el sistema crea el contenido de la presentación, lo que requiere el procesamiento de datos de negocio dinámico; se presentan los siguientes problemas:

- Los cambios en la capa de presentación se dan con frecuencia y son difíciles de desarrollar y mantener cuando la lógica de acceso a los datos de negocio y la lógica del formato de la presentación están mezclados. Esto hace el sistema menos flexible, menos reutilizable, y menos adaptable a cambios.
- Mezclar la lógica de negocio y de sistema con el procesamiento de la vista reduce la modularidad y no proporciona separación de roles entre los equipos de producción Web y de desarrollo de software.

La solución propuesta es que la vista contiene código para dar formato, delegando a las clases de ayuda el procesamiento, implementadas como JavaBeans o etiquetas personalizadas. Las clases de ayuda también almacenan el modelo de datos intermedio de la vista. Como resultado de esta solución se consigue la mejora del particionamiento de la aplicación, la reutilización y el mantenimiento así como la separación de roles.

3.3. Otras tecnologías aplicables

3.3.1. HTML

La mayor parte de las páginas Web actuales han sido creadas con el lenguaje HTML (cabe comentar que cada vez tiene más fuerza este lenguaje de programación), mediante ficheros de texto modificado y utilizando este lenguaje de programación, que sigue un estándar marcado por una organización llamada consorcio WWW.

Las principales características que ofrece el HTML son las siguientes.

-Usa etiquetas sencillas y permite combinar texto y soporte grafico.
-Soporta etiquetas especificas para la creación de tablas.
-Facilita la creación de formularios.
-Visualización de una web mediante marcos ajustables.
-Permite introducir objetos Multimedia avanzados.
-Funcionalidades de diseño 3D.
-Inclusión de objetos animados.
-Posibilidad de incluir estilos dinámicos mediante CSS .
-Inclusión de código JavaScript.

3.3.2. CSS

Las hojas de estilo en cascada (CSS), son el núcleo principal de DHTML o HTML dinámico. Las CSS permiten modificar etiquetas HTML para adaptarlas a un esquema de diseño deseado. Se puede utilizar como archivos separados llamados hojas de estilo o embebidas en los propios archivos HTML.

3.3.3. JavaScript

JavaScript es un lenguaje de programación interpretado, que se utiliza en muchos entornos web para crear páginas dinámicas, aportando interactividad, respuesta e integración con programas externos o complementarios. Desgraciadamente desde un principio no se ha seguido un estándar y en la actualidad hay que tener en cuenta y diferenciar los navegadores que interpretan este lenguaje de scripts.

3.3.4. MySQL

MySQL es una de las bases de datos relacionales más apropiada para el desarrollo web por presentar una conectividad, velocidad y seguridad excelentes para este entorno de red. Esta base de datos sigue la filosofía de cliente – servidor, y por tanto tiene capacidades multihilo. Las características principales de MySQL son las siguientes.

- Pose conectividad con diferentes sistemas mediante ODBC.
- Es multiplataforma, soporta tanto plataforma Linux como Windows.
- Licencia gratuita no comercial.
- Alta velocidad de sus operaciones.
- Alta capacidad de almacenamiento.
- Soporta integridad referencial.
- Posee funciones para mejorar su seguridad.

3.3.5. TOMCAT

Tomcat es un contenedor web de Servlets y JSPs, que se integra a la perfección con el servidor web Apache y contenedores de aplicaciones como Jboss.

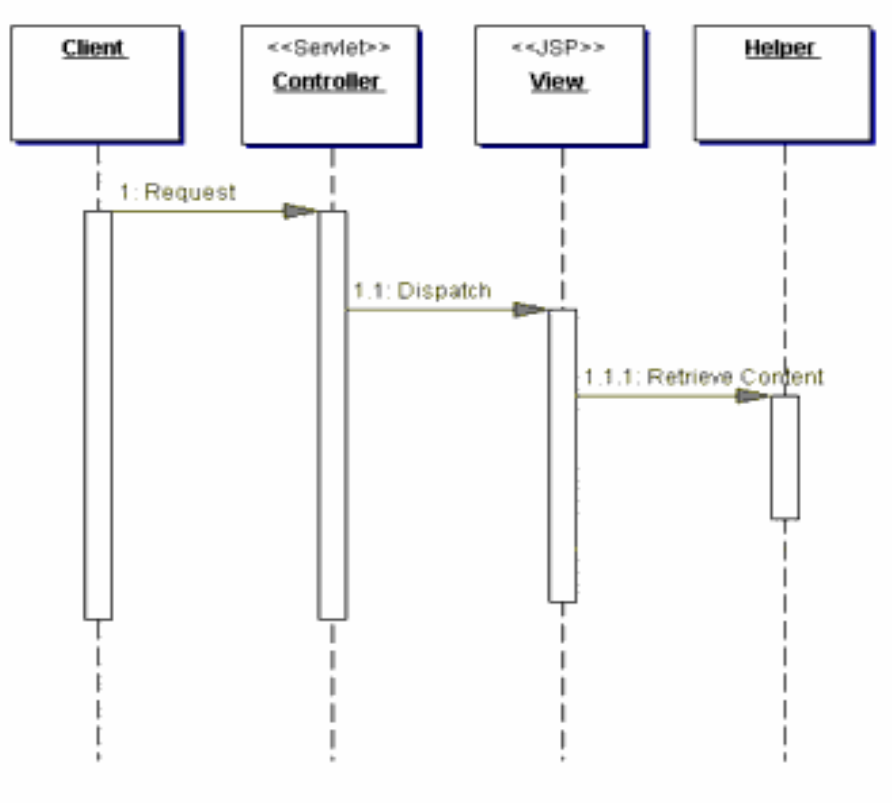
Es destacable, para mostrar la extensión de Tomcat como herramienta de negocio, que se está convirtiendo en parte de algunas herramientas de desarrollo.

Tras el estudio de la ciencia se ha pasado a crear una arquitectura adecuada al producto deseado y se ha realizado el diseño de dicho producto con todas las fases que esto supone. Para exponer estas dos fases fundamentales del proyecto se dedica el siguiente capitulo, Arquitectura y diseño del proyecto.

4. Arquitectura y Diseño del Proyecto

En este capítulo se pretende explica de manera sistemática el por qué de la arquitectura del proyecto, así como su diseño.

4.1 Arquitectura



El diseño de la web se basa en los patrones descritos a continuación, la única modificación que realizada en esta estructura es que no se utilizan EJBs sino clases de Java o manejadores encargados de realizar la labor que harían los EJBs.

Por tanto mi proyecto consta de:

- Un controlador frontal, *Front Controller (Servlet "Controler.java")*

- Los *dispatchers (Clase interna del Controler)*
- Las vistas compuestas, *Composite Views (Plantilla JSP "Plantilla.jsp")*
- Las vistas, *JSPs (JSPs varios.jsp)*
- Clases ayudantes. (Clases de conexión a BD, de acción sobre BD, etc)

El controller que se presenta valida al usuario tomando los valores de BD, valida que hay una sesión activa, controla el acceso a las páginas de tal manera que no se puede acceder a un JSP si no es a través de una validación de usuario, sesión y precedencia de JSP. (Ver Controller.java)

El dispatcher se encarga de preparar la página que se ha de servir tras una petición validada. Esto lo realiza mediante el método forward y la inserción en el request de los atributos necesarios. (Ver Controller.java clase interna Dispatch)

Las vistas compuestas, se crean tomando los atributos pasados por el dispatcher, por la plantilla que interpreta que archivos ha de incluir, pudiendo ser tanto estáticos HTML, como dinámicos JSP. (Ver Plantilla.jsp)

Plantilla, Plantilla.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java"
import="java.sql.*,sisres.*" errorPage="" %>

<%
        String sLinks, sForm, sArgs, sUser;
        sLinks =(String)request.getAttribute("links");
        sForm =(String)request.getAttribute("form");
        sArgs =(String)request.getAttribute("args");
        sUser =(String)request.getAttribute("user");

        Conexion db = new Conexion();
        Statement sentencia = db.getConexion().createStatement();
        Sesion ses = new Sesion();

        ses.consultaClienteXSesion(request.getRequestedSessionId());
        String nom = ses.getNombreCliente();
        String ap1 = ses.getApellido1Cliente();
        String ap2 =ses.getApellido2Cliente();
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Plantilla</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link href="../estilos/prueba1.css" rel="stylesheet" type="text/css">
<script language="JavaScript" src="../javascript/Validaciones.js" type="text/javascript"></script>
<script language="JavaScript" src="../javascript/submit_02.js" type="text/javascript"></script>
</head>
<body>
<table width="100%"  border="0">
  <tr>
    <td></td>
    <td width="800" valign="top">
        <table width="100%" border="1" cellpadding="0" cellspacing="0" class="ficha">
        <tr>
        <td height="10" class="label1">Hola, <%=nom%> <%=ap1%> <%=ap2%> usted es un
usuario de tipo "<%=sUser%>"</td>
        </tr>
        <tr>
```

```
          <td width="100%" height="100%">
          <jsp:include page="<%=sLinks%>" flush="true" />
          </td>
          </tr>
          </table>
          </td>
  </tr>
  <tr>
    <td height="30"> </td>
    <td width="800">
<table width="800" height="200" border="0" cellpadding="0" cellspacing="0">
 <tr>
    <td width="9" height="9" valign="top"><img src="../imagenes/empty.gif" width="9"
height="1"></td>
    <td width="4" rowspan="2" valign="top"><img src="../imagenes/empty.gif" alt="" width="2"
height="1"></td>
    <td width="130" rowspan="2" valign="top" class="labelBox">SIS RES</td>
    <td width="656" valign="top"><img src="../imagenes/empty.gif" alt="" width="656"
height="1"></td>
    <td width="9" valign="top"><img src="../imagenes/empty.gif" alt="" width="9" height="1"></td>
  </tr>
  <tr>
    <td width="9" height="9" valign="top"><img src="../imagenes/fichas/azul_sup_izda.gif" alt=""
width="9" height="9"></td>
    <td valign="top" class="BordeSupAzul"><img src="../imagenes/empty.gif" alt="" width="9"
height="1"></td>
    <td width="9" valign="top"><img src="../imagenes/fichas/azul_sup_dcha.gif" width="9"
height="9"></td>
  </tr>
  <tr>
    <td width="9" height="173" class="BordeIzqAzul"><img src="../imagenes/empty.gif" alt=""
width="9" height="1"></td>
    <td> </td>

    <td colspan="2" valign="top">
                  <table width="776" border="0" cellpadding="0" cellspacing="0">
                  <tr>
                  <td width="776" height="175">
          <table  height="100%" width="100%" border="0" cellpadding="0" cellspacing="0">
                  <tr>
                        <td height="20" colspan="3">
<jsp:include page="<%= sForm %>" flush="true" />
                        </td>
                  </tr>
                  <tr>
                        <td height="260" colspan="3" valign="top">
<iframe src="../html/blanco.htm"  name="detalle_hotel" width="775" marginwidth="0" height="100%"
marginheight="0" align="middle"  scrolling="yes" frameborder="0"  vspace="0" hspace="0"
class="BarraBeige"></iframe>
                        </td>
                  </tr>
          </table>
</tr>
 </table>
</td>
</tr>
    <td width="9" class="BordeDchoAzul"><img src="../imagenes/empty.gif" alt="" width="8"
height="1"></td>
  </tr>
  <tr>
    <td width="9" height="9" valign="bottom"><img src="../imagenes/fichas/azul_inf_izda.gif" alt=""
width="9" height="9"></td>
    <td height="9" colspan="3" align="left" valign="bottom" class="BordeInfAzul"><img
src="../imagenes/empty.gif" alt="" width="2" height="1"></td>
```

```
   <td width="9" valign="top" align="right"><img src="../imagenes/fichas/azul_inf_dcha.gif" alt=""
width="9" height="9"></td>
  </tr>
</table>
          </td>
   <td height="30"> </td>
  </tr>
</table>
</body>
</html>
```

Los JSPs son partes de un conjunto que se cristaliza a través de la plantilla, de tal manera que éstos tienen el mínimo código necesario para ejecutarse. Además cualquier funcionalidad que tengan que realizar está delegada a las clases ayudantes. (Ver *.jsp)

Delegación de acciones a los helpers, extracto de disponibilidad.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java" import="sisres.*"
errorPage="" %>

<%
String hotel = request.getParameter("p_hotel");
String fecini = request.getParameter("p_fecini");
String numnoc = request.getParameter("p_numnoc");
String tiphab = request.getParameter("p_tiphab");
String regime = request.getParameter("p_regime");
String adulto = request.getParameter("p_adulto");
String menore = request.getParameter("p_menore");

Disponibilidad dis = new Disponibilidad();
if (dis.disponibleEnFechas(hotel, tiphab, fecini, Utiles.stringToInt(numnoc))) {
double precio = dis.precioConsulta(hotel, tiphab, fecini, Utiles.stringToInt(numnoc));

Hotel hot = new Hotel();
Listado lisHot = (Listado)hot.getDatosHotelPorCodigo(hotel);

Sesion ses = new Sesion();
ses.consultaClienteXSesion(request.getRequestedSessionId());
String nom = ses.getNombreCliente();
String ap1 = ses.getApellido1Cliente();
String ap2 =ses.getApellido2Cliente();
%>
...

...
```

