Apéndice B

Código Fuente

B.1. Definición de la base de datos

```
1  CREATE DATABASE redes;
2  USE redes;
3
4  CREATE TABLE usuario (
5  idusuario tinyint UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
6  usuario  varchar(10) NOT NULL UNIQUE,
7  contrasena varchar(10) NOT NULL,
8  administrador boolean NOT NULL DEFAULT 0,
9
10 PRIMARY KEY (idusuario),
11 /* Se impide la coincidencia de pares usuario-contrasena */
12 UNIQUE INDEX (usuario,contrasena)
13 )ENGINE=InnoDB;
14
15 CREATE TABLE configuracion
16 (
17 idconfiguracion smallint UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
18 usuario  tinyint UNSIGNED NOT NULL,
19 nombre  varchar(20),
20 modificado datetime NOT NULL,
21
22 PRIMARY KEY (idconfiguracion),
23 FOREIGN KEY (usuario) REFERENCES usuario (idusuario) ON DELETE CASCADE
       ON UPDATE CASCADE
24 )ENGINE=InnoDB;
25
26
27 CREATE TABLE pc
28 (
29 idconfiguracion smallint UNSIGNED NOT NULL,
30 idpc  tinyint UNSIGNED NOT NULL,
31 ip  int UNSIGNED,
32 mascara  int UNSIGNED,
33 puerta  int UNSIGNED,
34 dns1  int UNSIGNED,
35 dns2  int UNSIGNED,
```

```
36
37  PRIMARY KEY (idconfiguracion,idpc),
38  FOREIGN KEY (idconfiguracion) REFERENCES configuracion (idconfiguracion) ON
        DELETE CASCADE ON UPDATE CASCADE
39  )ENGINE=InnoDB;
40
41
42  CREATE TABLE conmutador
43  (
44  idconfiguracion smallint UNSIGNED NOT NULL,
45  idconmutador tinyint UNSIGNED NOT NULL,
46  archivo  varchar(255),
47
48  PRIMARY KEY (idconfiguracion,idconmutador),
49  FOREIGN KEY (idconfiguracion) REFERENCES configuracion (idconfiguracion) ON
        DELETE CASCADE ON UPDATE CASCADE
50  )ENGINE=InnoDB;
51
52
53  CREATE TABLE red
54  (
55  idred  tinyint UNSIGNED NOT NULL UNIQUE,
56  default_config smallint UNSIGNED,
57  en_uso  boolean NOT NULL DEFAULT 0,
58  ult_acceso datetime,
59  ult_usuario tinyint UNSIGNED,
60
61  PRIMARY KEY (idred),
62  FOREIGN KEY (default_config) REFERENCES configuracion (idconfiguracion) ON
        DELETE SET NULL ON UPDATE CASCADE,
63  FOREIGN KEY (ult_usuario) REFERENCES usuario (idusuario) ON DELETE SET
        NULL ON UPDATE CASCADE
64  )ENGINE=InnoDB;
65
66  CREATE TABLE turno
67  (
68  idturno  tinyint UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
69  dia  tinyint UNSIGNED NOT NULL,
70  inicio  time NOT NULL,
71  fin   time NOT NULL,
72
73  PRIMARY KEY (idturno)
74  )ENGINE=InnoDB;
75
76
77  CREATE TABLE reserva
78  (
79  idusuario tinyint UNSIGNED NOT NULL UNIQUE,
80  idred  tinyint UNSIGNED NOT NULL,
81  idturno  tinyint UNSIGNED NOT NULL,
82
83  PRIMARY KEY (idusuario),
84  FOREIGN KEY (idusuario) REFERENCES usuario (idusuario) ON DELETE
        CASCADE ON UPDATE CASCADE,
85  FOREIGN KEY (idred) REFERENCES red (idred) ON DELETE CASCADE ON
        UPDATE CASCADE,
86  FOREIGN KEY (idturno) REFERENCES turno (idturno) ON DELETE CASCADE ON
         UPDATE CASCADE,
87  /* Se impide la coincidencia de pares idred-idturno */
88  UNIQUE INDEX (idred,idturno)
89  )ENGINE=InnoDB;
90
```

```
91
92  CREATE TABLE acceso
93  (
94  idacceso varchar(10) NOT NULL UNIQUE,
95  inicio  datetime NOT NULL,
96  fin   datetime NOT NULL,
97
98  PRIMARY KEY (idacceso)
99  )ENGINE=InnoDB;
```

B.2. Sitio Web

aplicacion.php

```
1   <?php
2   session_start();
3   require_once('admin/func_comunes.php');
4   ?>
5   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
6   <html>
7   <head>
8   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9   <title>Redes de Ordenadores</title>
10  <link href="/css/interfaz.css" rel="stylesheet" type="text/css" />
11  <script type="text/javascript" src="/js/jquery-1.3.2.min.js"></script>
12  <script type="text/javascript" src="/js/jquery-ui-1.7.2.custom.min.js"></script>
13  <script type="text/javascript" src="/js/jquery.blockUI.js"></script>
14  <script type="text/javascript" src="/js/jquery.cookie.js"></script>
15  <script type="text/javascript" src="/js/jquery.countdown.min.js"></script>
16  <script type="text/javascript" src="/js/jquery.qtip-1.0.0-rc3.min.js"></script>
17  <script type="text/javascript" src="/js/jquery.svg.min.js"></script>
18  <script type="text/javascript" src="/js/jquery.svgdom.min.js"></script>
19  <script type="text/javascript" src="/js/func_varias.js"></script>
20  <script type="text/javascript" src="/js/interfaz.js"></script>
21  </head>
22
23
24  <body>
25
26  <div id="main">
27  <!-- Cabecera -->
28  <div id="navigation" class="menu">
29  <ul>
30  <?php
31  try {
32    ob_start();
33    if (!Usuario::accesoAplicacion()) {
34      echo '<li><span class="disabled">Inicio</span></li>';
35    } else {
36      if (!isset($_SESSION['red'])) {
37        echo '<li><a href="" onclick="aplicacion.start(); return false;">Inicio</a></li>';
38      } else {
39  ?>
40    <li><a href="/admin/aplicacion.php" onclick="aplicacion.start(); return false;">Reinicio
          </a></li>
41    <li><a href="" onclick="return false">Configuracion</a>
42    <ul>
```

```
43      <li><a id="navadmin" href="" onclick="return false">Administrar</a></li>
44      <li><a id="navload" href="" onclick="return false">Cargar</a></li>
45      <li><a id="navstore" href="" onclick="return false">Guardar</a></li>
46    </ul>
47    </li>
48    <li><a id="navping" href="" onclick="return false;">Ping</a></li>
49    <li><a href="" onclick="aplicacion.end(); return false;">Cerrar</a></li>
50  <?php
51      }
52    }
53  ?>
54  </ul>
55  </div>
56
57  <!-- Contenido -->
58  <div id="main_content">
59  <noscript>
60  <div class="alert">La página que estás viendo requiere para su funcionamiento el uso de
        JavaScript. Si lo ha deshabilitado de manera intencionada, por favor vuelva a activarlo
        .</div>
61  </noscript>
62
63  <?php
64    if (!Usuario::accesoAplicacion())
65      echo '<div class="alert">En estos momentos no tiene permiso para acceder a la aplicació
          n</div>';
66    if (!isset($_SESSION['red'])) {
67  ?>
68  <div id="instrucciones">
69  <p>A continuación se proporcionan los enlaces para la descarga de las fichas técnicas y
        manuales de configuración de los conmutadores empleados en la práctica:</p>
70  <br />
71  <ul>
72    <li><a href="/downloads/Switch2510.zip">Modelo 2510</a></li>
73    <li><a href="/downloads/Switch2610.zip">Modelo 2610</a></li>
74  </ul>
75  <br />
76  <p>Por motivos relacionados con el funcionamiento interno de la aplicación, no se permite
        el empleo de los siguientes valores tanto en la configuración de los PCs como de los
        Conmutadores. Cualquier configuración que incluya cualquiera de estos valores es
        considerada errónea por la aplicación:</p>
77  <br />
78  <ul>
79    <li>Direcciones IP: <?php echo $cnxn_admin_red['red'].' / '.$cnxn_admin_red['mascara'];
          ?></li>
80    <li>VLAN: <?php echo $cnxn_admin_red['vlan']; ?></li>
81    <li>Puertos: <?php foreach ($cnxn_admin_red['puertos_conm'] as $puerto) echo $puerto.', '
          ; echo 'all'; ?></li>
82  </ul>
83  <br />
84  <h4>Advertencia: </h4><p>Esta aplicación hace uso de JavaScript, Cookies e imágenes en
        formato SVG. Si ha deshabilitado el
85  uso de los mismos en su navegador de manera intencionada, por favor vuelva ha habilitarlo.
