5.5. *Aspectos de Seguridad*

En el diseño de la aplicación tenemos dos aspectos seguridad a tener en cuenta: el acceso a la misma por parte de los usuarios y el respaldo de la base de datos de la aplicación.

Acceso al sistema

Existen dos perfiles para la aplicación: recepcionista y administrador. Decidimos implantar el uso de contraseña a nivel de perfil en vez de usuario, para que les resulte más cómodo a éstos el cambio de usuario, sin necesidad de salir de la aplicación para volver a entrar.
Estas contraseñas se guardan en la base de datos utilizando un algoritmo de hashing.

El acceso a las páginas del servidor está restringido, salvo para la página *view.php*. De esta forma, solo necesitamos controlar los permisos para visitar una determinada página en un único sitio.

Respaldo de la base de datos

Para el respaldo de la base de datos utilizamos un script que se ejecutará de forma automática por el servidor.

Debido a que el tamaño de la base de datos no será muy grande, y no tenemos problemas de capacidad en el disco duro, desechamos la posibilidad de realizar copias de seguridad incrementales, optando únicamente por copias completas.

Teniendo en cuenta los horarios de recepción, se programa la realización de los siguientes respaldos:

- Copia diaria: se realizaran tres a lo largo del día, y se conservarán durante una semana.
- Copia mensual: se realizará una copia mensual, que se guardará indefinidamente.

Registro de actividad

La aplicación guarda en la base de datos un registro de actividad, registrando los inicios de sesión y las operaciones realizadas por cada usuario.

6. IMPLEMENTACIÓN

En este capítulo se detalla el proceso de implementación de diversas partes relevantes de la aplicación.

Posteriormente se explican los juegos de prueba realizados para comprobar su correcto funcionamiento.

Por último, se explican las tecnologías y lenguajes de programación utilizados para llevar a cabo la aplicación. También se enumera el software empleado para el desarrollo y funcionamiento de la misma.

6.1. Implementación de la página Calendario

La página de calendario es una de las más complejas de la aplicación. Desde ella se pueden realizar las siguientes operaciones:

- Desplegar las literas de una habitación común.
- Iniciar el proceso de reserva haciendo clic en dos fechas determinadas.
- Mostrar un globo de información al hacer clic en un evento.
- Iniciar el proceso de check-in/out haciendo clic en un evento de reserva o check-in respectivamente.
- Modificar las fechas o el alojamiento de una reserva.

El código necesario para la generación de la página de calendario se distribuye en los siguientes archivos:

- calendario.php: contiene una cabecera en php que genera todo el contenido variable de la pantalla(alojamientos, eventos). Posteriormente se encuentra el código html, y las funciones javascript específicas.

- calendario.func.php: contiene funciones php que generan y devuelven el código html de diferentes partes del calendario. Éstas son llamadas desde calendario.php.

- xajax.func.php: contiene funciones en ajax que se utilizan para variar partes de la página de una forma dinámica.

A continuación se muestran partes del código de cada una de estas páginas, comentando su funcionalidad.

calendario.php (Código php interpretado en el servidor)

```php
<?php
/* Protección que impide el acceso directo a este archivo.
   Este archivo es incluido a través de view.php
*/
if (basename(__FILE__) == basename($_SERVER['SCRIPT_FILENAME']))
     exit;

/* Se incluyen las clases de Dominio necesarias para la generación
del calendario.
*/
include($_SERVER['DOCUMENT_ROOT'] . '/Dominio/class_eventos.php');
include($_SERVER['DOCUMENT_ROOT'] . '/Dominio/class_calendario.php');
include($_SERVER['DOCUMENT_ROOT'] . '/Dominio/class_habitaciones.php');

/* Se incluye el archivo de funciones específicas para la
generación del calendario.
*/
include($_SERVER['DOCUMENT_ROOT'] . '/Presentacion/calendario.func.php');

/* Extracción de las variables recibidas mediante POST. Estas
variables permiten generar el calendario centrando el día actual en
una fecha determinada. En caso de no existir, el calendario se
genera tomando como fecha el día actual.
*/
if($_POST!=null){
       $d=$_POST['d'];
       $m=$_POST['m'];
       $y=$_POST['y'];
       $t=$_POST['t'];
}
else{
       $d=date("d");
       $m=date("m");
       $y=date("Y");
}
$center_date=$d."/".$m."/".$y;

/* Creación de los objetos necesarios para crear la página:
calendario, alojamiento y eventos. Éste último recibe como
parámetros el día inicio y fin del calendario a visualizar,
cargando todos los eventos del intervalo.
*/
$cal=new calendario($d, $m, $y);
$habit=new alojamiento();
$habit->get_all_aloj();
$evento=new eventos($cal->get_day_ini(), $cal->get_day_end());


/* Se incluye el archivo con definiciones de las funciones a
utilizar mediante xajax en el calendario. Posteriormente se generan
las funciones javascript que comunicaran con ellas.
*/
require_once ($_SERVER['DOCUMENT_ROOT'] . '/xajax.req.php');
$xajax->printJavascript('xajax/');
?>
```