Las Clases ayudantes se encargan de las funciones pesadas de proceso, como la conexión a base de datos, acciones sobre entidades o cálculos intensivos de tal manera que los JSPs sólo tienen que instanciar a la clases que necesitan y pasarle los valores necesarios. (Ver *.java)

Clase ayudante de conexión a base de datos, Conexion.java

```
package sisres;

import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.SQLException;


/**
 * @author César
 *
 */
public class Conexion {

	static String URL="jdbc:mysql://localhost/sisres_des";
	static String userID="user";
	static String password="";
	static Connection con = null;

	public Conexion(){

		try	{
			Class.forName("com.mysql.jdbc.Driver");
			con=
DriverManager.getConnection(URL,userID,password);
		}

		catch (ClassNotFoundException error1){
			System.err.println("No se puede cargar JDBC/ODBC");
			System.exit(1);
		}
		catch(SQLException error2){
			System.err.println("No se puede conectar a Base de
Datos. "+ error2);
			System.exit(2);
		}
	}

	public Connection getConexion() {
		return con;
	}

	public void closeConexion(){
		try {
			con.close();
		}
		catch (SQLException error3) {
			error3.printStackTrace();
		}
	}
}
```

Ejecución del Proyecto
http://localhost:8081/SISRES/reshot/

La ejecución en la barra direcciones del navegador de: http://localhost:8081/SISRES/reshot/, supone que no es necesario introducir ninguna página .html o jsp, porque el controller desde el primer momento rige la navegación. Además, una vez validado el usuario, la introducción de páginas en la barra de direcciones provocará la desconexión de la sesión, por producirse una violación del flujo de navegación.

A continuación se exponen las diferentes partes del diseño de la aplicación, desde la definición de los actores que intervienen hasta la generación de pantallas, pasando por la creación de casos de uso y de la base de datos.

4.2 Diseño

Para definir el dominio tenemos que el proyecto consiste en realizar un sistema de reservas para un hotel. De tal manera que una reserva de un cliente es para un hotel, con un tipo de habitación y con un régimen de alojamiento.

4.2.1 Los actores

En este proyecto son el cliente, el recepcionista, el administrador del sistema.

4.2.1.1 El guión del cliente

El cliente desea realizar una reserva para una fecha y unas noches determinadas en un hotel, para un cierto numero de personas y para un tipo de habitación con un régimen. El cliente, puede seleccionar que hay menores de 2 años o menores entre 2 y 12 años. De todas maneras el cliente sabrá si dispone de lo solicitado en su reserva.
En caso de haber disponibilidad y estar de acuerdo el cliente con el precio ofrecido, éste tendrá que introducir sus datos: nombre, apellido, teléfono, móvil, email, dirección, código postal, tarjeta de crédito, fecha caducidad tarjeta y tipo de tarjeta, además de un campo abierto a observaciones. Tras confirmar sus datos la reserva ya estará introducida en el sistema.

4.2.1.2 El guión del recepcionista

El recepcionista puede realizar las acciones de creación, modificación, cancelación, consulta de reservas y consulta de disponibilidad. El recepcionista puede escoger en un menú una cualquiera de estas categorías mencionadas.

Creación: La creación de la reserva se realiza de igual modo que la efectuada por el cliente.

Modificación: El recepcionista recibe una petición de modificación de una reserva. Para buscarla introduce el apellido y/o la fecha de entrada y/o el nombre del cliente o en su caso el localizador y le aparecen los datos de la reserva, que podrá modificar. Después confirma la modificación y se introduce la modificación en el sistema.

Cancelación: El recepcionista recibe una cancelación de reserva, para buscarla introduce el apellido y/o la fecha de entrada y/o el nombre del cliente o en su caso el localizador. Tras esto le aparecen los datos de la reserva, que podrá cancelar. Finalmente, se confirma el borrado y la reserva desaparece del sistema.

Consulta: El recepcionista recibe solicitud de consulta de una reserva. En primer lugar la busca por apellido y/o la fecha de entrada y/o el nombre del cliente o en su caso el localizador y le aparecen los datos de la reserva.

Disponibilidad: El recepcionista introduce la fecha de entrada, la fecha de salida o número de noches y/o puede el tipo de habitación y como resultado aparecen los datos de disponibilidad.

4.2.1.3 El guión del administrador del sistema

El administrador del sistema puede realizar las acciones de creación, modificación, baja y consulta las entidades del sistema. Este actor puede escoger de un menú una de estas categorías mencionadas, así como la entidad en la que quiere actuar.

Su actuación sobre las entidades se realiza en el sentido de dar de alta, baja, modificar o consulta de hotel, tipos de habitación, etc.

Alta: Alta de un nuevo miembro de una entidad.

Baja: Consulta del miembro y baja de éste de una entidad (borrado de todos sus atributos en cascada).

Modificación: Consulta del miembro y modificación de sus atributos.

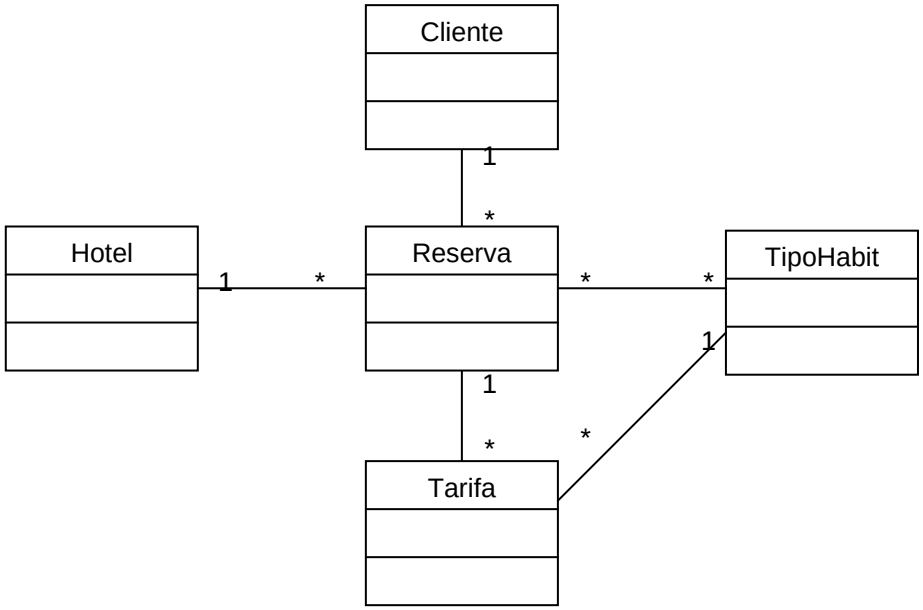
Consulta: Consulta de los atributos de un miembro de una entidad. Una de las funciones principales del administrador del sistema es la introducción de las tarifas. Éstas tienen que tener en cuenta los valores de la reserva: tipo de habitación, numero de adultos / de menores de 2 años / de menores entre 2 y 12 años y el periodo de estancia así como el régimen.

El administrador puede realizar las mismas funciones que el recepcionista.

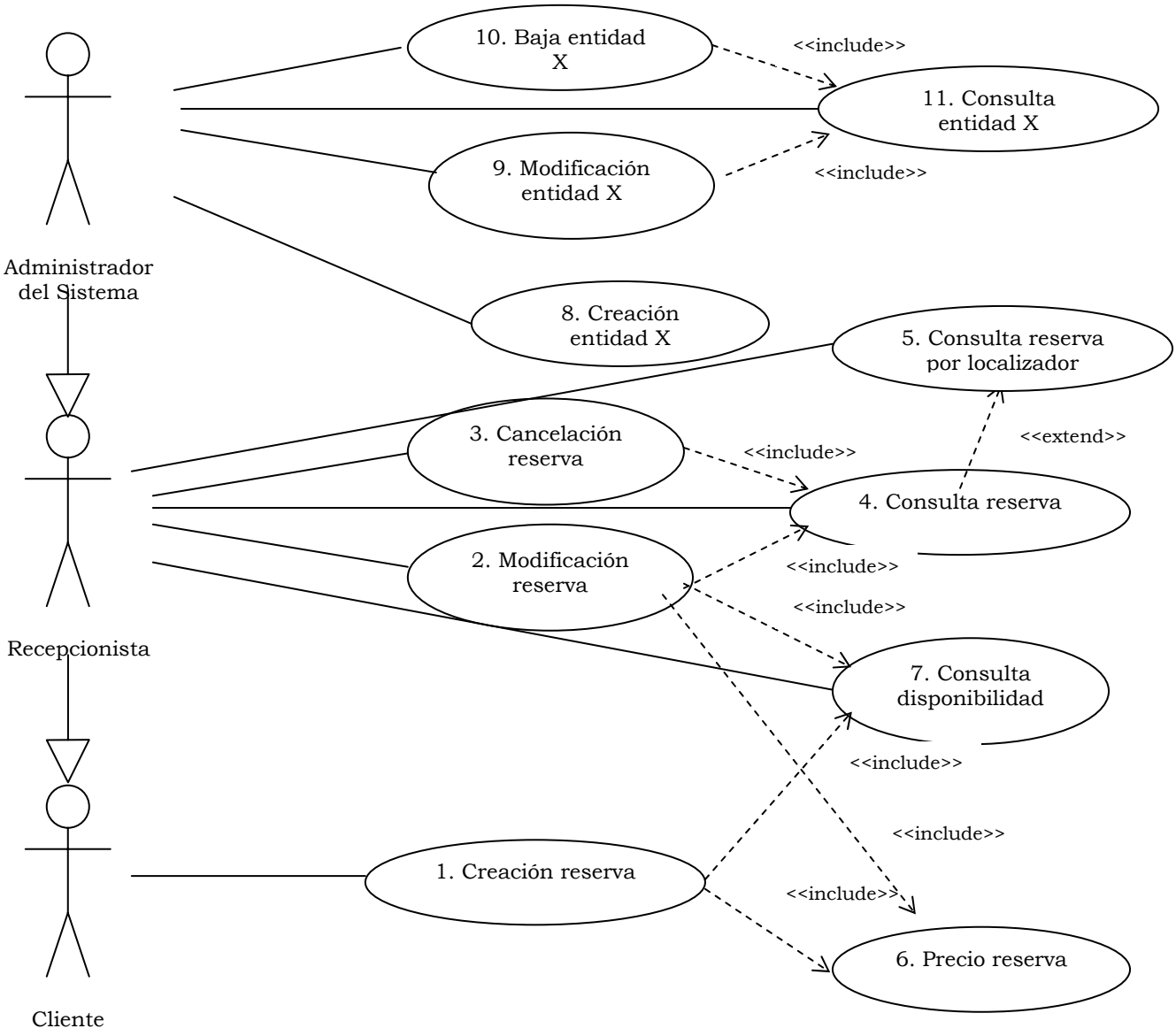
Después de definir a los actores y sus funciones, estamos en condiciones de presentar un diagrama de dominio básico, como se expone en el siguiente apartado, tras éste diagrama de dominio tenemos el diagrama de casos de uso que se desprende del estudio de los actores.

4.2.2 Diagrama de dominio

Este diagrama de dominio se basa en la información dada en el apartado anterior.



4.2.3 Diagrama de casos de uso de requisitos



Ahora que se ha presentado el diagrama de casos de uso de requisitos si puede realizar un estudio más profundo de dichos casos de uso, este estudio se aborda en el siguiente apartado.

4.2.4 Casos de Uso

Caso de uso: 1.Creación reserva

Resumen de la funcionalidad: introduce una reserva.

Papel dentro del trabajo Cliente: básico.

Actores: Cliente, Recepcionista, Administrador.

Casos de uso relacionados: 7.Consulta disponibilidad, 6.Precio reserva

Precondición: ninguna.

Postcondición: se ha realizado una reserva.

Proceso normal principal:
El sistema pide el *hotel, fecha de llegada, número noches, tipo de habitación, régimen, número de adultos, número de menores de 2 años, número de menores de 2 a 12 años*.
El cliente introduce los datos.
El sistema ejecuta el caso de uso 7. consulta la disponibilidad y obtiene el precio ejecutando el caso de uso 6. precio reserva para estos datos.
El sistema oferta un *precio*, y solicita el *nombre, apellido, teléfono, móvil, email, dirección, provincia, país, código postal, número de tarjeta de crédito, fecha caducidad, tipo de tarjeta y un campo de observaciones.*
El cliente acepta el *precio*, introduce los datos.
El sistema guarda la *reserva* y resta en uno la *disponibilidad* para ese *tipo de habitación*.