        En el caso
86  de que su navegador no soporte el formato SVG (Internet Explorer), puede instalar el applet
87  <a href="http://www.adobe.com/svg/viewer/install/main.html" target="_blank">Adobe SVG
        Viewer</a>,
88  si bien en estos casos se recomienda el uso de Firefox para la realización de las prácticas
        .</p>
89  </div>
90  <?php
91    }
```

```
92    if (Usuario::accesoAplicacion() && isset($_SESSION['red'])) {
93      // Obtengo del archivo de configuración la información sobre la red
94      $pcs = array_keys($cnxn_red[1]['pc']);
95      $conmutadores = array_keys($cnxn_red[1]['conmutador']);
96  ?>
97
98
99  <!-- Panel con la imagen de la red -->
100 <div id="scheme" class="panel"><span class="title">Imagen</span>
101 <div id="diagrama"></div>
102 </div>
103
104
105 <!-- Panel para mostrar el resultado de las peticiones al servidor -->
106 <div id="console" class="panel"><span class="title">Consola<a href="" onclick="$('#
        resultados').children('p').remove(); return false;">(Limpiar)</a></span>
107 <div id="info"><p>Usuario: <span><?php echo $_SESSION['usuario']; ?></span>, Tiempo
        restante: <span id="time"></span></p></div>
108 <div>
109 <p>&gt;&gt <span id="cursor">|</span></p>
110 <span id="resultados"></span></div>
111 </div>
112
113
114 <!-- Panel con los forumalarios para la configuración de la red -->
115 <div id="config" class="panel"><span class="title"></span>
116 <h2></h2>
117
118
119 <!-- Formulario para la configuración de un PC -->
120 <form id="pc" class="narrow" action="" method="post" onsubmit="aplicacion.setPC($(this).
        serialize()); return false;">
121   <fieldset>
122     <input type="hidden" name="act" value="setPC" />
123     <input type="hidden" name="id" value="" />
124     <div><label><span>Dirección IP:</span><input type="text" name="ip" maxlength="15" value=
          "" /></label>
125     <label><span>Máscara subred:</span><input type="text" name="mascara" maxlength="15"
          value="" /></label>
126     <label><span>Puerta de enlace:</span><input type="text" name="puerta" maxlength="15"
          value="" /></label>
127     <label><span>Servidor DNS 1:</span><input type="text" name="dns1" maxlength="15" value="
          " /></label>
128     <label><span>Servidor DNS 2:</span><input type="text" name="dns2" maxlength="15" value="
          " /></label>
129     <br />
130     <label><input type="checkbox" name="default" /><span>Aplicar configuración por defecto</
          span></label>
131     <label><input type="checkbox" name="confirmado" /><span>Confirmar los cambios?</span></
          label></div>
132     <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
          Reiniciar" /></div>
133   </fieldset>
134 </form>
135
136
137 <!-- Formulario para la configuración de un Conmutador -->
138 <form id="conmutador" class="narrow" enctype="multipart/form-data" target="upload" action="
        /admin/aplicacion.php" method="post" onsubmit="$.blockUI()">
139   <fieldset>
140     <input type="hidden" name="act" value="setConmutador" />
```

```
141       <input type="hidden" name="id" value="" /> <input type="hidden" name="MAX_FILE_SIZE"
              value="5000" />
142       <div><label><span>Configuracion:</span>
143       <br />
144       <input type="file" name="archivo" /></label>
145       <br />
146       <label><input type="checkbox" name="default" /><span>Aplicar configuración por defecto</
              span></label>
147       <label><input type="checkbox" name="confirmado" /><span>Confirmar los cambios?</span></
              label></div>
148       <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
              Reiniciar" /></div>
149     </fieldset>
150     <iframe id="upload" name="upload" src="about:blank" onload="aplicacion.setConmutador()"
            ></iframe>
151   </form>
152
153
154   <!-- Formulario para adminstrar las configuraciones -->
155   <form id="adminconfig" " class="narrow" action="" method="post" onsubmit="aplicacion.
          configs($(this).serialize()); return false;">
156     <fieldset>
157       <div><label><span>Configuración:</span><br /><select name="idconfig">
158         <option value=""></option>
159         <optgroup class="configslist">
160         </optgroup>
161       </select></label>
162       <br />
163       <label><input type="radio" name="act" value="store" checked="checked" /><span>Renombrar
              configuración</span></label>
164       <label><span>Nombre:</span><input type="text" name="nombre" maxlength="15" value="" /></
              label>
165       <br />
166       <label><input type="radio" name="act" value="delete" /><span>Borrar configuración</span
              ></label>
167       <label><input type="checkbox" name="confirmado" /><span>Confirmar los cambios?</span></
              label></div>
168       <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
              Reiniciar" /></div>
169     </fieldset>
170   </form>
171
172
173   <!-- Formulario para cargar una configuracion -->
174   <form id="loadconfig" class="narrow" action="" method="post" onsubmit="aplicacion.configs($
          (this).serialize()); return false;">
175     <fieldset>
176       <input type="hidden" name="act" value="load" />
177       <div><label><span>Configuración:</span><br /><select name="idconfig">
178         <option value="default">Por defecto</option>
179         <optgroup class="configslist">
180         </optgroup>
181       </select></label> <br />
182       <label><input type="checkbox" name="confirmado" /><span>Confirmar los cambios?</span></
              label></div>
183       <div class="buttons"><input type="submit" value="Enviar" /></div>
184     </fieldset>
185   </form>
186
187
188   <!-- Formulario para almacenar una nueva configuración -->
```

```
189 <form id="storeconfig" class="narrow" action="" method="post" onsubmit="aplicacion.configs(
        $(this).serialize()); return false;">
190   <fieldset>
191     <input type="hidden" name="act" value="store" />
192     <div><label><span>Guardar como:</span><br /><select name="idconfig">
193       <option value="new" selected="selected">Nueva...</option>
194       <optgroup class="configslist" label="Sobreescribir existente">
195       </optgroup>
196     </select></label> <br />
197     <label><span>Nombre:</span><input type="text" name="nombre" maxlength="15" value="" /></
          label>
198     <label><input type="checkbox" name="confirmado" /><span>Confirmar los cambios?</span></
          label></div>
199     <div class="buttons"><input type="submit" value="Guardar" /><input type="reset" value="
          Reiniciar" /></div>
200   </fieldset>
201 </form>
202
203
204 <!-- Formulario para la realización de un ping desde uno de los PCs -->
205 <form id="ping" class="narrow" action="" method="post" onsubmit="aplicacion.ping($(this).