calendario.php (Funciones javascript)

```
/* Las literas inicialmente no se escriben en la página, debido a
que su cantidad ralentizaría demasiado la carga.
En la fila siguiente a una habitación común, existe una fila oculta
con un contenedor div para las literas.
Cuando se hace clic para desplegar una habitación común, se llama a
la función showLiteras, que muestra dicho contenedor, y ésta a su
vez a la función ajax que carga el código HTML de las literas en el
contenedor.
*/
function showLiteras(idParent){
hideBox();

if(document.getElementById("containerLeft_"+idParent).style.display == ""){

      document.getElementById("containerLeft_"+idParent).style.display="no
ne";
      document.getElementById("containerMid_"+idParent).style.display="non
e";
      document.getElementById("alojLeft_"+idParent).style.backgroundImage=
"url(/img/arrow_d.gif)";
}
else{
      document.getElementById("alojLeft_"+idParent).style.backgroundImage=
"url(/img/ajax-loader.gif)";
      xajax_loadLit(idParent, d, m, y);
}
}

/* Esta función mueve el scroll horizontal del calendario para
centrar y que sea visible el día actual.
Además, añade un EventListener para ocultar el globo de información
cada vez que se mueva el scroll, ya que si no quedaría desplazado
respecto al evento al que apunta.
*/
function moveCalend(){
document.getElementById('calend_mid').scrollLeft=(24*<?php echo
calc_scrollLeft($t, $cal); ?>);
document.getElementById('calend_top').scrollLeft=document.getElementById('c
alend_mid').scrollLeft;
if(window.addEventListener){ // Mozilla, Netscape, Firefox
      document.getElementById('calend_mid').addEventListener("scroll",hide
Box,true);
}
else {
      document.getElementById('calend_mid').attachEvent("onscroll",hideBox
);
}
}

/* Esta función coloca el elemento div del globo de información y
lo carga mediante ajax con los datos del evento.
*/
function showBox(event, idEv, tipo){
if (!event) var event = window.event;
if(tipo>=0){ //si no es "modificar/cancelar reserva"
      hideBox();
      box=document.getElementById("BoxDiv");
      // para NS e.pageX/Y, hay que pasar evento...
      box.style.top=event.clientY + 15;
      box.style.left=event.clientX - 35;
}
xajax_loadBox(idEv, tipo);
}
```

calendario.php (Cuerpo html de la página)

```
<body>
<div id="base">
/* Se incluye el menú, común a todas las páginas. */
<?php include('menu.php'); ?>

<div id="principal" style="width:100%">
/* Estos formularios están ocultos, y envían mediante POST las
variables necesarias a la página de destino.
El primero se utiliza para crear una reserva, y el segundo para
hacer un check-in/out.
*/
<form action="/view.php?page=reserva" method="POST" name="FormRes">
      <input type="hidden" name="modo" value="crear"/>
      <input type="hidden" name="id_habit" value=""/>
      <input type="hidden" name="fecha_start_calend" value="<?php echo
date("d/"."m/"."Y",$cal->get_day_ini()); ?>"/>
      <input type="hidden" name="fecha_ini" value=""/>
      <input type="hidden" name="fecha_fin" value=""/>
</form>
<form action="" method="POST" name="FormCheck">
      <input type="hidden" name="modo" value="crear"/>
      <input type="hidden" name="id_ev" value=""/>
</form>

/* Se escriben las fechas de inicio y fin del calendario mostrado.
*/
<div style="float:left">
      <?php echo date("d/"."m/"."Y",$cal->get_day_ini()); ?>
</div>
<div style="float:right; padding-right:10px">
      <?php echo date("d/"."m/"."Y",$cal->get_day_end()); ?>
</div>

/* Comienza el código HTML del calendario. Hay cuatro
subdivisiones: izquierda, derecha(superior y central), inferior.
Para las tres primeras se llaman a funciones php específicas que
generan y escriben el código mediante los objetos de Dominio
pertinentes.
*/
<div id="calend">
  <div id="calend_left"><?php genera_left($habit); ?></div>

  <div id="calend_right">
    <div id="calend_top"><?php genera_top($cal); ?></div>

    <div id="calend_mid"><?php genera_mid($habit, $evento, $cal); ?></div>
  </div>
```