Alternativas de proceso y excepciones:
2a. El cliente no ha introducido datos obligatorios.
    2a1. El sistema solicita los datos obligatorios.
3a. Hay un exceso de personas para una habitación, el sistema da un aviso y vuelve a pedir los datos.
3b. No hay disponibilidad para ese tipo de habitación.
3c. La fecha de entrada esta fuera de rango o el número de noches es demasiado alto para este hotel.
5a. El cliente no acepta la reserva.
5b. El cliente no ha introducido datos obligatorios.
    5b1. El sistema solicita los datos obligatorios.

Caso de uso: 2. Modificación de reserva

Resumen de la funcionalidad: modifica los datos de una reserva.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Recepcionista, Administrador.

Casos de uso relacionados: 4.Consulta reserva, 5.Consulta reserva por localizador, 6.Precio reserva, 7.Consulta disponibilidad.

Precondición: ninguna.

Postcondición: se ha modificado la reserva.

Proceso normal principal:
El sistema solicita el *hotel, nombre del cliente, apellido, fecha de entrada y localizador*.
El recepcionista introduce los datos.
El sistema muestra las coincidencias con los datos introducidos, realiza esto utilizando los casos de uso 4.Consulta reserva y 5.Consulta reserva por localizador.
El recepcionista modifica los datos de la reserva.
El sistema ejecuta el caso de uso 7. consulta la disponibilidad y obtiene el precio ejecutando el caso de uso 6. precio reserva para estos datos, en caso de que esto sea necesario.
El sistema pide confirmación de modificación.
El recepcionista acepta la modificación.
El sistema guarda la *reserva.*

Alternativas de proceso y excepciones:
2a. El recepcionista introduce el localizador.
2a1. El sistema ejecuta el caso de uso 5.Consulta reserva por localizador.
2b. El recepcionista introduce otros datos que no son el localizador.
2b1. El sistema ejecuta el caso de uso 4.Consulta reserva.
2c. El recepcionista no ha introducido datos obligatorios.
    2c1. El sistema solicita los datos obligatorios.
3a. El sistema muestra la reserva del localizador dado.
3b. El sistema muestra una lista de casos coincidentes.
3b1. El recepcionista escoge uno de los casos.
3b2. El sistema muestra la reserva.
3b3. Si no es ésa, el recepcionista retorna a la lista.
4a. El recepcionista modifica datos que afectan a la disponibilidad.
4a1. El sistema va al paso 5 ejecuta los casos de uso 7. consulta la disponibilidad y 6. precio reserva.
4b. El recepcionista modifica datos que no afectan a la disponibilidad.
    4b1. El sistema va al paso 6.
4c. El recepcionista no ha introducido datos obligatorios.
    4c1. El sistema solicita los datos obligatorios.
5a. Hay un exceso de personas para una habitación, el sistema da un aviso y vuelve a pedir los datos.
5b. No hay disponibilidad para ese tipo de habitación.
5c. La fecha de entrada o el número de noches esta fuera de rango para este hotel.
7a. El recepcionista no acepta la modificación.
    7a1. El sistema no modifica la reserva.

Caso de uso: 3. Cancelación de reserva

Resumen de la funcionalidad: Cancela una reserva.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Recepcionista, Administrador.

Casos de uso relacionados: 4.Consulta reserva, 5.Consulta reserva por localizador.

Precondición: Ninguna.

Postcondición: se ha eliminado la reserva.

Proceso normal principal:
El sistema solicita el *hotel, nombre del cliente, apellido, fecha de entrada y localizador*.
El recepcionista introduce los datos.
El sistema muestra las coincidencias con los datos introducidos, realiza esto utilizando los casos de uso 4.Consulta reserva y 5.Consulta reserva por localizador.
El recepcionista cancela la reserva.
El sistema elimina la reserva y suma en una unidad el tipo de habitación para las fechas de la reserva.

Alternativas de proceso y excepciones:
2a. El recepcionista introduce el localizador.
2a1. El sistema ejecuta el caso de uso 5.Consulta reserva por localizador.
2b. El recepcionista introduce otros datos que no son el localizador.
2b1. El sistema ejecuta el caso de uso 4.Consulta reserva.
2c. El recepcionista no ha introducido datos obligatorios.
    2c1. El sistema solicita los datos obligatorios.
3a. El sistema muestra la reserva del localizador dado.
3b. El sistema muestra una lista de casos coincidentes.
3b1. El recepcionista escoge uno de los casos.
3b2. El sistema muestra la reserva.
3b3. Si no es ésa, el recepcionista retorna a la lista.

Caso de uso: 4. Consulta de reserva

Resumen de la funcionalidad: Consulta los datos de una reserva.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Recepcionista, Administrador.

Casos de uso relacionados: 2. Modificación de reserva, 3. Cancelación de reserva, 5.Consulta reserva por localizador

Precondición: Ninguna.

Postcondición: se ha mostrado la información de la reserva.

Proceso normal principal:
1. El sistema solicita el *hotel, nombre del cliente, apellido, fecha de entrada y localizador*.
El recepcionista introduce los datos.
El sistema muestra las coincidencias con los datos introducidos.

Alternativas de proceso y excepciones:
2a. El recepcionista introduce el localizador.
2a1. El sistema ejecuta el caso de uso 5.Consulta reserva por localizador.
2b. El recepcionista introduce otros datos que no son el localizador.
2b1. El sistema ejecuta el caso de uso 4.Consulta reserva.
2c. El recepcionista no ha introducido datos obligatorios.
    2c1. El sistema solicita los datos obligatorios.
3a. El sistema muestra la reserva del localizador dado.
3b. El sistema muestra una lista de casos coincidentes.
3b1. El recepcionista escoge uno de los casos.
3b2. El sistema muestra la reserva.
3b3. Si no es ésa, el recepcionista retorna a la lista.

Caso de uso: 5. Consulta de reserva por localizador

Resumen de la funcionalidad: Consulta los datos de una reserva por localizador.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Recepcionista, Administrador.

Casos de uso relacionados: 4.Consulta reserva

Precondición: ninguna.

Postcondición: se ha mostrado la información de la reserva.

Proceso normal principal:
1. El sistema solicita el *hotel, nombre del cliente, apellido, fecha de entrada y localizador*.
El recepcionista introduce los datos.
El sistema muestra las coincidencias con los datos introducidos.

Alternativas de proceso y excepciones:
2a. El recepcionista introduce el localizador.
2a1. El sistema ejecuta el caso de uso 5.Consulta reserva por localizador.
2b. El recepcionista introduce otros datos que no son el localizador.
2b1. El sistema ejecuta el caso de uso 4.Consulta reserva.
2c. El recepcionista no ha introducido datos obligatorios.
2c1. el sistema solicita los datos obligatorios.
3a. El sistema muestra la reserva del localizador dado.
3b. El sistema muestra una lista de casos coincidentes.
3b1. El recepcionista escoge uno de los casos.
3b2. El sistema muestra la reserva.
3b3. Si no es ésa, el recepcionista retorna a la lista.

Caso de uso: 6. Precio Reserva

Resumen de la funcionalidad: Retorna el precio de una reserva.

Papel dentro del trabajo Recepcionista: auxiliar

Actores: Recepcionista, Administrador.

Casos de uso relacionados: 1.Creación reserva, 2.Modificación reserva.

Precondición: Ha empezado la ejecución los caso de uso 1.Creación reserva, 2.Modificación reserva.

Postcondición: Retorna el precio de una reserva para unos valores dados.

Proceso normal principal:
El sistema ejecuta 1.Creación reserva o 2.Modificación reserva, que mandan los datos *hotel, fecha de llegada, número noches, tipo de habitación, régimen, número de adultos, número de menores de 2 años, número de menores de 2 a 12 años*.
El sistema calcula el precio de la reserva mediante los datos introducidos, consultando las diferentes tarifas por noche para el periodo de fechas de inicio y fin de la reserva y retorna el resultado a los casos de uso 1.Creación reserva o 2.Modificación reserva.

Alternativas de proceso y excepciones: Ninguna.

Caso de uso: 7. Consulta disponibilidad

Resumen de la funcionalidad: Retorna la disponibilidad para un tipo de habitación.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Recepcionista, Administrador.

Casos de uso relacionados: 1.Creación reserva, 2.Modificación reserva.

Precondición: ninguna o ha empezado la ejecución los caso de uso 1.Creación reserva, 2.Modificación reserva.

Postcondición: Retornada la disponibilidad para un tipo de habitación.

Proceso normal principal:
El sistema ejecuta 1.Creación reserva o 2.Modificación reserva. Que mandan los datos el *hotel, fecha de llegada, número noches, tipo de habitación, régimen, número de adultos, número de menores de 2 años, número de menores de 2 a 12 años* a este caso de uso 6. Precio Reserva.
El sistema calcula la disponibilidad mediante los datos pasados y retorna el resultado a los casos de uso 1.Creación reserva o 2.Modificación reserva.

Alternativas de proceso y excepciones:
1a. Si solicita el recepcionista consulta de disponibilidad el sistema le solicita los datos *hotel, fecha de llegada, numero noches, tipo de habitación, régimen, numero de adultos, numero de menores de 2 años, numero de menores de 2 a 12 años*. El recepcionista introduce los datos.
2a. El sistema calcula la disponibilidad mediante los datos introducidos y retorna el resultado al recepcionista.

Los siguientes casos de uso son genéricos para todas las entidades básicas de la base de datos.

Caso de uso: 8. Creación entidad X

Resumen de la funcionalidad: Crea un nuevo miembro de la entidad X.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Administrador.

Casos de uso relacionados: Ninguno.

Precondición: Ninguno.

Postcondición: Se ha creado un nuevo miembro de la entidad X.

Proceso normal principal:
El sistema solicita los atributos de la entidad X.
El administrador introduce los atributos de la entidad X.

Alternativas de proceso y excepciones:
2a. El cliente no ha introducido datos obligatorios.
    2a1. El sistema solicita los datos obligatorios.

Caso de uso: 9. Modificación entidad X

Resumen de la funcionalidad: Modifica al miembro de la entidad X.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Administrador.

Casos de uso relacionados: 11. Consulta entidad X

Precondición: Ninguno.

Postcondición: Se ha modificado un miembro de la entidad X.

Proceso normal principal:
El sistema solicita los atributos de la entidad X a modificar.
El administrador introduce los datos.
El sistema muestra las coincidencias con los datos introducidos, realiza esto utilizando los casos de uso 11.Consulta reserva
El administrador modifica los datos de la entidad X.
El sistema guarda la modificación.

Alternativas de proceso y excepciones:
2a. El administrador no ha introducido datos obligatorios.
    2a1. El sistema solicita los datos obligatorios.
3b. El sistema muestra una lista de casos coincidentes.
3b1. El administrador escoge uno de los casos.
3b2. El sistema muestra la entidad.
3b3. Si no es esa el administrador retorna a la lista.
4a. El recepcionista no ha introducido datos obligatorios.
    4a1. El sistema solicita los datos obligatorios.

Caso de uso: 10. Baja entidad X

Resumen de la funcionalidad: Baja del miembro de la entidad X.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Administrador.

Casos de uso relacionados: 11. Consulta entidad X

Precondición: Ninguno.

Postcondición: Se ha dado de baja un miembro de la entidad X.

Proceso normal principal:
El sistema solicita los atributos de la entidad X a modificar.
El administrador introduce los datos.
El sistema muestra las coincidencias con los datos introducidos, realiza esto utilizando los casos de uso 11.Consulta reserva
El administrador da de baja al miembro de la entidad X.
El sistema elimina al miembro de la entidad.

Alternativas de proceso y excepciones:
2a. El administrador no ha introducido datos obligatorios.
    2a1. el sistema solicita los datos obligatorios.
3b. El sistema muestra una lista de casos coincidentes.
3b1. El administrador escoge uno de los casos.
3b2. El sistema muestra la entidad.
3b3. Si no es este el miembro, el administrador retorna a la lista.

Caso de uso: 11. Consulta entidad X

Resumen de la funcionalidad: Consulta los datos de un miembro de la entidad X.

Papel dentro del trabajo Recepcionista: ocasional.

Actores: Administrador.

Casos de uso relacionados: 9. Modificación entidad X, 10. Baja de la entidad X

Precondición: Ninguno.

Postcondición: retorna los datos de un miembro de la entidad X.

Proceso normal principal:
El sistema solicita los atributos de la entidad X a modificar.
El administrador introduce los datos.
El sistema muestra las coincidencias con los datos introducidos.

Alternativas de proceso y excepciones:
2a. El administrador no ha introducido datos obligatorios.
    2a1. el sistema solicita los datos obligatorios.
3b. El sistema muestra una lista de casos coincidentes.
3b1. El administrador escoge uno de los casos.
3b2. El sistema muestra la entidad.
3b3. Si no es el miembro, el administrador retorna a la lista.