      serialize()); return false;">
206   <fieldset>
207     <input type="hidden" name="act" value="ping" />
208     <div><label><span>Origen:</span><select name="idorigen">
209 <?php
210     $i = 0;
211     foreach ($pcs as $id) {
212       $i++;
213       echo '<option value="'.$id.'">PC '.$i.'</option>'."\n";
214     }
215 ?>
216     </select></label> <label><span>IP Destino:</span><input type="text" name="ipdestino"
          maxlength="15" value="127.0.0.1" /></label></div>
217     <div class="buttons"><input type="submit" value="Ping" /></div>
218   </fieldset>
219 </form>
220 </div>
221
222 <script type="text/javascript">aplicacion.gui();</script>
223 <?php
224   }
225 }
226 catch (Exception $excepcion) {
227   ob_clean();
228   if (Usuario::esAdmin()) {
229     echo '<div class="error">'.(string) $excepcion.'</div>';
230   } else {
231     echo '<div class="error">Se he producido un error durante la carga de la página, pongase
           en contacto con el administrador.</div>';
232   }
233 }
234 ob_end_flush();
235 ?></div>
236 </div>
237 </body>
238 </html>
```

index.php

```
1  <?php
2  session_start();
3  require_once('admin/func_comunes.php');
4  ?>
5  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
6  <html>
7  <head>
8  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9  <title>Redes de Ordenadores</title>
10 <link href="/css/redes.css" rel="stylesheet" type="text/css" />
11 <script type="text/javascript" src="/js/jquery-1.3.2.min.js"></script>
12 <script type="text/javascript" src="/js/jquery-ui-1.7.2.custom.min.js"></script>
13 <script type="text/javascript" src="/js/jquery.cookie.js"></script>
14 <script type="text/javascript" src="/js/func_varias.js"></script>
15 <?php if (Usuario::esAdmin()) {?>
16 <script type="text/javascript" src="/js/administrar.js"></script>
17 <?php }?>
18 </head>
19
20
21 <body>
22
23 <div id="main">
24 <!-- Cabecera -->
25 <div id="navigation" class="menu">
26 <ul>
27   <li <?php if (isset($_GET['inicio'])) echo ' class="active"'; ?>><a href="/">Inicio</a></
       li>
28   <li <?php if (isset($_GET['aplicacion'])) echo ' class="active"'; ?>><a href="" onclick="
       ventanas.abrir('/aplicacion/', 'Aplicación', 'height=715,width=960,resizable=no,
       menubar=no,location=no,toolbar=no,status=no'); return false;">Aplicación</a></li>
29   <li <?php if (isset($_GET['turnos'])) echo ' class="active"'; ?>><a href="/turnos/">Turnos
       </a></li>
30 <?php if (Usuario::esAdmin()) {?>
31   <li <?php if (isset($_GET['administrar'])) echo ' class="active"'; ?>><a href="/
       administrar/">Administrar</a>
32   <ul>
33     <li><a href="/administrar/admin_usuarios/">Usuarios</a></li>
34     <li><a href="/administrar/admin_acceso/">Fechas de acceso</a></li>
35     <li><a href="/administrar/admin_turnos/">Turnos</a></li>
36     <li><a href="/administrar/admin_reservas/">Reservas</a></li>
37     <li><a href="" onclick="ventanas.abrir('/admin/log.php', 'Log', 'location=no,toolbar=no,
         status=no'); return false;">Log</a></li>
38   </ul></li>
39 <?php }?>
40 </ul>
41 </div>
42 <div id="login_button">
43 <ul>
44 <?php if (Usuario::logged()) {?>
45   <li><a href="" onclick="usuario.logout(); return false;">Logout</a></li>
46 <?php } else {?>
47   <li><a href="" onclick="usuario.login(); return false;">Login</a></li>
48 <?php }?>
49 </ul>
50 </div>
51
52 <!-- Contenido -->
53 <div id="main_content">
54 <noscript>
```

```
55  <div class="alert">La página que estás viendo requiere para su funcionamiento el uso de
        JavaScript. Si lo ha deshabilitado de manera intencionada, por favor vuelva a activarlo
        .</div>
56  </noscript>
57  <div id="msg"></div>
58  <?php
59  try {
60    ob_start();
61    if (isset($_GET['aplicacion'])) {
62      include('include/aplicacion.php');
63    } elseif (isset($_GET['turnos'])) {
64      include('include/turnos.php');
65    } elseif (isset($_GET['administrar'])) {
66      include('include/administrar.php');
67    } else {
68      include('include/inicio.php');
69    }
70  }
71  catch (Exception $excepcion) {
72    ob_clean();
73    if (Usuario::esAdmin()) {
74      echo '<div class="error">'.(string) $excepcion.'</div>';
75    } else {
76      echo '<div class="error">Se he producido un error durante la carga de la página, pó
            ngase en contacto con el administrador.</div>';
77    }
78  }
79  ob_end_flush();
80  ?></div>
81  </div>
82
83  <!-- Pie de página -->
84  <div id="footer">
85  <div id="links">
86  <h5>Enlaces:</h5>
87  <p><a href="http://trajano.us.es/~rafa/REDES/index.html">Redes de ordenadores</a> | <a href
        ="http://cdc-web1.us.es/">E.S.I.</a> | <a href="http://www.us.es/">Universidad de
        Sevilla</a></p>
88  </div>
89  <div id="contact_info">
90  <h4>Rafael M Estepa Alonso</h4>
91  <div>
92  <p>Web: <a href="http://trajano.us.es/~rafa/">http://trajano.us.es/~rafa/</a></p>
93  <p>E-Mail: <a href="mailto:rafa@trajano.us.es ">rafa@trajano.us.es</a></p>
94  </div>
95  <div>
96  <p>Tel: (+34) 95 448 73 84</p>
97  <p>Fax: (+34) 95 448 73 85</p>
98  </div>
99  </div>
100 </div>
101 </body>
102 </html>
```

admin/administrar.php

```
1  <?php
2  /**
```

```
3   * Clases para la administración de la base de datos
4   *
5   * @author Jesús Algeciras <jes.algeciras@gmail.com>
6   * @version 2010-10-24
7   *
8   */
9   require_once('func_comunes.php');
10
11  /**
12   * Clase que deberan extender todas las clases de administración
13   * lo que permitirá un tratamiento común de los formularios de
14   * administración independiente del manejador asignado
15   */
16  abstract class Administrar {
17    /**
18     * @var Database
19     */
20    protected $database;
21
22    function __construct() {
23      $this->database = Database::conectar();
24    }
25
26    /**
27     * Aadir a la base de datos
28     * @param $inputs campos del formulario
29     * @return array errores producidos
30     */
31    abstract function add($inputs);
32
33    /**
34     * Editar elemento de la base de datos
35     * @param $inputs campos del formulario
36     * @return array errores producidos
37     */
38    abstract function edit($inputs);
39
40    /**
41     * Eliminar elemento de la base de datos
42     * @param $inputs campos del formulario
43     * @return array errores producidos
44     */
45    abstract function del($inputs);
46
47    /**