calendario.php (Cuerpo html de la página)

```
/* La subdivisión inferior contiene los enlaces a "Atrás" y
Siguiente", así como la opción de "Ir a..."
*/

<div id="calend_bottom">
 <table class="t_general">
 <tr>
   <td  bgcolor="#ecf8cb" width="100px"> </td>
   <td  bgcolor="#ecf8cb"  width="100%">
     <div>
     <ul style="margin:0px">
     <li style="display:inline">
       <form action="view.php?page=calendario" method="POST"
       name="FormBack">
         <?php genera_bottom_hiddens($cal, 1)?>
         <a href="javascript:document.FormBack.submit()">Atras</a>
       </form>
     </li>
     <li style="display:inline">
       <input id="go_to" name="go_to" type="text" value="<?php echo
       $center_date; ?>" size="9"/>
       <input type="button" value="Ir a..." onClick="ir_a()">
     </li>
     <li style="display:inline;">
       <form action="/view.php?page=calendario" method="POST"
       name="FormNext">
      <?php genera_bottom_hiddens($cal, 2)?>
      <a href="javascript:document.FormNext.submit()">Siguiente</a>
       </form>
     </li>
     </ul>
     </div>
   </td>
 </tr>
 </table>
</div>
</div>

</div>
</div>

</div>

/* Este elemento es el globo de información, que inicialmente está
oculto. Cuando se hace clic en algún evento, éste se actualiza con
la información de dicho evento y se muestra en la parte de la
pantalla correspondiente.
*/
<div id="BoxDiv" style="display:none">
<img style="position: absolute; left: 33px; top: -16px;"
src="/img/cal_box_border_arrow.gif" onClick="hideBox()">
</div>

</body>
```

xajax.func.php (Funciones php llamadas mediante ajax)

```
/* Esta función se utiliza para cargar y mostrar las literas en
pantalla cuando se despliega una habitación común.
Recibe como parámetros el id de la habitación común a desplegar, y
el día central del calendario que se está mostrando. */

function loadLit($idParent, $d, $m, $y){
/* El objeto objResponse contiene el código XML con los datos que
se envían al navegador, donde es tratado por funciones javascript.
$objResponse = new xajaxResponse('ISO-8859-1');

/* Se cargan las literas pertenecientes a la habitación común y los
eventos de éstas */
$cal=new calendario($d, $m, $y);
$habit=new alojamiento();
$habit->get_all_aloj($idParent);

$evento=new eventos($cal->get_day_ini(), $cal->get_day_end());

/* Comienza la generación del código HTML que será devuelto. */
$htmlLeft='<table  class="t_general">';
$htmlMid='<table  class="t_general"
id="tableContainerMid_'.$idParent.'">';
if($habit->get_count())
do{
      $htmlLeft.='<tr class="t_row"><td  class="t_col_habit"
      style="background-color:'.$habit->get_color().'"
      title="'.$habit->get_tipo().'"><div align="right">'.$habit-
      >get_nombre().'</div></td></tr>';
      $htmlMid.=genera_row($evento, $cal, $habit->get_id(), true);
}while($habit->movenext());
$htmlLeft.='</table>';
$htmlMid.='</table>';

/* Se añade el código HTML generado al objeto Response, indicándole
el id del contenedor donde deberá ser úbicado, y el tipo de
información. */
$objResponse-
>addAssign("containerLeftTD_".$idParent,"innerHTML",$htmlLeft);
$objResponse-
>addAssign("containerMidTD_".$idParent,"innerHTML",$htmlMid);

/* Se añade también código javascript para mostrar los contenedores
div. Éste se ejecutará en el navegador cuando el objeto sea
procesado. */
$objResponse-
>addScript('document.getElementById("contLeft_'.$idParent.'").style
.display="";');
$objResponse-
>addScript('document.getElementById("contrMid_'.$idParent.'").style
.display="";');

return $objResponse;
}
```