4.2.5 Obtención de clases de entidades

Una vez definidos todos los casos de uso se pasa a obtener las clase de entidades que aparecen en el proyecto.

Del caso de uso 1. Creación de reserva

Reserva, hotel, fecha llegada, número de noches, tipo de habitación, régimen, número de adultos, número de menores de 2 años, número de menores entre 2 y 12 años, cliente, disponibilidad, precio, nombre, apellido, teléfono, móvil, email, dirección, provincia, país, código postal, número tarjeta de crédito, fecha caducidad tarjeta de crédito, tipo de tarjeta de crédito, observaciones, número máximo de personas por habitación, fecha máxima de entrada, número máximo de noches, Valores Reserva.

Del caso de uso 2. Modificación de reserva

Reserva*, hotel*, fecha llegada*, nombre*, apellido*, localizador, disponibilidad*, precio*, número máximo de personas por habitación*, fecha máxima de entrada*, número máximo de noches*, Valores Reserva*.

Del caso de uso 3. Cancelación de reserva

Reserva*, hotel*, fecha llegada*, nombre*, apellido*, localizador*, tipo de habitación*.

Del caso de uso 4. Consulta de reserva

Reserva*, fecha llegada*, nombre*, apellido*, localizador*.

Del caso de uso 5. Consulta de reserva por localizador

Reserva*, hotel*, fecha llegada*, nombre*, apellido*, localizador*.

Del caso de uso 6. Precio reserva

Reserva*, hotel*, precio*, fecha llegada*, número de noches*, tipo de habitación*, régimen*, número de adultos*, número de menores de 2 años*, número de menores entre 2 y 12 años*, tarifa, precio por noche, fecha inicio, fecha fin.

Del caso de uso 7. Consulta disponibilidad

Reserva*, hotel*, disponibilidad*, fecha llegada*, número de noches*, tipo de habitación*, régimen*, número de adultos*, número de menores de 2 años*, número de menores entre 2 y 12 años*, precio*, tarifa.

De los casos de uso 8 a 11 tratan entidades X

Las posibles clases dependen de las entidades que salgan en esta fase del análisis.

Eliminaciones:

Hotel,
Nombre es un atributo **Hotel**,
fecha máxima de entrada es un atributo **Hotel**,
número máximo de noches es un atributo de **Hotel**,

Reserva,
fecha llegada es un atributo de **Reserva**,
número de noches es un atributo de **Reserva**,
número de adultos es un atributo de **Reserva**,
número de menores de 2 años es un atributo de **Reserva**,
número de menores entre 2 y 12 años es un atributo de **Reserva**,
observaciones es un atributo de **Reserva**,
localizador es un atributo de **Reserva**,
precio es un atributo de **Reserva**,
Régimen es un atributo de **Reserva**,

Tipo de habitación,
tipo es un atributo de **Tipo de Habitación**,
Num. Máx. de personas por hab. es un atributo de **Tipo de Habitación**,
disponibilidad es un atributo de **Tipo de Habitación**,

Cliente,
nombre es un atributo de **Cliente**,
apellido es un atributo de **Cliente**,
dirección es un atributo de **Cliente**,
provincia es un atributo de **Cliente**,
país es un atributo de **Cliente**,
código postal es un atributo de **Cliente**,
teléfono es un atributo de **Cliente**,
móvil es un atributo de **Cliente**,
email es un atributo de **Cliente**,

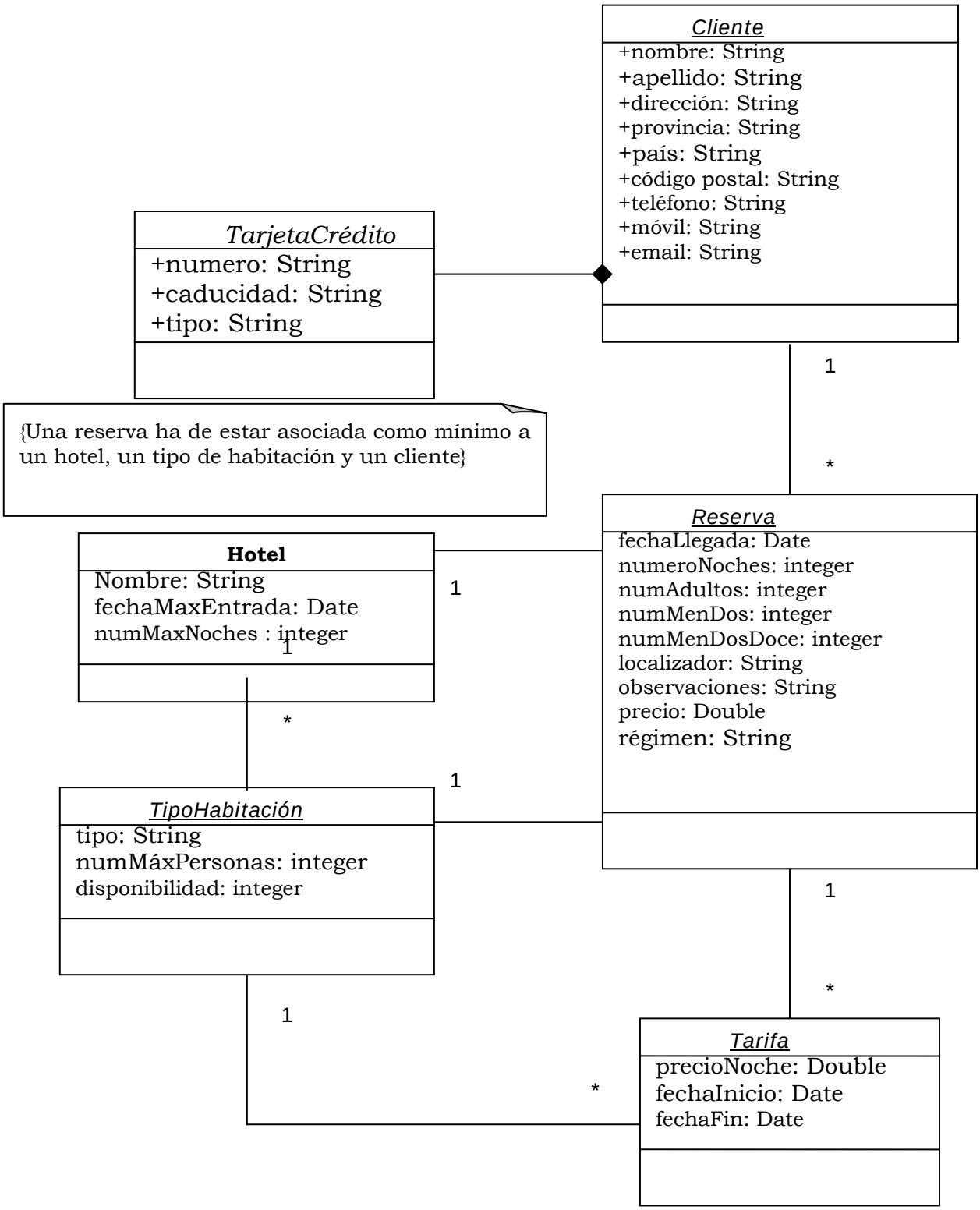
Tarjeta de Crédito,
número tarjeta de crédito es un atributo de **Tarjeta de Crédito**,
fecha caducidad tarjeta de crédito es un atributo de **Tarjeta de Crédito**,
tipo de tarjeta de crédito es un atributo de **Tarjeta de Crédito**,

Tarifa,
precio por noche es un atributo de **Tarifa,**
Fecha inicio es un atributo de **Tarifa**,
Fecha fin es un atributo de **Tarifa**,

Clases entidades:
Nos quedan las clases **Hotel**, **Reserva**, **Tipo de Habitación**, **Cliente**, **Tarjeta de crédito**, **Valores Reserva**, **Tarifa**.

Ahora se esta en condiciones de crear un diagrama de las clases de entidades, este diagrama se muestra en el siguiente apartado.

4.2.6 Diagrama de las clases de entidades



En este diagrama de clases de entidades se puede ver que hay una agregación entre cliente y tarjeta de crédito, puesto que una tarjeta siempre tiene que estar

asociada a un cliente y sin éste no tiene sentido, ya que una tarjeta pertenece a un sólo cliente.

También observamos que varias reservas están asociadas a un hotel y que diferentes tipos de habitación están asociados a un mismo hotel.

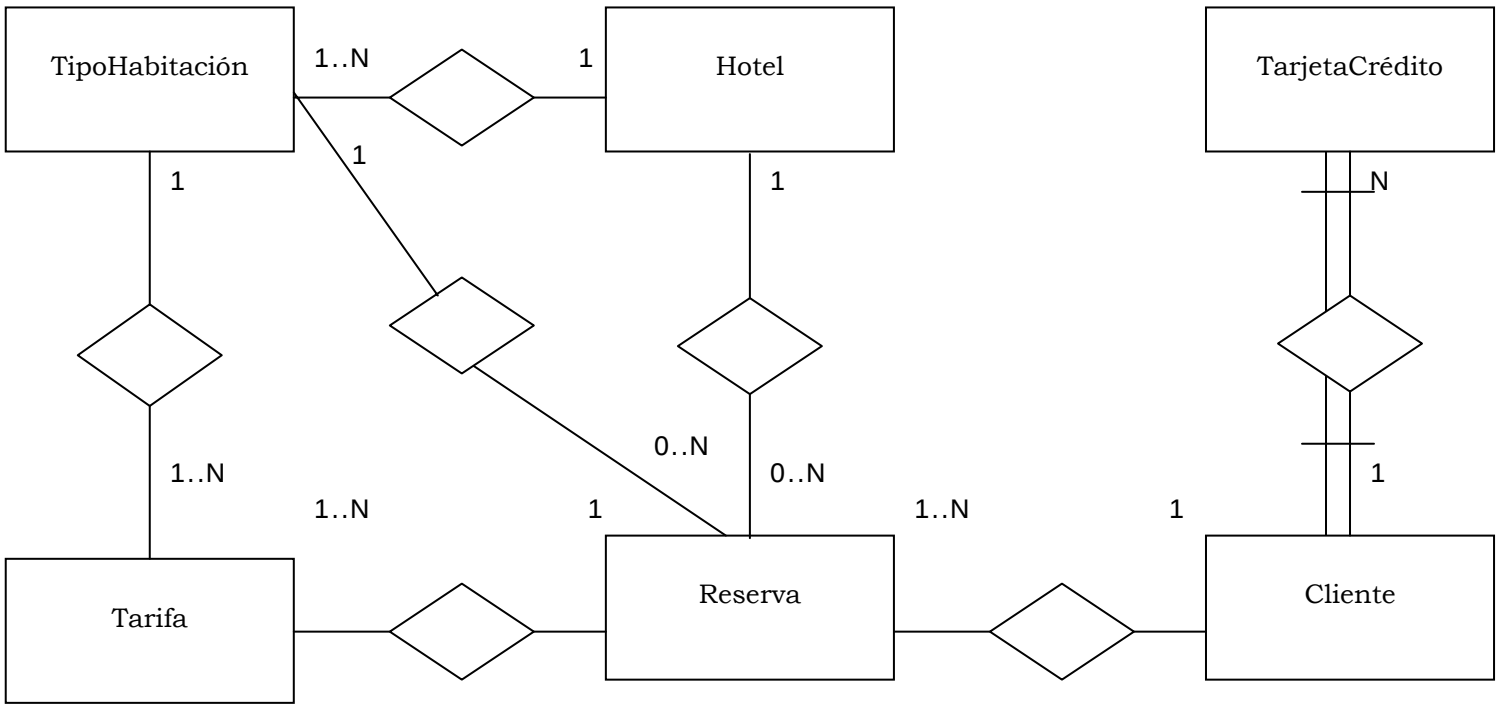
En cuanto a las tarifas tenemos que hay varias tarifas asociadas a un tipo de habitación y que puede haber N tarifas asociadas a una reserva, puesto que una reserva puede abarcar dos periodos de tarificación.

Por último tenemos reservas que están asociadas a cliente de tal manera que un cliente puede tener varias reservas y varias reservas pueden tener un tarjeta de crédito. Tal y como se ha mencionado anteriormente, tenemos que varias reservas están asociadas a un hotel y puede haber N reservas para N tipos de habitación y finalmente también puede haber N tarifas asociadas a una reserva.

Finalmente y para culminar la parte central del diseño de la aplicación, en el apartado 4.2.7 que viene a continuación, se presenta la creación de la base de datos teniendo en cuenta todos los pasos dados en toda la fase de diseño.