48     * Eleminiar todos los elementos de la base de datos
49     * @param $inputs campos del formulario
50     * @return array errores producidos
51     */
52    abstract function delall($inputs);
53  }
54
55  /**
56   * Administrador de las fechas de acceso
57   */
58  class AdministrarAcceso extends Administrar {
59
60    function add($inputs) {
61      // Validación
62      $validacion = new Validacion;
63      if (!isset($inputs['confirmado']))
64        $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
```

```
65      $validacion->esFecha('fecha_inicio', $inputs['fecha_inicio']);
66      $validacion->esHora('hora_inicio', $inputs['hora_inicio']);
67      $validacion->esFecha('fecha_fin', $inputs['fecha_fin']);
68      $validacion->esHora('hora_fin', $inputs['hora_fin']);
69
70      if ((!isset($inputs['idacceso'])) || (($inputs['idacceso'] != 'reservas') && ($inputs['
            idacceso'] != 'aplicacion')))
71        $validacion->setErrores('idacceso', 'No ha seleccionado fecha de acceso');
72
73      if (!count($validacion->getErrores())) {
74        if (strtotime($inicio) > strtotime($fin))
75          $validacion->setErrores('fecha_fin','La fecha final es anterior a la de inicio');
76      }
77
78      // Se procesa el formulario si no hay errores
79      $errores = $validacion->getErrores();
80      if (!count($errores)) {
81        $inicio = $inputs['fecha_inicio'].' '.$inputs['hora_inicio'].':00';
82        $fin = $inputs['fecha_fin'].' '.$inputs['hora_fin'].':00';
83        $this->database->query('DELETE FROM acceso WHERE idacceso="'.$inputs['idacceso'].'"');
84        $this->database->query('INSERT INTO acceso (idacceso,inicio,fin) VALUES ("'.$inputs['
              idacceso'].'","'.$inicio.'","'.$fin.'")');
85      }
86      return $errores;
87    }
88
89    function edit($inputs) {
90      return $this->add($inputs);
91    }
92
93    function del($inputs) {
94      // Validación
95      $validacion = new Validacion;
96      if (!isset($inputs['confirmado']))
97        $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
98      if ((!isset($inputs['idacceso'])) || (($inputs['idacceso'] != 'reservas') && ($inputs['
            idacceso'] != 'aplicacion')))
99        $validacion->setErrores('idacceso', 'No ha seleccionado fecha de acceso');
100
101     // Se procesa el formulario si no hay errores
102     $errores = $validacion->getErrores();
103     if (!count($errores))
104       $this->database->query('DELETE FROM acceso WHERE idacceso="'.$inputs['idacceso'].'"');
105
106     return $errores;
107   }
108
109   function delall($inputs) {
110     // Validación
111     $validacion = new Validacion;
112     if (!isset($inputs['confirmado'])) {
113       $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
114     } else {
115       $this->database->query('TRUNCATE TABLE acceso');
116     }
117     return $validacion->getErrores();
118   }
119 }
120
121
122 /**
123  * Administrador de turnos
```

```
124   */
125   class AdministrarTurnos extends Administrar {
126
127     function add($inputs) {
128       // Validación
129       $validacion = new Validacion;
130       if (!isset($inputs['confirmado']))
131         $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
132       if ($validacion->esEntero('dia', $inputs['dia']) && ($inputs['dia'] < 1 || $inputs['dia']
              > 5))
133         $validacion->setErrores('dia', 'Formato erroneo');
134       $validacion->esHora('inicio', $inputs['inicio']);
135       $validacion->esHora('fin', $inputs['fin']);
136       if ($validacion->esEntero('sesiones', $inputs['sesiones']) && $inputs['sesiones'] == 0)
137         $validacion->setErrores('sesiones', 'Ha de ser mayor que 0');
138
139       if (!count($validacion->getErrores())) {
140         if (strtotime($inputs['inicio']) > strtotime($inputs['fin']))
141           $validacion->setErrores('fin', 'La hora final es anterior a la de inicio');
142
143         // Se comprueba si los turnos entran en conflicto entre sí
144         if (!$this->validarTurno($inputs['dia'], $inputs['inicio'], $inputs['fin']))
145           $validacion->setErrores('dia', 'Conflicto con los turnos ya establecidos');
146       }
147
148       // Se procesa el formulario si no hay errores
149       $errores = $validacion->getErrores();
150       try {
151         if (!count($errores)) {
152           $this->database->autocommit(FALSE);
153           $inicio = strtotime($_POST['inicio']);
154           $fin = strtotime($_POST['fin']);
155           $duracion = ($fin - $inicio)/($inputs['sesiones']);
156
157           for($hora = $inicio; $hora < $fin; $hora += $duracion)
158             $this->database->query('INSERT INTO turno (dia,inicio,fin) VALUES ('.$inputs['dia'
                    ].','"'.date('H:i',$hora).'","'.date('H:i',$hora + $duracion - 300).'")');
159           $this->database->commit();
160           $this->database->autocommit(TRUE);
161         }
162         return $errores;
163       }
164       catch (DBException $error) {
165         $this->database->rollback();
166         $this->database->autocommit(TRUE);
167         throw $error;
168       }
169     }
170
171     function edit($inputs) {
172       // Validación
173       $validacion = new Validacion;
174       if (!isset($inputs['confirmado']))
175         $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
176       if ($validacion->esEntero('idturno', $inputs['idturno'])) {
177         $respuesta = $this->database->query('SELECT * FROM turno WHERE idturno='.$inputs['
                idturno']);
178         if ($respuesta->num_rows > 0) {
179           // Los datos no proporcionados en el formulario se obtienen de la base de datos
180           $datos = $respuesta->fetch_assoc();
181           $value['dia'] = ($inputs['dia']) ? $inputs['dia'] : $datos['dia'];
```

```
182         $value['inicio'] = ($inputs['inicio']) ? $inputs['inicio'] : substr($datos['inicio'
                ],0,5);
183         $value['fin'] = ($inputs['fin']) ? $inputs['fin'] : substr($datos['fin'],0,5);
184
185         if ($validacion->esEntero('dia', $value['dia']) && ($value['dia'] < 1 || $value['dia'
                ] > 5))
186           $validacion->setErrores('dia', 'Formato erroneo');
187         $validacion->esHora('inicio', $value['inicio']);
188         $validacion->esHora('fin', $value['fin']);
189
190         if (!count($validacion->getErrores())) {