calendario.func.php (Funciones php utilizadas por calendario.php)

```
/* Esta función recibe como parámetro un objeto calendario y genera
la parte superior de la tabla de calendario, que contiene los
nombres de los meses y días representados. */

function genera_top($cal){
$str_d_num='';

/* Se escribe el comienzo de la tabla. En la primera fila irán los
nombres de los meses mostrados, uno en cada columna. */
print '<table class="t_general"><tr class="t_row_header">';

/* Se obtienen el mes y año iniciales. */
$m_ini=date("m",$cal->get_day_ini());
$y=date("Y",$cal->get_day_ini());

$d_mes=array("Enero","Febrero","Marzo","Abril","Mayo","Junio","Juli
o","Agosto","Septiembre","Octubre","Noviembre","Diciembre");

/* Se recorren los meses que se mostrarán. */
for($i=0;$i<$cal->get_months_shown();$i++){
      $mes=(($m_ini+$i-1)%12)+1;

      /* Se obtiene el número de días del mes. */
      $m_days=cal_days_in_month(CAL_GREGORIAN, $mes, $y);

      /* Se escribe la columna del mes. */
      print '<td colspan="'.$m_days.'">'.$d_mes[$mes-1].'</td>';

      /* Se generan las columnas con los días del mes, y se guardan
      para escribirlas al final. */
      $str_d_num=$str_d_num.genera_dias_num($m_days);

      if($m_ini+$i>12) $y++;
}

/* Comienza la siguiente fila, donde irán los días de la semana.
*/
print '</tr><tr class="t_row_header">';
genera_dias_sem($cal->get_day_ini(), $cal->get_sum_till_act(),
$cal->get_sum_days());

/* Comienza la última fila, donde irán los días del mes, en formato
numérico.   */
print '</tr><tr class="t_row_header">'.$str_d_num.'</tr></table>';
}
```

6.2. *Implementación de la clase Alojamiento*

La clase alojamiento proporciona toda la información referente a los alojamientos del hotel y a sus diferentes tipos.

Esta clase utiliza a su vez la clase eventos en la función *get_free_aloj*, encargada de encontrar los alojamientos disponibles para un intervalo de fechas. De esta forma la clase alojamiento solo maneja la información de las habitaciones, y es la clase eventos la que busca reservas o check-in existentes entre dos fechas, devolviendo el id de los alojamientos.
A continuación se muestran algunas partes del código de la clase.

class_alojamiento.php

```
<?php
/* Se incluye la clases de Datos y Dominio necesarias. */
require ($_SERVER['DOCUMENT_ROOT'] . '/Datos/Dhabitaciones.php');
require ($_SERVER['DOCUMENT_ROOT'].'/Dominio/class_eventos.php');

class alojamiento{
/* Variables privadas para almacenar alojamientos y tipos. */
private $habit;
private $taloj;
/* Constantes con los tipos de alojamiento fijos del sistema. */
public static $TIPO_COMUN=1;
public static $TIPO_CAMA=5;
/* Constantes informativas del resultado de una operación. */
public static $ID=6;
public static $OK_INS=1;
public static $OK_ELIM=2;
public static $ERR=-1;

/* Esta función recibe como parámetros el id del alojamiento, en el
caso de que queramos obtener las literas del mismo. Como resultado
carga en la variable interna $habit el total de alojamientos del
sistema o el total de literas de un determinado alojamiento. */

function get_all_aloj($id_parent=0){
$datos = new Dhabitaciones();

if($id_parent)
      $rs = $datos->get_literas($id_parent);
else
      $rs = $datos->get_aloj_all();

$this->habit=null;
while($rs->next()) {
      $this->habit[$rs->getInt('Id_aloj')] = array("nombre"=>$rs-
>getString('nombre'), "id_tipo"=>$rs->getInt('Id_tipo'),
"tipo"=>$rs->getString('descripcion'), "color"=>$rs-
>getString('color'), "id_parent"=>$rs->getInt('Id_parent'),
"num_matrim"=>$rs->getInt('num_matrim'), "num_indiv"=>$rs-
>getInt('num_indiv'), "orden"=>$rs->getInt('orden') );
}
return $rs->getRecordCount();
}

}
```