4.2.7 Diseño de la base de datos

4.2.7.1 Transforma del modelo estático al ER.



4.2.7.2 Transforma del modelo ER en Relacional.

Se crean las tablas correspondientes a las clases y relaciones siguientes:

Reserva: T_Reserva
Tarifa: T_Tarifa
Hotel: T_Hotel
TipoHabitación: T_TipHab
TarjetaCrédito: T_TarCre
Cliente: T_Cliente

4.2.7.3 Atributos, claves e índices.

Se presentan las tablas, con los atributos e índices así como las explicaciones necesarias.

Tabla T_Reserva

fechaLlegada: Date
numNoches: integer
numAdultos: integer
numMenDos: integer
numMenDosDoce: integer

localizador: String
observaciones: String
precio: Double
régimen: String
codCliente: String
codHotel: String
tipoHab: String

Clave primaria: localizador
Clave foránea hacia **T_Cliente** codCliente
Clave foránea hacia **T_Hotel** codHotel
Clave foránea hacia **T_TipHab** tipoHab, codHotel
Clave foránea hacia **T_Tarifa** fechaLlegada, regimen, tipoHab, codHotel

Como clave primaria alternativa tenemos, codigoHotel, codigoCliente, FechaLlegada, pero de esta manera sólo puede haber un reserva para un cliente en una fecha determinada.

Tabla T_Cliente

codCliente: String
nombre: String
apellido: String
dirección: String
provincia: String
país: String
código postal: String
teléfono: String
móvil: String
email: String

Clave primaria: codCliente

Tabla T_TarCre

número: String
caducidad: String
tipo: String
codCliente: String
Clave primaria: número

Clave foránea hacia **T_Cliente** codCliente

Tabla T_Hotel

codigoHotel: String
Nombre: String
fechaMaxEntrada: Date
numMaxNoches : integer

Clave primaria: codHotel

Tabla T_TipHab

tipo: String
numMáxPersonas: integer
disponibilidad: integer
codHotel: String
precio: real

Clave primaria: tipo, codHotel
Clave foránea hacia **T_Hotel** codHotel

Tabla T_Tarifa

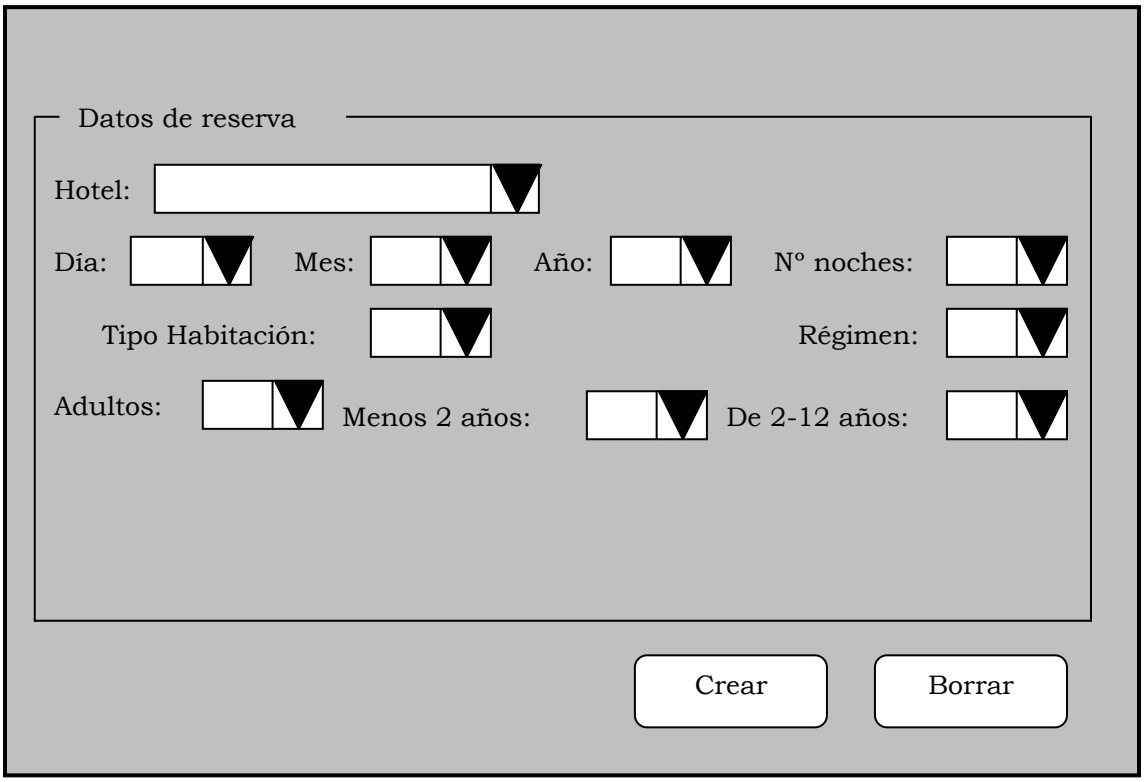
precioNoche: Double
fechaInicio: Date
fechaFin: Date
regimen: String
tarifa: Double
codHotel: String
tipoHab: String

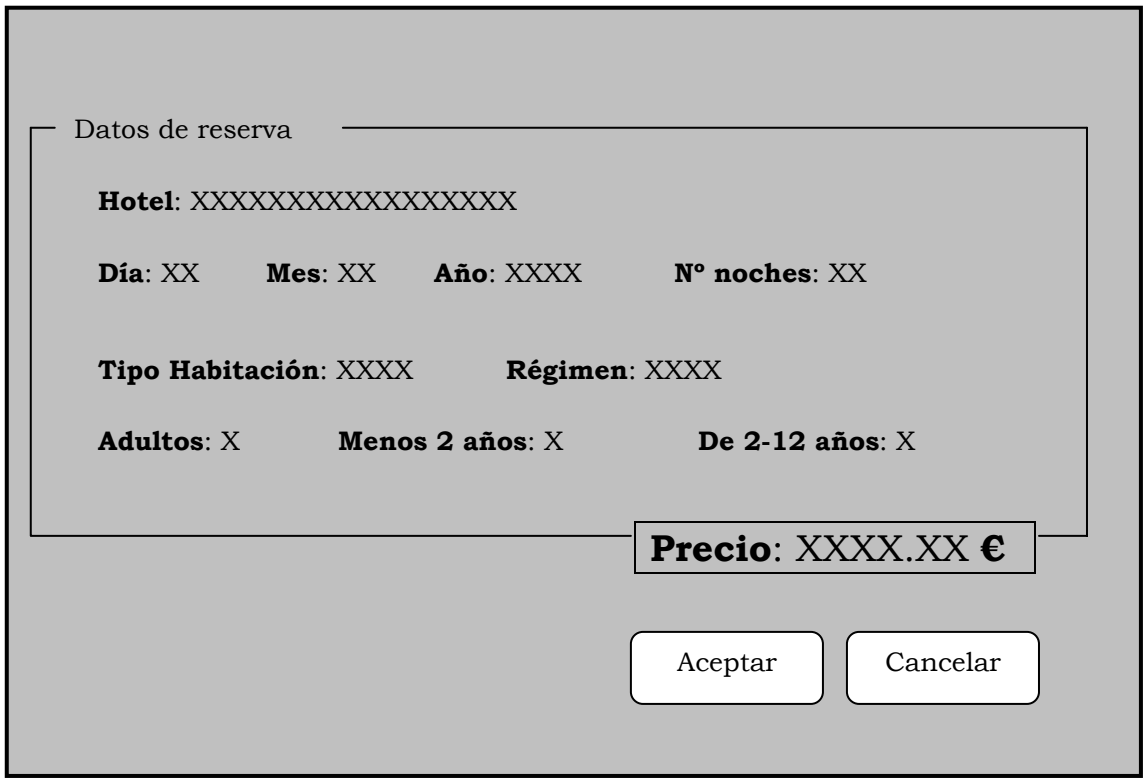
Clave primaria: fechaInicio, regimen, tipoHab, codHotel,
Clave foránea hacia **T_TipHab** tipoHab, codHotel

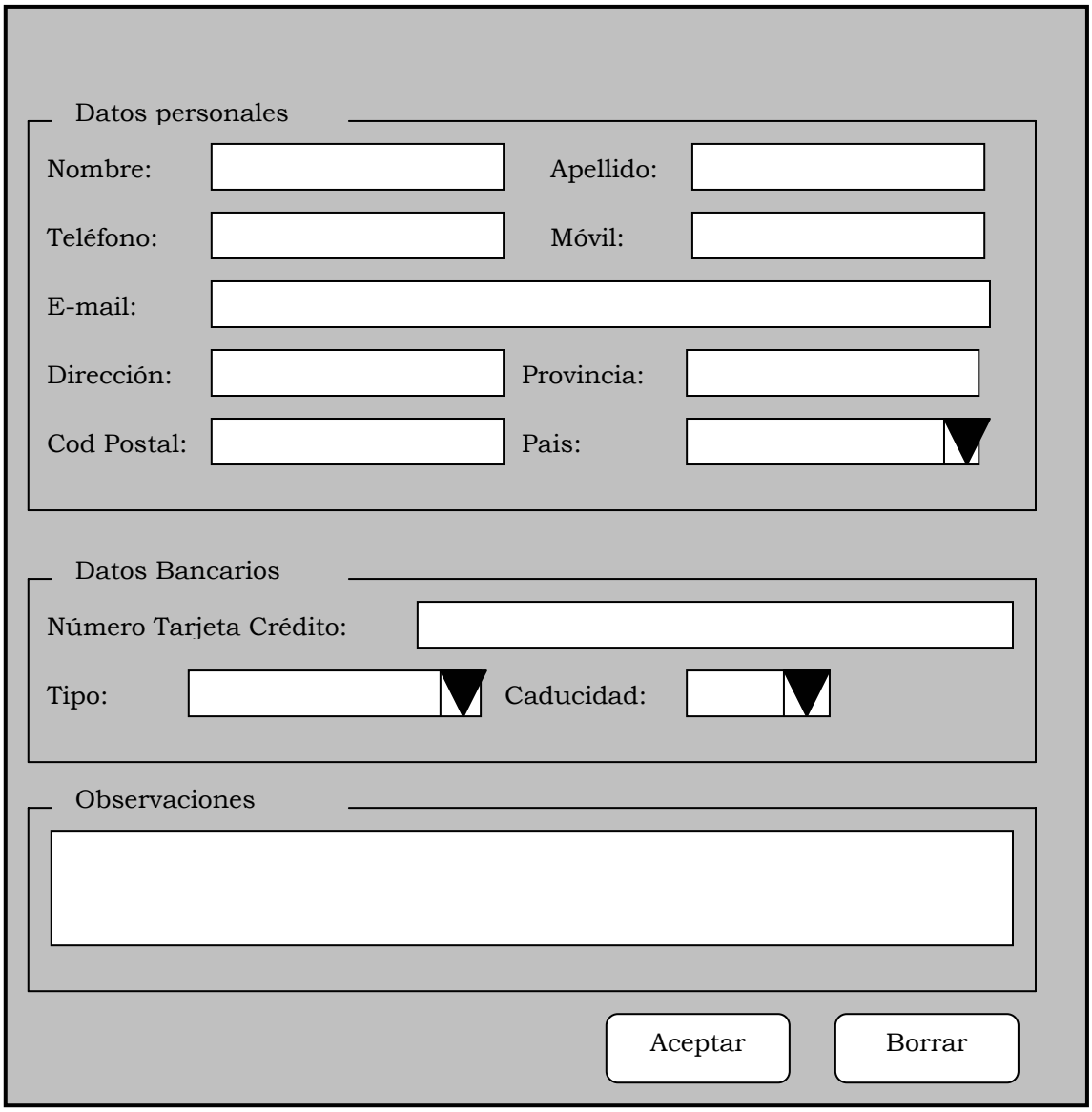
4.2.8 Diseño de la interficie de usuario

Tanto en este apartado como en el siguiente se presentan cuestiones relacionadas con la interficie de usuario. En este apartado en el ámbito de soporte al desarrollo y en el siguiente, se exponen vistas reales del producto final.