191           if (strtotime($value['inicio']) > strtotime($value['fin']))
192             $validacion->setErrores('fin', 'La hora final es anterior a la de inicio');
193
194           // Se comprueba si los turnos entran en conflicto entre sí
195           if (!$this->validarTurno($value['dia'], $value['inicio'], $value['fin'], $inputs['
                  idturno']))
196             $validacion->setErrores('dia', 'Conflicto con los turnos ya establecidos');
197         }
198       } else {
199         $validacion->setErrores('idturno', 'Turno no valido');
200       }
201       $respuesta->free();
202     }
203
204     // Se procesa el formulario si no hay errores
205     $errores = $validacion->getErrores();
206     if (!count($errores))
207       $this->database->query('UPDATE turno SET dia='.$value['dia'].',inicio="'.$value['
              inicio'].'",fin="'.$value['fin'].'" WHERE idturno='.$inputs['idturno']);
208     return $errores;
209   }
210
211   function del($inputs) {
212     // Validación
213     $validacion = new Validacion;
214     if (!isset($inputs['confirmado']))
215       $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
216     if ($validacion->esEntero('idturno', $inputs['idturno'])) {
217       $respuesta = $this->database->query('SELECT * FROM turno WHERE idturno='.$inputs['
              idturno']);
218       if ($respuesta->num_rows == 0)
219         $validacion->setErrores('idturno', 'Turno no valido');
220       $respuesta->free();
221     }
222
223     // Se procesa el formulario si no hay errores
224     $errores = $validacion->getErrores();
225     if (!count($errores))
226       $this->database->query('DELETE FROM turno WHERE idturno="'.$inputs['idturno'].'"');
227     return $errores;
228   }
229
230   function delall($inputs) {
231     $validacion = new Validacion;
232     if (!isset($inputs['confirmado'])) {
233       $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
234     } else {
235       $this->database->query('TRUNCATE TABLE turno');
236       $this->database->query('ALTER TABLE turno auto_increment=1');
237     }
238     return $validacion->getErrores();
```

```
239     }
240
241     /**
242      * Función que comprueba conflictos entre los turnos
243      * @param $dia
244      * @param $hora_inicio
245      * @param $hora_fin
246      * @param $idturno para descartar conflictos con sigo mismo
247      * @return boolean
248      */
249     private function validarTurno($dia, $inicio, $fin, $idturno = NULL) {
250       $peticion = 'SELECT idturno FROM turno WHERE dia='.$dia.' AND ("'.$inicio.'" BETWEEN
              inicio AND fin OR "'.$fin.'" BETWEEN inicio AND fin OR inicio BETWEEN "'.$inicio.'"
               AND "'.$fin.'" OR fin BETWEEN "'.$inicio.'" AND "'.$fin.'")';
251       $peticion = $idturno ? ($peticion.' AND idturno!='.$idturno) : $peticion;
252       $respuesta = $this->database->query($peticion);
253       $conflicto = ($respuesta->num_rows > 0);
254       $respuesta->free();
255       if ($conflicto) {
256         return FALSE;
257       } else {
258         return TRUE;
259       }
260     }
261   }
262
263
264   /**
265    * Administrador de usuarios
266    */
267   class AdministrarUsuarios extends Administrar {
268
269     function add($inputs) {
270       // Validación
271       $validacion = new Validacion;
272       if (!isset($inputs['confirmado']))
273         $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
274       if ($validacion->esEntero('num_usuarios', $inputs['num_usuarios']) && $inputs['
              num_usuarios'] == 0)
275         $validacion->setErrores('num_usuarios', 'Ha de ser mayor que 0');
276
277       // Se procesa el formulario si no hay errores
278       $errores = $validacion->getErrores();
279       try {
280         if (!count($errores)) {
281           $this->database->autocommit(FALSE);
282           // Se comprueba el último usuario introducido
283           $respuesta = $this->database->query('SELECT usuario FROM usuario WHERE administrador
                  =0 AND usuario REGEXP "^grupo" ORDER BY usuario DESC LIMIT 1');
284           $datos = $respuesta->fetch_assoc();
285           $datos = preg_replace('/[^0-9]/', '', $datos['usuario']);
286           $respuesta->free();
287
288           $password = array_flip(array_merge(range('a','z'),range(0,9)));
289           for ($i = 1; $i <= $inputs['num_usuarios']; $i++)
290             $this->database->query('INSERT INTO usuario (usuario,contrasena,administrador)
                    VALUES ("'.sprintf('grupo%02d',($datos + $i)).'","'.implode("",array_rand(
                    $password, 6)).'",0)');
291           $this->database->commit();
292           $this->database->autocommit(TRUE);
293         }
294         return $errores;
```

```
295       }
296       catch (DBException $error) {
297         $this->database->rollback();
298         $this->database->autocommit(TRUE);
299         throw $error;
300       }
301     }
302
303     function edit($inputs) {
304       // Validación
305       $validacion = new Validacion;
306       if (!isset($inputs['confirmado']))
307         $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
308       if ($validacion->esEntero('idusuario', $inputs['idusuario'])) {
309         $respuesta = $this->database->query('SELECT * FROM usuario WHERE idusuario='.$inputs['
              idusuario']);
310         if ($respuesta->num_rows > 0) {
311           // Los datos no proporcionados en el formulario se obtienen de la base de datos
312           $datos = $respuesta->fetch_assoc();
313           $respuesta->free();
314           $value['usuario'] = ($inputs['usuario']) ? $inputs['usuario'] : $datos['usuario'];
315           $value['contrasena'] = ($inputs['contrasena']) ? $inputs['contrasena'] : $datos['
                contrasena'];
316           $value['administrador'] = ($inputs['administrador'] != '') ? $inputs['administrador'
                ] : $datos['administrador'];
317
318           $validacion->esTexto('usuario', $value['usuario']);
319           $validacion->esTexto('contrasena', $value['contrasena']);
320           $validacion->esBinario('administrador', $value['administrador']);
321
322           if (!count($validacion->getErrores())) {
323             $respuesta = $this->database->query('SELECT usuario FROM usuario WHERE usuario="'.