class_alojamiento.php

```
/* Esta función recibe como parámetros el id del alojamiento, en el
caso de que queramos obtener las literas disponibles de una
habitación común, y un intervalo de fechas. Como resultado carga en
la variable interna $habit los alojamientos disponibles. */

function get_free_aloj($id_parent=0, $fec_ini, $fec_fin){
/* Se crea el objeto de la capa de datos encargado de acceder a los
datos de alojamientos. */
$datos = new Dhabitaciones();


if($id_parent)
      $rs = $datos->get_literas($id_parent);
else
      $rs = $datos->get_aloj_all();

/* Se inicializa la variable $habit y se cargan en ella todos los
alojamientos. */
$this->habit=null;
$count=0;
while($rs->next()) {
      $this->habit[$rs->getInt('Id_aloj')] = array("nombre"=>$rs-
      >getString('nombre'), "id_tipo"=>$rs->getInt('Id_tipo'),
      "tipo"=>$rs->getString('descripcion'), "color"=>$rs-
      >getString('color'), "id_parent"=>$rs->getInt('Id_parent'),
      "num_matrim"=>$rs->getInt('num_matrim'), "num_indiv"=>$rs-
      >getInt('num_indiv'));
      $count++;
}

/* Se llama a la función get_ocup_id_aloj de la clase eventos, que
nos devuelve un vector de ids de alojamientos ocupados para las
fechas dadas. Posteriormente se eliminan de $habit los alojamientos
ocupados.*/

$ev = new eventos();
$ocup_res=$ev->get_ocup_id_aloj($fec_ini, $fec_fin);
if(sizeof($ocup_res)>0)
      foreach($ocup_res as $id){
          unset($this->habit[$id]);
      }

/* La función devuelve el número de alojamientos libres */
return $count-sizeof($ocup_res);
}
```

class_alojamiento.php

```
/* Esta función inserta un alojamiento nuevo en el sistema. Recibe
como parámetros un array con la información del alojamiento. */

function insertar_aloj($data){
$dtemp = new Dhabitaciones();

/* Si no se ha especificado el tipo de alojamiento se devuelve
error. */
if($data['aloj_tipo']==0)
      $result=alojamiento::$ERR_ALOJ_TIPO;
else{
      if (strlen($data['aloj_idparent'])>0)
            $data['aloj_idparent']=1;
      else
            $data['aloj_idparent']=0;
/* Se inserta el alojamiento en la base de datos y se obtiene el id
asignado. */
      $id = $dtemp->insert_aloj($data['aloj_tipo'],
      $data['aloj_nombre'], $data['aloj_matrim'],
      $data['aloj_indiv'], $data['aloj_idparent'],
      $data['aloj_orden']);
 if($id>0){
  if ($data['aloj_idparent']>0){
/* Si es un aloj. común se asigna su propio id al id del
alojamiento padre. */
      $data['aloj_idparent']=$id;
      $data['aloj_tipo']=alojamiento::$TIPO_COMUN;
      $rs = $dtemp->update_aloj($id, $data['aloj_tipo'],
      $data['aloj_nombre'], 0, $data['aloj_indiv'],
      $data['aloj_idparent'], $data['aloj_orden']);
  }
  /* se insertan las literas correspondientes.
  if($data['aloj_tipo']==alojamiento::$TIPO_COMUN){
      $count=0; //$this->get_all_aloj($data['id_aloj']);
      //se insertan la diferencia de camas
      while($count<$data['aloj_indiv']){
        $count++;
        $nombre=$data['aloj_nombre']."-".sprintf("%02d",$count);
        $res=$dtemp-
>insert_aloj(alojamiento::$TIPO_CAMA,$nombre,0,1,$id,0);
      }
  }
  $result=alojamiento::$OK_INS;
 }
 else{
      $result=alojamiento::$ERR_INS_ALOJ;
 }
}
/* Se devuelve una de las constantes informativas según el
resultado del proceso de insertado. */
return $result;
}
```

6.3. *Implementación del sistema de Avisos*

El sistema de avisos de la aplicación tiene como objetivo informar al usuario de una forma clara sobre el resultado de una operación, bien si ésta se ha efectuado con éxito o si se ha producido algún error.