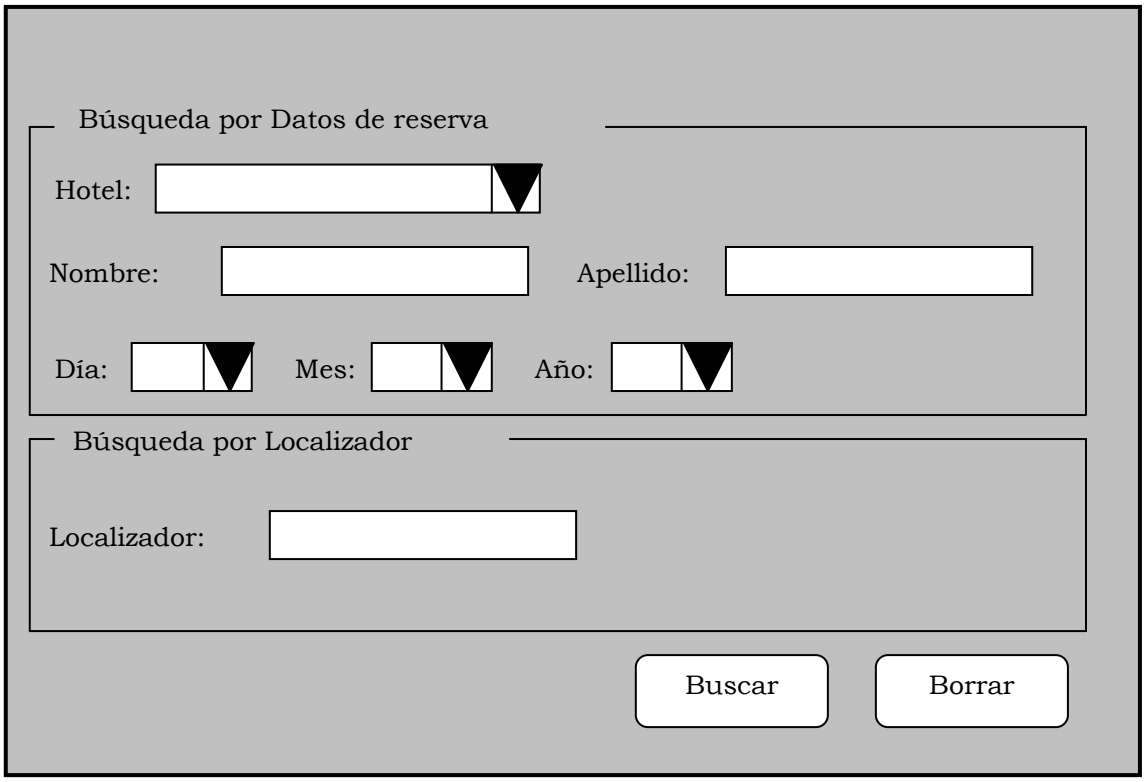
Caso de uso: 1.Creación reserva

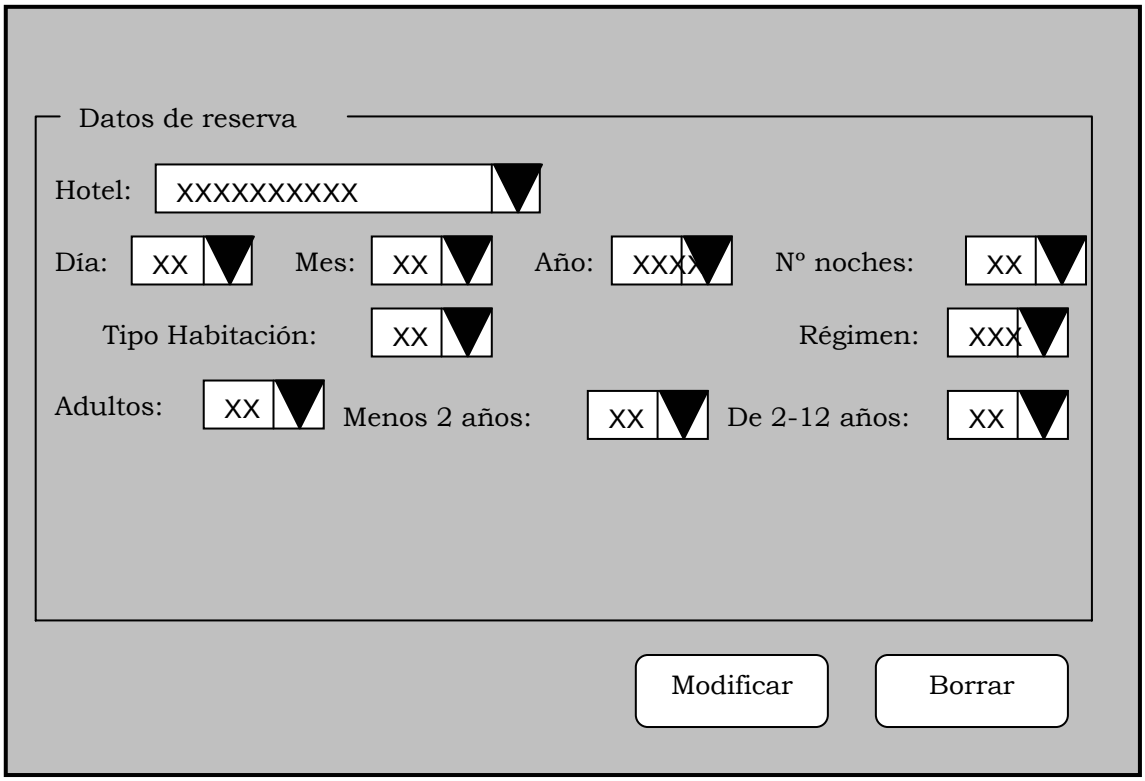


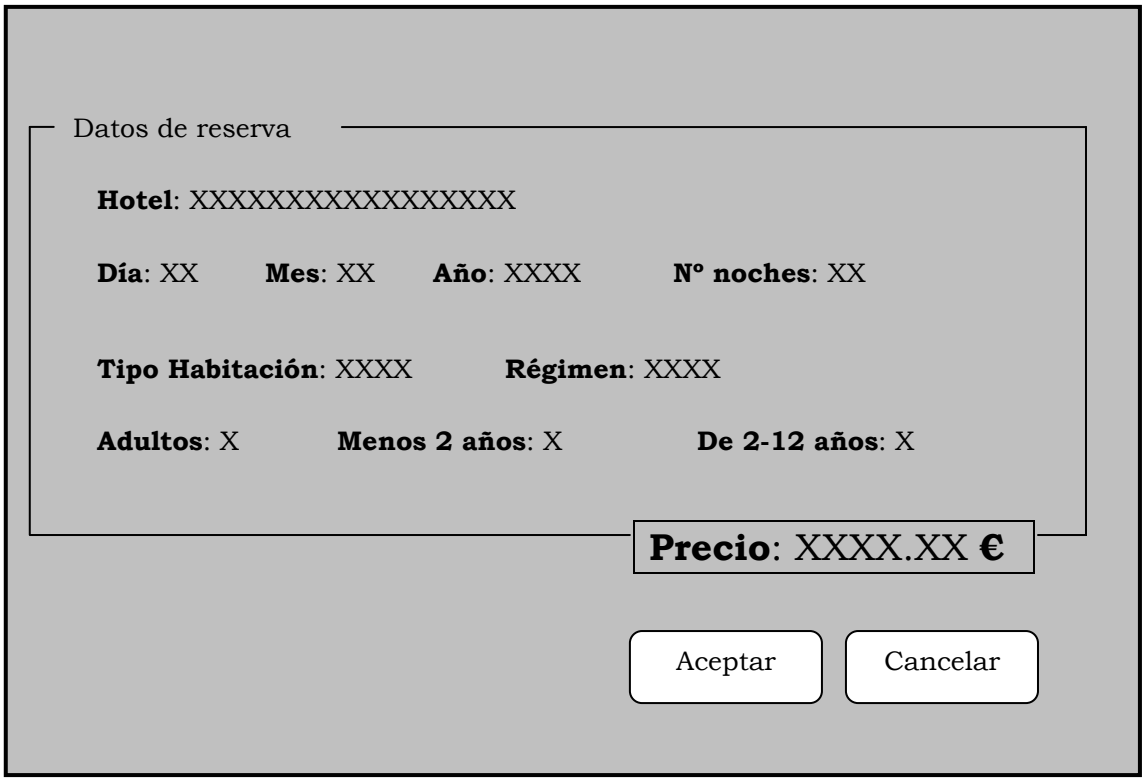



Datos personales
Nombre:
Apellido:
Teléfono:
Móvil:
E-mail:
Dirección:
Provincia:
Cod Postal:
Pais:
Datos Bancarios
Número Tarjeta Crédito:
Tipo:
Caducidad:
Observaciones
Aceptar
Borrar