                   $inputs['usuario'].'"');
324             if ($respuesta->num_rows > 0)
325               $validacion->setErrores('usuario', 'Ya existe un usuario con ese nombre');
326             $respuesta->free();
327
328             // Se comprueba que no se eliminen los permisos del administrador principal
                   propietario de las configuraciones por defecto del sistema
329             if ($inputs['administrador'] != '' && $inputs['administrador'] == 0) {
330               $respuesta = $this->database->query('SELECT usuario FROM red NATURAL LEFT JOIN
                     configuracion WHERE idred=1');
331               $datos = $respuesta->fetch_assoc();
332               $respuesta->free();
333               if ($datos['usuario'] == $inputs['idusuario'])
334                 $validacion->setErrores('administrador', 'Cuenta del administrador principal');
335             }
336           }
337         } else {
338           $respuesta->free();
339           $validacion->setErrores('idusuario', 'Usuario no valido');
340         }
341       }
342
343       // Se procesa el formulario si no hay errores
344       $errores = $validacion->getErrores();
345       if (!count($errores))
346         $this->database->query('UPDATE usuario SET usuario="'.$value['usuario'].'",contrasena
              ="'.$value['contrasena'].'",administrador='.$value['administrador'].' WHERE
              idusuario='.$inputs['idusuario']);
347       return $errores;
348     }
```

```
349
350   function del($inputs) {
351     // Validación
352     $validacion = new Validacion;
353     if (!isset($inputs['confirmado']))
354       $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
355     if ($validacion->esEntero('idusuario', $inputs['idusuario'])) {
356       $respuesta = $this->database->query('SELECT administrador FROM usuario WHERE idusuario
              ='.$inputs['idusuario']);
357       if ($respuesta->num_rows > 0) {
358         $datos = $respuesta->fetch_assoc();
359         $respuesta->free();
360         // Se comprueba que no se elimine el administrador principal propietario de las
               configuraciones por defecto del sistema
361         if ($datos['administrador'] == 1) {
362           $respuesta = $this->database->query('SELECT usuario FROM red NATURAL LEFT JOIN
                configuracion WHERE idred=1');
363           $datos = $respuesta->fetch_assoc();
364           $respuesta->free();
365           if ($datos['usuario'] == $inputs['idusuario'])
366             $validacion->setErrores('idusuario', 'Cuenta del administrador principal');
367         }
368       } else {
369         $respuesta->free();
370         $validacion->setErrores('idusuario', 'Usuario no valido');
371       }
372     }
373
374     // Se procesa el formulario si no hay errores
375     $errores = $validacion->getErrores();
376     if (!count($errores))
377       $this->borrarUsuario($inputs['idusuario']);
378     return $errores;
379   }
380
381   function delall($inputs) {
382     // Validación
383     $validacion = new Validacion;
384     if (!isset($inputs['confirmado'])) {
385       $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
386     } else {
387       // Se eliminan todas las cuentas sin derecho de administración
388       $respuesta = $this->database->query('SELECT idusuario FROM usuario WHERE administrador
              =0');
389       for ($i = 0; $i < $respuesta->num_rows; $i++) {
390         $datos = $respuesta->fetch_assoc();
391         $this->borrarUsuario($datos['idusuario']);
392       }
393       $respuesta->free();
394     }
395     return $validacion->getErrores();
396   }
397
398   /**
399    * Función que elimina todos los archivos pertenecientes
400    * al usuario y lo borra de la base de datos
401    * @param $idusuario
402    * @return boolean
403    */
404   private function borrarUsuario($idusuario) {
405     $respuesta = $this->database->query('SELECT archivo FROM configuracion NATURAL LEFT JOIN
            conmutador WHERE usuario='.$idusuario);
```

```
406       for ($i = 0; $i < $respuesta->num_rows; $i++) {
407         $datos = $respuesta->fetch_assoc();
408         $file = FILES_STORAGE.$datos['archivo'];
409         $respuesta->free();
410         if (is_file($file))
411           Archivo::borrar($file);
412       }
413       $this->database->query('DELETE FROM usuario WHERE idusuario='.$idusuario);
414       return TRUE;
415     }
416   }
417
418
419   /**
420    * Administrador de las reservas
421    */
422   class AdministrarReservas extends Administrar {
423
424     function add($inputs) {
425       // Validación
426       $validacion = new Validacion;
427       if (!isset($inputs['confirmado']))
428         $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
429
430       $validacion->esEntero('idusuario', $inputs['idusuario']);
431       $validacion->esEntero('idturno', $inputs['idturno']);
432       $validacion->esEntero('idred', $inputs['idred']);
433       if (!count($validacion->getErrores())) {
434         // Se comprueba la existencia del turno usuario y red
435         $respuesta = $this->database->query('SELECT * FROM usuario WHERE idusuario='.$inputs['
              idusuario']);
436         if ($respuesta->num_rows == 0)
437           $validacion->setErrores('idusuario', 'Usuario no valido');
438         $respuesta->free();
439
440         $respuesta = $this->database->query('SELECT * FROM turno WHERE idturno='.$inputs['
              idturno']);
441         if ($respuesta->num_rows == 0)
442           $validacion->setErrores('idturno', 'Turno no valido');
443         $respuesta->free();
444
445         $respuesta = $this->database->query('SELECT * FROM red WHERE idred='.$inputs['idred'])
              ;
446         if ($respuesta->num_rows == 0)
447           $validacion->setErrores('idred', 'Red no valida');
448         $respuesta->free();
449       }
450
451       // Se procesa el formulario si no hay errores
452       $errores = $validacion->getErrores();
453       if (!count($errores)) {
454         $this->database->query('DELETE FROM reserva WHERE idusuario='.$inputs['idusuario'].'
              OR (idturno='.$inputs['idturno'].' AND idred='.$inputs['idred'].')');
455         $this->database->query('INSERT INTO reserva (idusuario,idred,idturno) VALUES ('.
              $inputs['idusuario'].','.$inputs['idred'].','.$inputs['idturno'].')');
456       }
457       return $errores;
458     }
459
460     function edit($inputs) {
461       return $this->add($inputs);
462     }
```

```
463
464   function del($inputs) {
465     // Validación
466     $validacion = new Validacion;
467     if (!isset($inputs['confirmado']))
468       $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
469     if ($validacion->esEntero('idusuario', $inputs['idusuario'])) {
470       $respuesta = $this->database->query('SELECT idusuario FROM reserva WHERE idusuario='.
              $inputs['idusuario']);
471       if ($respuesta->num_rows == 0)
472         $validacion->setErrores('idusuario', 'Usuario no valido');
473       $respuesta->free();
474     }
475
476     // Se procesa el formulario si no hay errores
477     $errores = $validacion->getErrores();
478     if (!count($errores))
479       $this->database->query('DELETE FROM reserva WHERE idusuario='.$inputs['idusuario']);
480     return $errores;
481   }
482
483   function delall($inputs) {
484     // Validación
485     $validacion = new Validacion;
486     if (!isset($inputs['confirmado'])) {
487       $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
488     } else {
489       $this->database->query('TRUNCATE TABLE reserva');
490     }
491     return $validacion->getErrores();
492   }
493 }
494 ?>
```

admin/aplicacion.php

```
1  <?php
2  /**
3   * Script de la aplicación
4   *
5   * Para la comunicación entre el cliente y este script
6   * se emplea AJAX utilizando JSON como notación para la
7   * transferencia de información al cliente
8   *
9   * @author Jesús Algeciras <jes.algeciras@gmail.com>
10  * @version 2010-10-24
11  *
12  */
13
14 if ((isset($_SERVER['HTTP_X_REQUESTED_WITH']) && $_SERVER['HTTP_X_REQUESTED_WITH'] == '
       XMLHttpRequest') || (isset($_POST['act']) && $_POST['act'] == 'setConmutador')) {
15   session_start();
16   require_once('func_comunes.php');
17   require_once('red.php');
18
19   try {
20     // Se lleva a cabo la petición correspondiente y se devuelve el resultado en formato
           JSON.