Capa de dominio

Cada clase tiene definidas una constante que la identifica entre el resto de clases, y un conjunto de constantes que identifican el resultado de cada operación. Éstas son devueltas por las funciones de la clase . A continuación podemos ver un ejemplo:

class_checkinres.php

```
<?php

require ($_SERVER['DOCUMENT_ROOT'] . '/Datos/Dcheckres.php');
require ($_SERVER['DOCUMENT_ROOT'] . '/Dominio/class_factura.php');

class checkinres{

//identificador de la clase
public static $ID=1;

//identificadores de los diferentes tipos de avisos
public static $OK=1;
public static $ERR_RES=-1;
public static $ERR_CHECK=-2;
public static $ERR_FRA=-4;
public static $ERR=-3;

//inserta los datos en tabla checkin y ocupantes
function make_checkinres($checkin){
      $datos = new Dcheckres();
      $rs=$datos->get_check_by_res($checkin['id_res']);

      if($rs->getRecordCount()==0){
            if($checkin['id_res']>0){
                  $rs = $datos->insert_checkres(...);
                  //check in realizado correctamente
                  $result=checkinres::$OK;
            }
            else  //error: reserve no seleccionada
                  $result=checkinres::$ERR_RES;
      }
      else{ //error: check in ya realizado
            $result=checkinres::$ERR_CHECK;
      }
      return $result;
}

}
```

Capa de Presentación

Existe una página llamada *message_box.php*, la cuál contiene todos los mensajes de aviso, agrupados por clases, y el código html de la ventana que donde muestra el aviso. Además incluye todas las clases que generan avisos para tener acceso a sus constantes.

Esta página recibe mediante el método GET dos variables: el identificador de la clase a la que pertenece la función llamada y el identificador del resultado devuelto por dicha función.
Seguidamente se muestra parte del código de esta página.

message_box.php

```
include($_SERVER['DOCUMENT_ROOT']. '/Dominio/class_factura.php');
include($_SERVER['DOCUMENT_ROOT'].'/Dominio/class_checkinres.php');
include($_SERVER['DOCUMENT_ROOT']. '/Dominio/class_cliente.php');
include($_SERVER['DOCUMENT_ROOT']. '/Dominio/class_eventos.php');
include($_SERVER['DOCUMENT_ROOT']. '/Dominio/class_listado.php');
include($_SERVER['DOCUMENT_ROOT']. '/Dominio/class_precios.php');

switch ($_GET['opc']){
  case checkinres::$ID:
      if($_GET['result']==checkinres::$OK)
        $html='<div id="text">Checkin Realizado</div>;
      elseif($_GET['result']==checkinres::$ERR_RES)
        $html='<div id="text">Error: No ha seleccionado ninguna
        reserva</div>';
      elseif($_GET['result']==checkinres::$ERR_CHECK)
        $html='<div id="text">Error: El check-in ya ha sido
        realizado</div>';
  break;
  case cliente::$ID:
      if($_GET['result']==cliente::$OK)
        $html='<div id="text">Cliente Insertado</div>';
      elseif($_GET['result']==cliente::$ELIM)
        $html='<div id="text">El cliente ha sido eliminado</div>';
      elseif($_GET['result']==cliente::$MODIF)
        $html='<div id="text">El cliente ha sido modificado</div>';
      elseif($_GET['result']==cliente::$ERR)
        $html='<div id="text">Error: no se pudo realizar la
        operación</div>';
  break;
}
```

Para abrir la página *message_box.php,* se utiliza la utilidad Greybox, que mediante javascript nos permite mostrar el aviso superpuesto en la página donde se efectuó la operación, evitando así abrir ventanas diferentes.
A continuación se muestra un ejemplo de una llamada a message_box.

```
$res=$checkprev->make_checkinres($_POST);

/*la variable $messagebox incrusta el código javascript en el
evento onLoad del tag <body> de la página. */
$messagebox="GB_showCenter('Check-in',
'/view.php?page=message_box&opc=".checkinres::$ID."&result=".$res."
',100,300)";
```

6.4. Juegos de prueba

Una vez hemos diseñado todos los módulos de nuestra aplicación es necesario probar que funcionan correctamente. Para ello realizamos pruebas en los módulos por separado, en el conjunto de módulos integrados y en la fase de implantación del sistema.

Utilizamos la técnica de Caja negra que estudia las entradas y salidas del sistema buscando aquellos casos en que los comportamientos entrada-salida del sistema no están de acuerdo con las especificaciones de la aplicación.

Para realizar muchas pruebas se crearon subconjuntos de operaciones desde de la Capa de presentación hasta la Capa de gestión de datos para comprobar si las actualizaciones en la Base de Datos eran correctas. Paralelamente realizamos pruebas para comprobar la integración adecuada de todos los elementos del sistema.