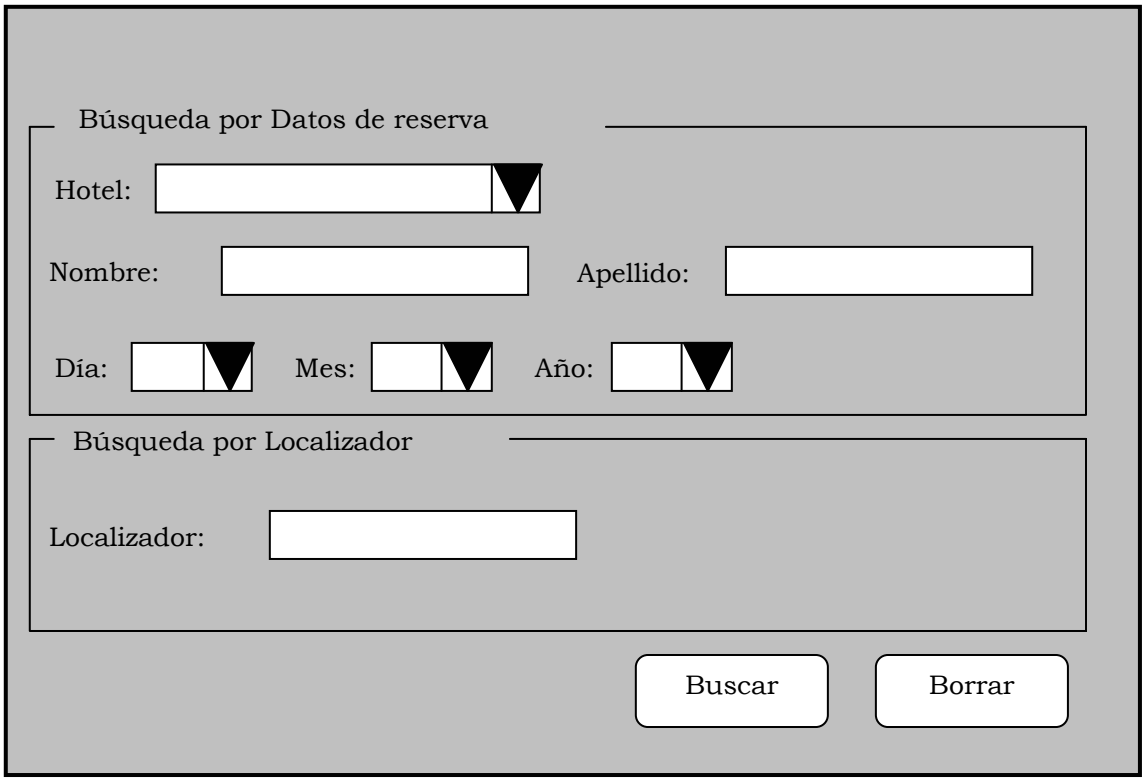
Caso de uso: 2. Modificación de reserva

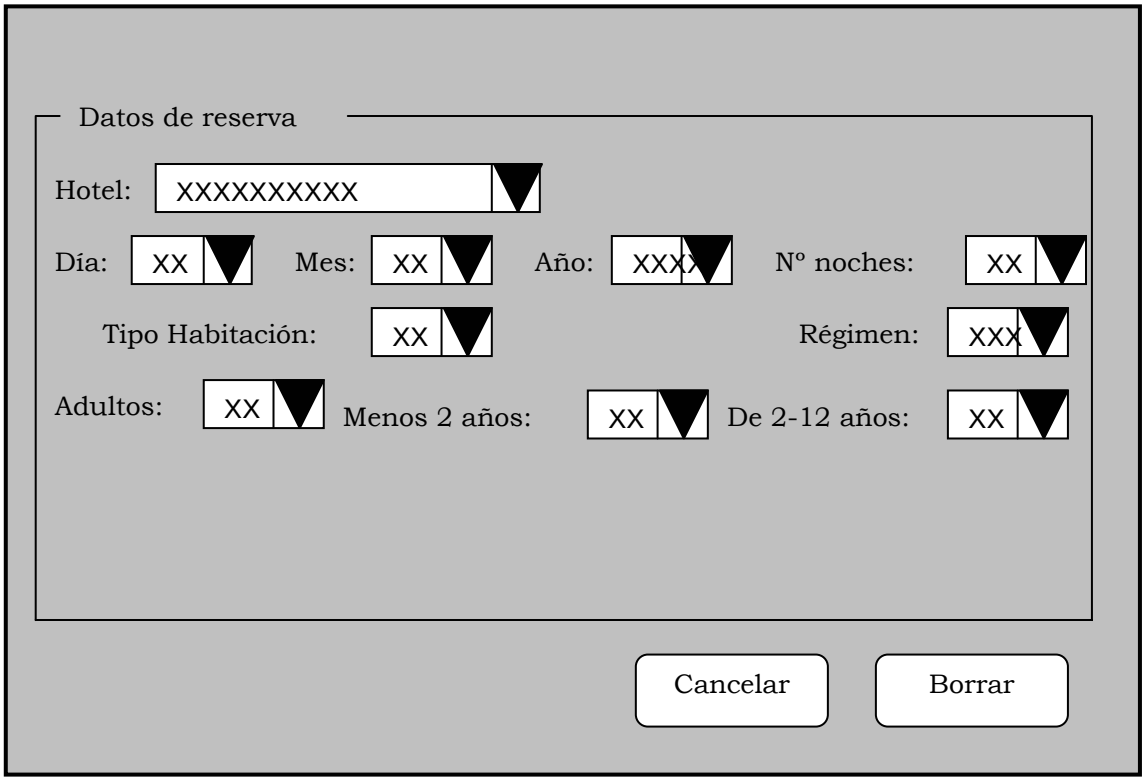




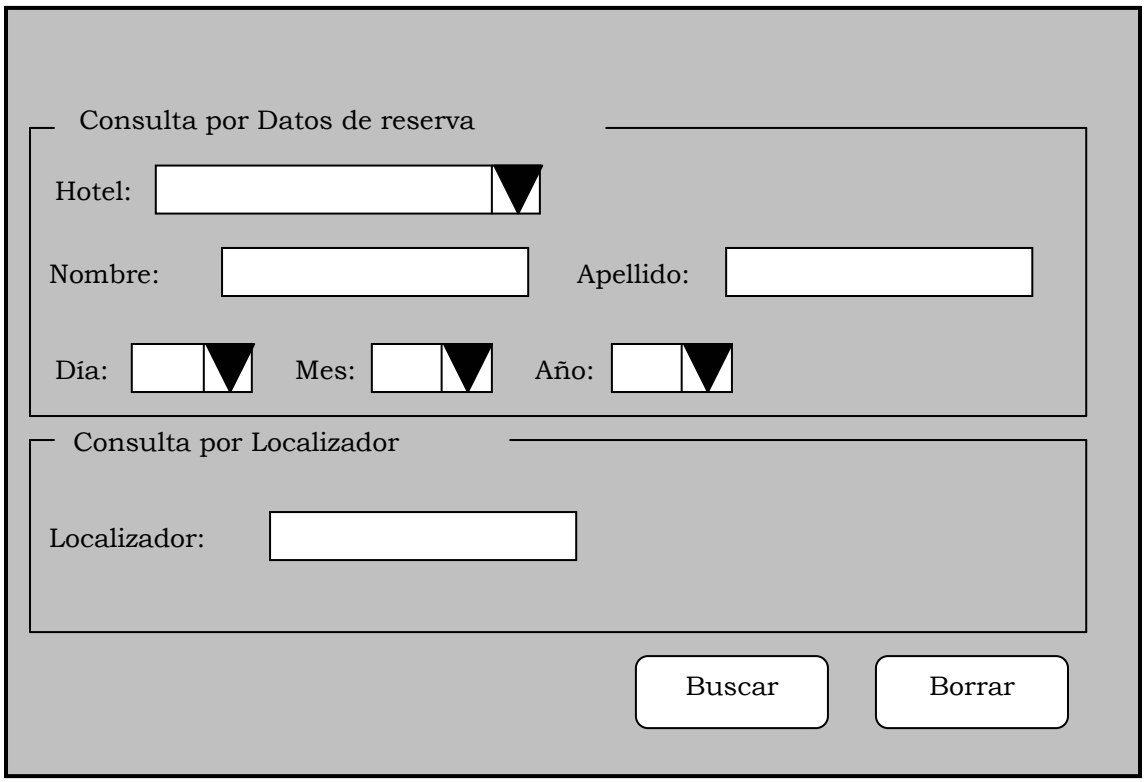

Datos de reserva
Hotel: XXXXXXXXXXXXXXXXX
Día: XX **Mes**: XX **Año**: XXXX **Nº noches**: XX
Tipo Habitación: XXXX **Régimen**: XXXX
Adultos: X **Menos 2 años**: X **De 2-12 años**: X
Precio: XXXX.XX €
Aceptar
Cancelar

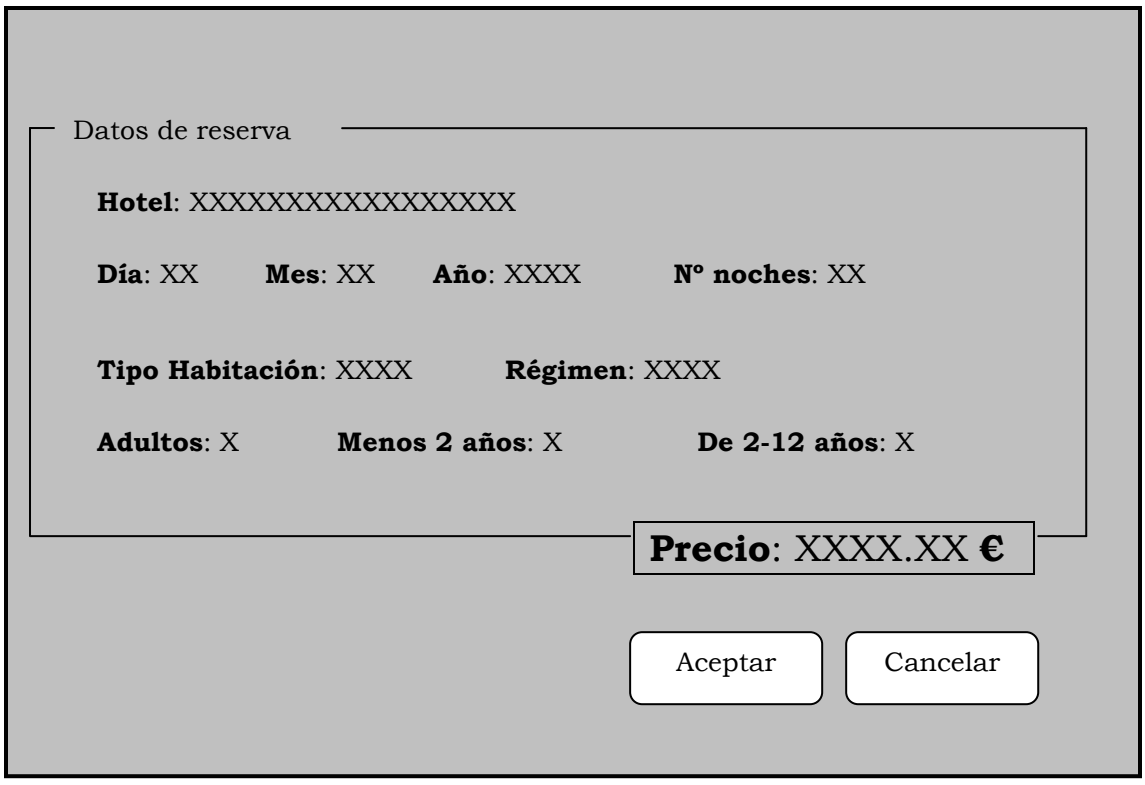
Caso de uso: 3. Cancelación de reserva





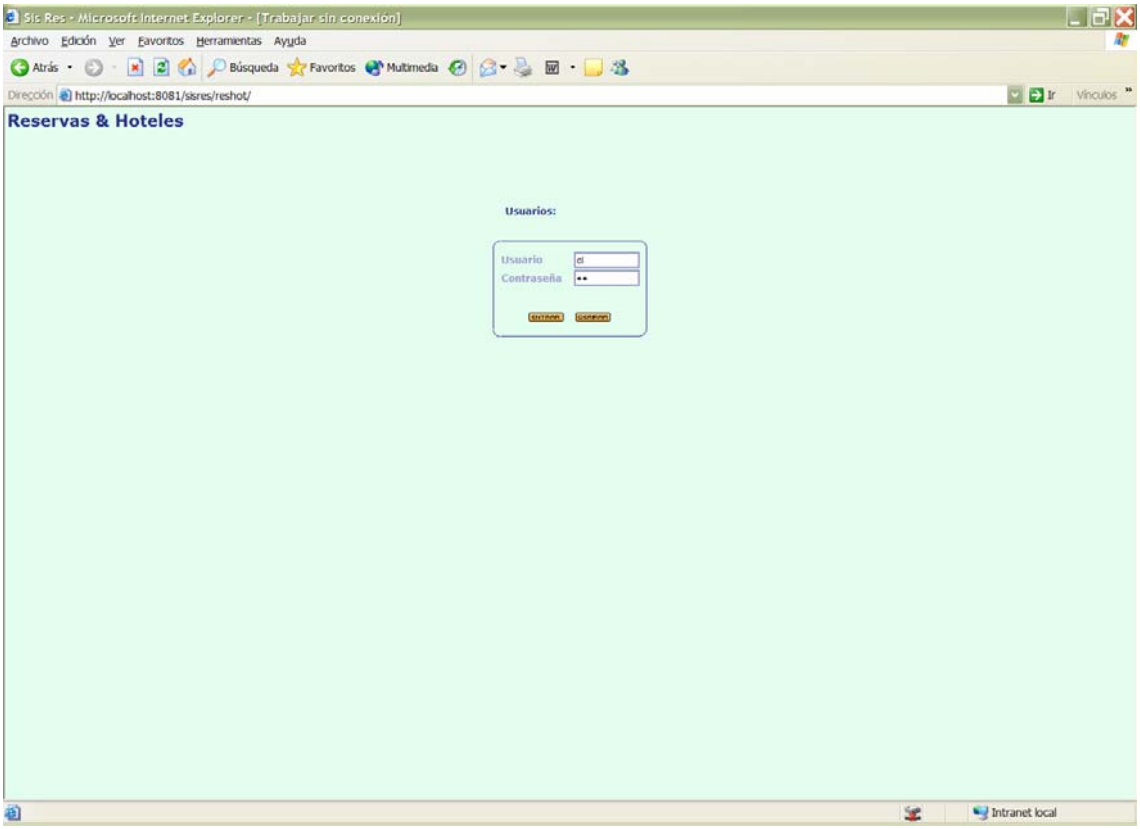
Caso de uso: 4. Consulta de reserva
Y
Caso de uso: 5. Consulta de reserva por localizador

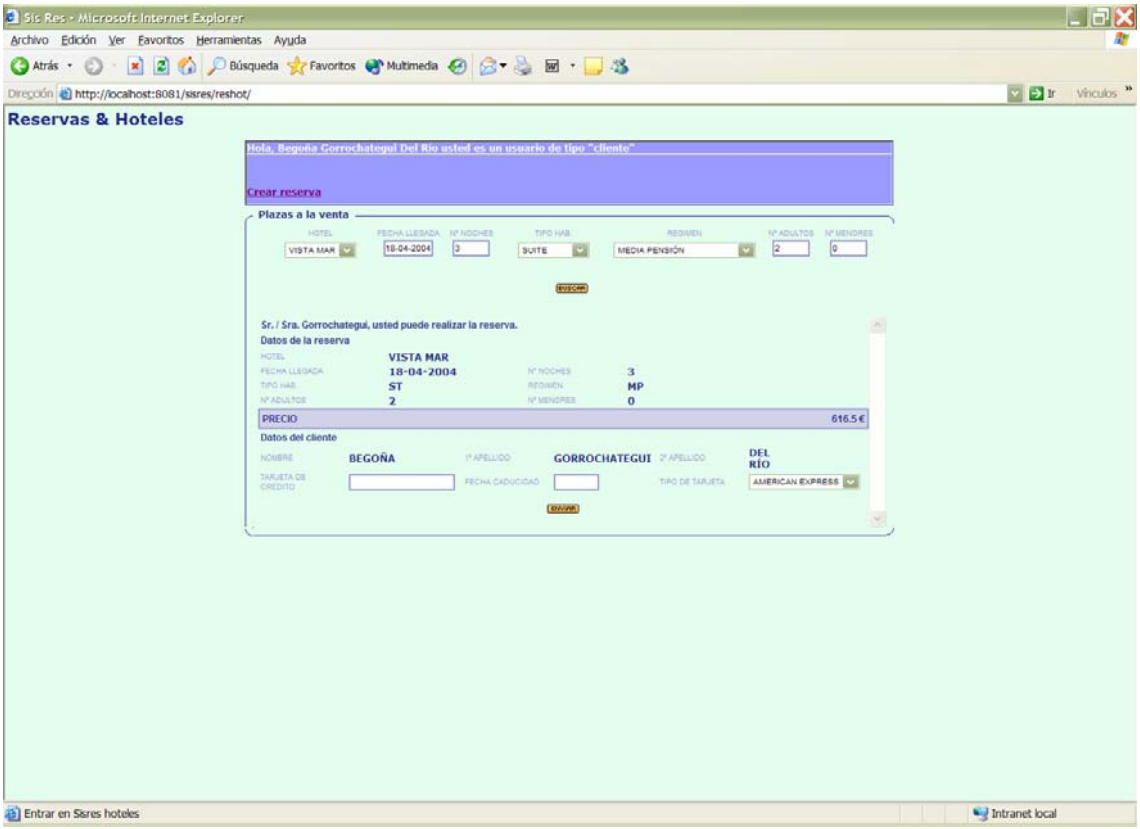




4.2.9. Vistas reales del proyecto

A continuación se presentan algunas vistas reales del proyecto.