```

```
21      // La estructura de la respuesta es siempre la misma:
22      // array( stdout => (str), stderr => array(error => (str), info => (str)))
23      // Esta respuesta es tratada segun corresponda en el navegador mediante Javascript
24      $respuesta['stdout'] = '';
25      $respuesta['stderr']['error'] = FALSE; // Tipos de errores: excepcion, validacion,
            aplicacion
26      $respuesta['stderr']['info'] = array();
27
28      $log = LogFile::abrirLog();
29      $database = Database::conectar();
30      if (!Usuario::accesoAplicacion()) {
31        $respuesta['stderr']['error'] = 'excepcion';
32        $respuesta['stderr']['info'] = array('Otro miembro de su grupo ha accedido a la
              aplicación y esta haciendo uso de ella en estos momentos. Para evitar este error
              en el futuro, no usar la aplicación desde más de un PC al mismo tiempo.');
33      } else {
34        // Si la aplicación ya ha sido iniciada correctamente, existirá una instancia de
35        // la clase Red almacenada. Se recupera dicha instancia.
36        if (isset($_SESSION['red']))
37          $red = unserialize($_SESSION['red']);
38
39        switch ($_POST['act']) {
40          // Inicio de la aplicación
41          case 'start':
42            $log->log('Iniciando la aplicación');
43            Usuario::accedeRed();
44
45            // Se obtienen los datos necesarios del archivo de configuración
46            $pcs = $cnxn_red[$_SESSION['idred']]['pc'];
47            $conmutadores = $cnxn_red[$_SESSION['idred']]['conmutador'];
48
49            // Se crea y configura la red
50            $red = new Red;
51            foreach ($pcs as $id => $pc)
52              $red->add(new PC($id, $pc['ip'], $pc['puerto']));
53            foreach ($conmutadores as $id => $conmutador)
54              $red->add(new Conmutador($id, $conmutador['ip'], $conmutador['password']));
55            $respuesta['stderr'] = $red->loadConfig($red->defaultConfig());
56            if (!$respuesta['stderr']['error'])
57              $respuesta['stderr'] = $red->sendConfig();
58
59            // Si se produce un error se aborta el inicio de la aplicación
60            if ($respuesta['stderr']['error']) {
61              Usuario::abandonaRed();
62            } else {
63              $respuesta['stdout'] = 'La aplicación ha sido iniciada con éxito';
64              // Se crean cookies con la información necesaria para la aplicación
65              foreach ($red->getPC() as $pc)
66                setcookie('pc'.$pc->getID(), urlencode(json_encode($pc->getConfig())),mktime
                      (substr($_SESSION['turno_fin'], 0, 2), substr($_SESSION['turno_fin'], 3,
                      2), 0), '/aplicacion/');
67              foreach ($cnxn_red[$_SESSION['idred']]['conmutador'] as $id => $conmutador) {
68                $topologia = array('modelo' => $conmutador['modelo']);
69                $topologia = array('conexiones' => $used_ports[$id]);
70                setcookie('conmutador'.$id, urlencode(json_encode($topologia)),mktime(substr
                      ($_SESSION['turno_fin'], 0, 2), substr($_SESSION['turno_fin'], 3, 2), 0),
                      '/aplicacion/');
71              }
72              $dbdata = $database->query('SELECT * FROM configuracion WHERE usuario='.
                    $_SESSION['idusuario']);
73              for ($i = 0; $i < $dbdata->num_rows; $i++) {
74                $config = $dbdata->fetch_assoc();
```

```
75              setcookie('config'.$config['idconfiguracion'], urlencode(json_encode($config))
                    , mktime(substr($_SESSION['turno_fin'], 0, 2), substr($_SESSION['
                    turno_fin'], 3, 2), 0), '/aplicacion/');
76            }
77            setcookie('idred', urlencode(json_encode($_SESSION['idred'])), mktime(substr(
                  $_SESSION['turno_fin'], 0, 2), substr($_SESSION['turno_fin'], 3, 2), 0), '/
                  aplicacion/');
78            $dbdata->free();
79            // Se almacena el objeto Red creado
80            $_SESSION['red'] = serialize($red);
81          }
82          break;
83
84
85        // Cierre de la aplicación
86        case 'end':
87          $log->log('Cerrando la aplicación');
88
89          // Se devuelve la red a su estado por defecto
90          $respuesta['stderr'] = $red->loadConfig($red->defaultConfig());
91          if (!$respuesta['stderr']['error'])
92            $respuesta['stderr'] = $red->sendConfig();
93
94          if (!$respuesta['stderr']['error']) {
95            unset($_SESSION['red']);
96            Usuario::abandonaRed();
97            $respuesta['stdout'] = 'La aplicación ha sido finalizada con éxito';
98            foreach($_COOKIE as $name => $value)
99              setcookie($name, $value, 1, '/aplicacion/');
100         } else {
101           $_SESSION['red'] = serialize($red);
102         }
103         break;
104
105
106       // Cargar configuración
107       case 'load':
108         // Validación
109         $validacion = new Validacion;
110         if (!isset($_POST['confirmado']))
111           $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
112         if ($_POST['idconfig'] != 'default') {
113           if ($validacion->esEntero('idconfig', $_POST['idconfig'])) {
114             $dbdata = $database->query('SELECT * FROM configuracion WHERE usuario='.
                      $_SESSION['idusuario'].' AND idconfiguracion='.$_POST['idconfig']);
115             if ($dbdata->num_rows == 0)
116               $validacion->setErrores('idconfig', 'Configuracion inexistente');
117             $dbdata->free();
118           }
119         }
120
121         // Carga de la configuración
122         if (count($validacion->getErrores()) == 0) {
123           $log->log('Cargando la configuración en la red '.$_POST['idconfig']);
124           if ($_POST['idconfig'] == 'default') {
125             $respuesta['stderr'] = $red->loadConfig($red->defaultConfig());
126           } else {
127             $respuesta['stderr'] = $red->loadConfig($_POST['idconfig']);
128           }
129           if (!$respuesta['stderr']['error'])
130             $respuesta['stderr'] = $red->sendConfig();
131
```

```
132             if (!$respuesta['stderr']['error']) {
133               $respuesta['stdout'] = 'La configuración ha sido cargada con éxito';
134               foreach ($red->getPC() as $pc)
135                 setcookie('pc'.$pc->getID(), urlencode(json_encode($pc->getConfig())),
                        mktime(substr($_SESSION['turno_fin'], 0, 2), substr($_SESSION['
                        turno_fin'], 3, 2), 0), '/aplicacion/');
136             }
137           } else {
138             $respuesta['stderr']['error'] = 'validacion';
139             $respuesta['stderr']['info'] = $validacion->getErrores();
140           }
141           $_SESSION['red'] = serialize($red);
142           break;
143
144
145         // Almacenar configuración
146         case 'store':
147           $log->log('Guardando configuración de red');
148
149           // Validación
150           $validacion = new Validacion;
151           if (!isset($_POST['confirmado']))
152             $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
153
154           $nombre = str_replace(" ","_",$_POST['nombre']);
155           $validacion->esTexto('nombre', $nombre);
156
157           if ($_POST['idconfig'] != 'new' && $validacion->esEntero('idconfig', $_POST['
                  idconfig'])) {
158             if ($validacion->esEntero('idconfig', $_POST['idconfig'])) {
159               $dbdata = $database->query('SELECT * FROM configuracion WHERE usuario='.