Además, hemos tenido la oportunidad de poner el proyecto en marcha y hacer un seguimiento in situ del funcionamiento del mismo, periodo en el que inevitablemente nos encontramos con algún que otro problema, que pudimos solucionar en esta fase.
A pesar de haber realizado un sinfín de pruebas y combinaciones de posibles acciones durante la fase de implementación, imaginando las situaciones más complicadas que se podrían dar en un hotel, nos resultó curioso comprobar que la realidad supera la ficción y que en el momento de la implantación durante el mes que estuvimos en Guatemala se dieron situaciones que no nos habíamos planteado y que nos obligaron a realizar cambios y correcciones.
En parte, por este motivo estamos realizando periódicamente un control remoto del funcionamiento del sistema, para poder hacer frente a posibles incidencias, pues después de tanto tiempo y esfuerzos dedicados pretendemos que el programa que dejamos en pleno funcionamiento continúe trabajando de la misma manera y sea una herramienta útil para Casa Guatemala.

6.5. *Tecnologías Utilizadas*

WEB

World Wide Web es un sistema de información multimedia, hipertexto y distribuido. Es multimedia porque puede incluir texto, gráficos, sonido y vídeo. El hipertexto es una forma de escribir documentos basada en la utilización del ratón o el teclado para seleccionar una frase o imagen del documento que nos acceso a otro documento hipertexto (enlace) que puede estar ubicado en cualquier otro sitio.

El lenguaje de escritura de documentos hipertexto que se utiliza en páginas WWW es el HTML. La arquitectura del WWW es arquitectura cliente-servidor, es decir, existe una máquina encargada de servir los documentos que un cliente pide.

El protocolo de comunicación utilizado entre el navegador del cliente y el servidor web es el HTTP (Hypertext Transfer Protocol). Las cabeceras HTTP sirven para que el servidor web y el navegador se pasen información entre ellos.

La comunicación en aplicaciones web entre el cliente y el servidor es unidireccional, es decir, el cliente realiza una petición al servidor y recibe una respuesta, y no al revés.

Nos decantamos por la tecnología web, ya que cumple con los requisitos especificados y ambos componentes del grupo estamos familiarizados con ella.

Las aplicaciones web se caracterizan por un entorno intuitivo y visual, que facilita la interactuación con usuarios no habituados a la informática.

Además, existen infinidad de entornos de desarrollo y aplicaciones cliente y servidor con licencias de carácter público.

AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Ésto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de cuatro tecnologías ya existentes:

* XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
* Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
* El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
* XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Como el DHTML, LAMP o SPA, AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

En nuestra aplicación utilizamos esta técnica para mejorar la usabilidad de la aplicación.
Aplicamos Ajax principalmente para responder a los eventos provocados por el usuario, mostrando información específica en partes concretas de la página, así como para la recarga parcial de partes de élla.

Podemos ver varios ejemplos del uso de esta técnica en la página de calendario, como los globos de información de un determinado evento, el despliegue de literas al clickar en un alojamiento común o la modificación de eventos.

6.6. *Lenguajes de Programación*

HTML

HyperText Markup Language (HTML), es el lenguaje de marcado más extendido para la creación de páginas web, independiente de la plataforma en la que se utilice.

La presentación de la página depende directamente del navegador utilizado. El mismo documento no tiene el mismo formato en pantalla si se visualiza con diferentes navegadores (lynx, Netscape, Explorer,...). HTML se limita a describir la estructura y el contenido de un documento y no el formato de la página en pantalla.

Unas de las claves principales del HTML, además de la presentación, son la organización y la coherencia. Todas las páginas web tienen una estructura basada en etiquetas que facilita su uso y aprendizaje. Las etiquetas o tags describen el contenido del documento y el comportamiento de los elementos. Estas etiquetas son fragmentos de texto que van entre los símbolos < > y pueden tener atributos.

HTML es un lenguaje muy sencillo, pero al mismo tiempo muy limitado. Por esta razón generalmente se utiliza acompañado de otros lenguajes tales como el Javascript, que ofrecen funcionalidades adicionales.

SQL

Structured Query Language es un lenguaje de consulta estructurado de tipo declarativo, utilizado para el acceso a bases de datos relacionales, el cuál permite especificar diversos tipos de operaciones sobre las mismas.

Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre la misma.