4.3 Planificación del proyecto

En este apartado se presenta la planificación inicial planteada para la creación del proyecto.

4.3.1 Tareas fundamentales

Las tareas fundamentales del proyecto han sido:

1 Instalación

1.1 Instalación de librerías
1.2 Instalación entornos de desarrollo.

2 Documentación

2.1 Búsqueda de información
2.2 Estudio de sistemas de reservas de diferentes empresas del ramo.

3 Diseño, métodos, y desarrollo.

3.1 Diseño conceptual de la base de datos.
3.2 Creación de la base de datos en MySQL.
3.3 Diseño de la interficie grafica.
3.4 Creación de las interficies en HTML para los diferentes usuarios.
3.5 Creación de un método de conexión entre el código JAVA y la base de datos con JDBC.
3.6 Creación de las consultas, inserciones, updates, etc. de SQL necesarias para el funcionamiento del sistema.
3.7 Ejecución del código SQL o los procedimientos de MySQL desde código JAVA.
3.8 Creación de las estructuras de los servlets con J2EE.
3.9 Encapsulamiento de métodos de conexión y llamadas a base de datos en servlets.
3.10 Creación de la respuesta HTML dirigida a los clientes.

4 Juego de pruebas

4.1 Preparar un juego de pruebas para comprobar la robustez del sistema de reservas.

5 Redacción de la documentación

5.1 Redacción de la memoria.
5.2 Preparar la presentación en PowerPoint.

Planificación con hitos y temporización

La planificación temporal de tareas contempla todos los trabajos a realizar en el TFC distribuidos temporalmente durante el cuatrimestre.

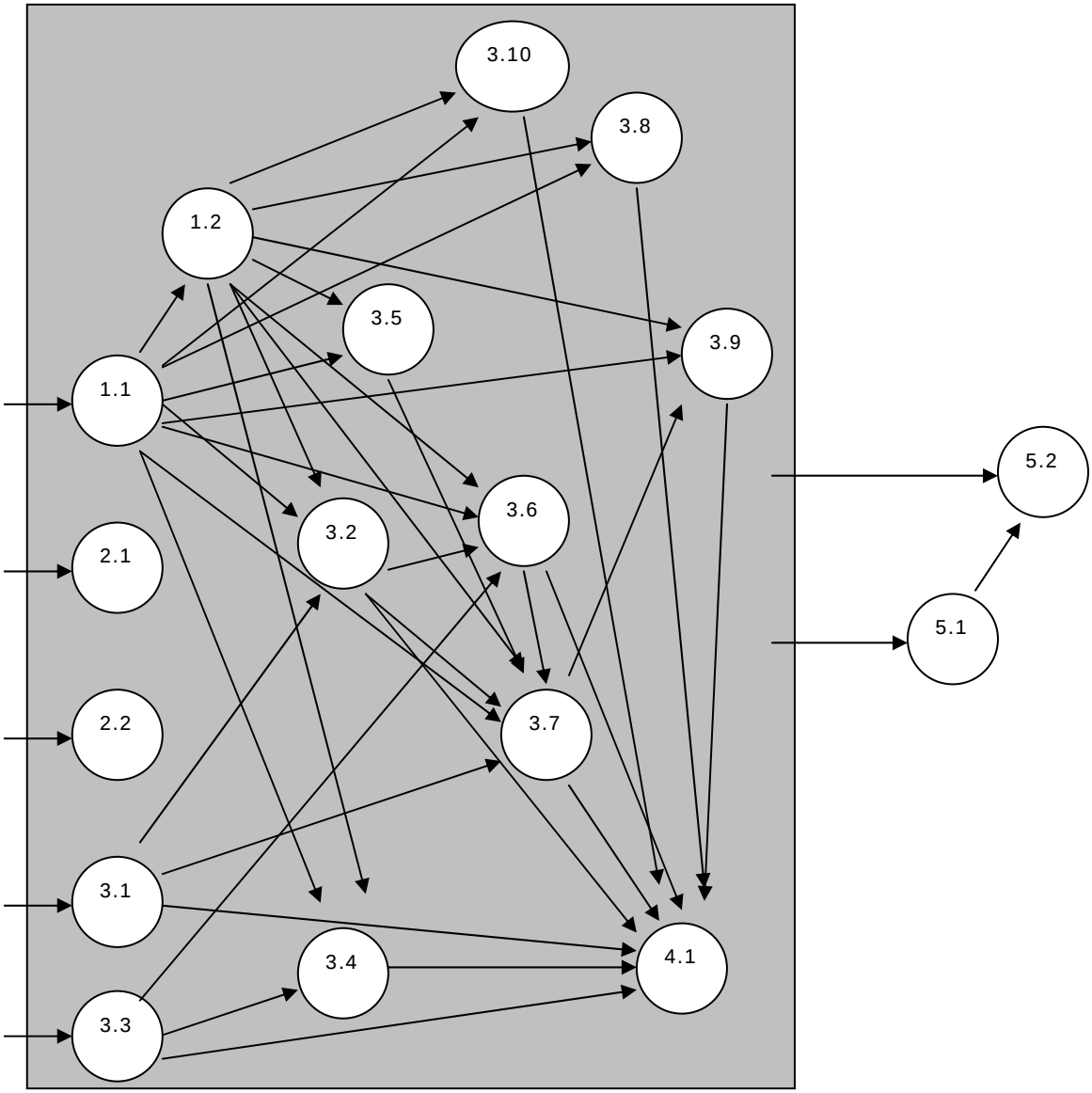
Semana	Fechas	Actividad	Evento
1	1 Marzo 7 Marzo	**Definir los objetos del proyecto.** **Realizar la planificación.** **Preparar un esbozo del esquema conceptual de la BD.**	PAC 1 Redacción de la PAC 1.
2	8 Marzo 14 Marzo	***1.1 Instalación de librerías*** **1.2 Instalación entornos de desarrollo.** **2.1 Búsqueda de información** **2.2 Estudio de sistemas de reservas de empresas hoteleras.** **5.1 Redacción de la memoria. Estudio del estado de la ciencia.**	8 Marzo Envío de la PAC1
3	15 Marzo 21 Marzo	**3.1 Diseño conceptual de la base de datos.** **3.3 Diseño de la interficie grafica.** **5.1 Redacción de la memoria. Diseño de la base de datos.** **5.1 Redacción de la memoria. Diseño de la interficie grafica.**	
4	22 Marzo 28 Marzo	**3.2 Creación de la base de datos en MySQL.** **5.1 Redacción de la memoria. Creación base de datos.**	PAC 2 Redacción de la PAC 2.
5	29 Marzo 4 Abril	**3.4 Creación de las interficies en HTML para los usuarios.** **5.1 Redacción de la memoria. Creación interficie grafica.**	29 Marzo Envío de la PAC2
6	5 Abril 11 Abril	**3.5 Creación de un método de conexión entre el código JAVA y la base de datos con JDBC.** **5.1 Redacción de la memoria. Conexión JAVA a BD con JDBC**	
7	12 Abril 18 Abril	**3.6 Creación de las consultas, inserciones, updates, etc. de SQL necesarias para el funcionamiento del sistema.** **5.1 Redacción de la memoria. Sentencias SQL.**	PAC 3 **Redacción de la PAC 3.**
8	19 Abril 25 Abril	**3.7 Ejecución del código SQL o los procedimientos de MySQL desde código JAVA.** **5.1 Redacción de la memoria. Llamadas a BD desde JAVA**	19 Abril Envío de la PAC 3.
9	26 Abril 2 Mayo	**3.8 Creación de las estructuras de los servlets con J2EE.** **5.1 Redacción de la memoria. Servlets**	
10	3 Mayo 9 Mayo	**3.9 Encapsulamiento de métodos de conexión y llamadas a base de datos en servlets.** **5.1 Redacción de la memoria. Servlets**	
11	10 Mayo 16 Mayo	**3.10 Creación de la respuesta HTML dirigida a los clientes.** **5.1 Redacción de la memoria. Integración de JAVA, JDBC y HTML.**	
12	17 Mayo 23 Mayo	**4.1 Preparar un juego de pruebas para comprobar la robustez del sistema de reservas.** **5.1 Redacción de la memoria. El juego de pruebas.**	
13	24 Mayo 30 Mayo	**5.1 Redacción de la memoria. Cuestiones varias, usabilidad, etc.**	
14	31 Mayo 6 Junio	**5.2 Redacción de la presentación, diapositivas explicativas del TFC.**	4 Junio Envío de la pre-memoria.
15	7 Junio 13 Junio	**Revisiones de la memoria, presentación, etc.** **Correcciones de ultima hora.**	
16	14 Junio 20 Junio	**Revisiones de la memoria, presentación, etc.** **Correcciones de ultima hora.**	18 Junio Envío Final. Memoria, producto y presentación.

En la planificación se contempla la carga que se habrá realizado para la fecha de cada una de las PACs por colores:

PAC 1 en verde.
PAC 2 en marrón.
PAC 3 en azul.
Pre-memoria en malva.
Entrega final en Rojo.

4.3.2 Grafo del algoritmo de tareas

El siguiente grafo muestra la sucesión de tareas y los precedentes que tiene cada una de las tareas para poder comenzar a realizarla.



4.3.3 Diagrama de Gantt del proyecto

Con el diagrama de Gantt pretendo ver la distribución de las tareas en el tiempo de una manera clara y la precedencia de tareas.

Id	**Tarea**	**Días**	**Precedentes**
1.1	*Instalación de librerías*	1	-
1.2	*Instalación entornos* de desarrollo.	1	-
2.1	Búsqueda de información	3	-
2.2	Estudio de sistemas de reservas de diferentes empresas del ramo.	2	-
3.1	Diseño conceptual de la base de datos.	4	-
3.2	Creación de la base de datos en MySQL.	3	1.1-1.2-3.1
3.3	Diseño de la interficie grafica.	2	-
3.4	Creación de las interficies en HTML para los diferentes usuarios.	3	1.1-1.2-3.3
3.5	Creación de un método de conexión entre el código JAVA y la base de datos con JDBC.	5	1.1-1.2
3.6	Creación de las consultas, inserciones, updates, etc. de SQL necesarias para el funcionamiento del sistema.	6	1.1-1.2-3.1-3.2
3.7	Ejecución del código SQL o los procedimientos de MySQL desde código JAVA.	6	1.1-1.2-3.1-3.2-3.5-3.6
3.8	Creación de las estructuras de los servlets con J2EE.	5	1.1-1.2
3.9	Encapsulamiento de métodos de conexión y llamadas a base de datos en servlets.	5	1.1-1.2-3.7
3.10	Creación de la respuesta HTML dirigida a los clientes.	5	1.1-1.2
4.1	Preparar un juego de pruebas para comprobar la robustez del sistema de reservas.	6	3
5.1	Redacción de la memoria.	30	1-2-3
5.2	Preparar la presentación en PowerPoint.	7	1-2-3-5.1

En el diagrama se puede ver la distribución de las tareas en el tiempo (unidad temporal es una semana).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Plan de trabajo																
	1.1															
	1.2															
	2.1															
	2.2															
		3.1														
			3.2													
		3.3														
				3.4												
					3.5											
						3.6										
							3.7									
								3.8								
									3.9							
										3.10						
											4.1					
								5.1								
															5.2	

5. BIBLIOGRAFÍA

Bobadilla, J. Y Sancho A.. Comunicaciones y bases de datos con JAVA a través de ejemplos. Madrid. RA – MA Editorial, 2003.

Daniel Diaz, Daniel. Diseño de aplicaciones J2EE, 2004, pp. 50-55 Revista profesional Sólo Programadores nº 110.

Mills R. et al. Perl, CGI y Javascript: Aprenda a desarrollar páginas Web interactivas. Madrid. Ed. Anaya Multimedia (Grupo Anaya S.A.), 2000.

MySQL AB, Tutorial básico de MySQL
[www.mysql-hispano.org/] y [http://www.programacion.com/], Marzo de 2004

Sun Microsystems. BluePrints Welcome to Core J2EE Patterns!
[http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html], Febrero de 2004

Universidad de Navarra. Escuela Superior de Ingenieros Industriales de San Sebastián. Aprenda Servlets de Java como si estuviera en segundo. San Sebastián, 1999.

Universidad de Navarra. Escuela Superior de Ingenieros Industriales de San Sebastián. Aprenda Java como si estuviera en primero. San Sebastián, 2000.

Zukowski, John. Programación: Java 2 J2SE 1.4. Madrid. Ed. Anaya Multimedia (Grupo Anaya S.A.), 2003.