                      $_SESSION['idusuario'].' AND idconfiguracion='.$_POST['idconfig']);
160               if ($dbdata->num_rows == 0)
161                 $validacion->setErrores('idconfig', 'Configuración inexistente');
162               $dbdata->free();
163             }
164           }
165
166           // Se guarda la configuración en ausencia de errores
167           if (count($validacion->getErrores()) == 0) {
168             $respuesta['stderr'] = $red->storeConfig($_POST['idconfig'], $nombre);
169             if (!$respuesta['stderr']['error']) {
170               $respuesta['stdout'] = 'La configuración ha sido guardada con éxito';
171               $dbdata = $database->query('SELECT * FROM configuracion WHERE usuario='.
                      $_SESSION['idusuario']);
172               for ($i = 0; $i < $dbdata->num_rows; $i++) {
173                 $config = $dbdata->fetch_assoc();
174                 setcookie('config'.$config['idconfiguracion'], urlencode(json_encode($config
                        )), mktime(substr($_SESSION['turno_fin'], 0, 2), substr($_SESSION['
                        turno_fin'], 3, 2), 0), '/aplicacion/');
175               }
176               $dbdata->free();
177             }
178           } else {
179             $respuesta['stderr']['error'] = 'validacion';
180             $respuesta['stderr']['info'] = $validacion->getErrores();
181           }
182           $_SESSION['red'] = serialize($red);
183           break;
184
185
186         // Borrar confguración
```

```
187         case 'delete':
188           // Validación
189           $validacion = new Validacion;
190           if (!isset($_POST['confirmado']))
191             $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
192           if ($validacion->esEntero('idconfig', $_POST['idconfig'])) {
193             $dbdata = $database->query('SELECT default_config FROM red WHERE default_config=
                    '.$_POST['idconfig']);
194             if ($dbdata->num_rows > 0)
195               $validacion->setErrores('idconfig', 'No es posible borrar esta configuración');
196             $dbdata->free();
197             if($validacion->esEntero('idconfig', $_POST['idconfig'])) {
198               $dbdata = $database->query('SELECT * FROM configuracion WHERE usuario='.
                      $_SESSION['idusuario'].' AND idconfiguracion='.$_POST['idconfig']);
199               if ($dbdata->num_rows == 0)
200                 $validacion->setErrores('idconfig', 'Configuración inexistente');
201               $dbdata->free();
202             }
203           }
204
205           // Borrado de la configuración
206           if (count($validacion->getErrores()) == 0) {
207             $log->log('Borrando la configuración de red '.$_POST['idconfig']);
208             if ($red->deleteConfig($_POST['idconfig'])) {
209               $respuesta['stdout'] = 'La configuración ha sido borrada con éxito';
210               setcookie('config'.$_POST['idconfig'], '', 1, '/aplicacion/');
211             }
212           } else {
213             $respuesta['stderr']['error'] = 'validacion';
214             $respuesta['stderr']['info'] = $validacion->getErrores();
215           }
216           $_SESSION['red'] = serialize($red);
217           break;
218
219
220         // Configurar PC
221         case 'setPC':
222           // Validación
223           $validacion = new Validacion;
224           if ($validacion->esEntero('id', $_POST['id'])) {
225             $pc = $red->getPC($_POST['id']);
226             if ($pc) {
227               $log->log('Configurando PC '. $_POST['id']);
228               if (!isset($_POST['confirmado']))
229                 $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
230               if (isset($_POST['default'])) {
231                 $pc->loadConfig($red->defaultConfig());
232               } else {
233                 $config = array('ip' => $_POST['ip'],
234                         'mascara' => $_POST['mascara'],
235                         'puerta' => $_POST['puerta'],
236                         'dns1' => $_POST['dns1'],
237                         'dns2' => $_POST['dns2']);
238                 $validacion->esPC($config);
239                 if (count($validacion->getErrores()) == 0)
240                   $pc->setConfig($config);
241               }
242
243               // Si la configuración es valida se envia al PC
244               if (count($validacion->getErrores()) == 0) {
245                 $respuesta['stderr'] = $pc->sendConfig();
246                 if (!$respuesta['stderr']['error']) {
```

```
247                   $respuesta['stdout'] = 'La configuración ha sido establecida con éxito';
248                   setcookie('pc'.$pc->getID(), urlencode(json_encode($pc->getConfig())),
                          mktime(substr($_SESSION['turno_fin'], 0, 2), substr($_SESSION['
                          turno_fin'], 3, 2), 0), '/aplicacion/');
249                 }
250               } else {
251                 $respuesta['stderr']['error'] = 'validacion';
252                 $respuesta['stderr']['info'] = $validacion->getErrores();
253               }
254             }
255           }
256           $_SESSION['red'] = serialize($red);
257           break;
258
259
260         // Configurar Conmutador
261         case 'setConmutador':
262           // Validación
263           $validacion = new Validacion;
264           if ($validacion->esEntero('id', $_POST['id'])) {
265             $conmutador = $red->getConmutador($_POST['id']);
266             if ($conmutador) {
267               $log->log('Configurando Conmutador '. $_POST['id']);
268               if (!isset($_POST['confirmado']))
269                 $validacion->setErrores('confirmado', 'Marque la casilla de confirmación');
270               if (isset($_POST['default'])) {
271                 $conmutador->loadConfig($red->defaultConfig());
272               } elseif (count($validacion->getErrores()) == 0) {
273                 if ($_FILES['archivo']['error'] == UPLOAD_ERR_OK) {
274                   if (preg_match('/\.txt$/i', $_FILES['archivo']['name'])) {
275                     if ($_FILES['archivo']['size'] > 5000) {
276                       $validacion->setErrores('archivo', 'Tamao de archivo excesivo');
277                     } else {
278                       $config = array();
279                       $config['archivo'] = FILES_TEMP.'user'.sprintf('%02d',$_SESSION['
                              idusuario']).'conf00conm'.$_POST['id'].'.txt';
280                       Archivo::almacenar($_FILES['archivo']['tmp_name'], $config['archivo']);
281                       if ($validacion->esConmutador('archivo', $_SESSION['idred'], $_POST['id'
                              ], $config['archivo']))
282                         $conmutador->setConfig($config);
283                     }
284                   } else {
285                     $validacion->setErrores('archivo', 'La extension debe ser "txt"');
286                   }
287                 } else {
288                   $validacion->setErrores('archivo', 'El archivo no ha sido subido
                          correctamente');
289                 }
290               }
291
292               // Si la configuración es valida se envia al Conmutador
293               if (count($validacion->getErrores()) == 0) {
294                 $respuesta['stderr'] = $conmutador->sendConfig();
295                 if (!$respuesta['stderr']['error'])
296                   $respuesta['stdout'] = 'La configuración ha sido establecida con éxito';
297               } else {
298                 $respuesta['stderr']['error'] = 'validacion';
299                 $respuesta['stderr']['info'] = $validacion->getErrores();
300               }
301             }
302           }
303           $_SESSION['red'] = serialize($red);
```

```
304           break;
305
306
307         // Ping
308         case 'ping':
309           // Validación
310           $validacion = new Validacion;
311           if (!$validacion->esEntero('idorigen', $_POST['idorigen']) || !$validacion->esIP('
                  ipdestino', $_POST['ipdestino'])) {
312             $respuesta['stderr']['error'] = 'validacion';
313             $respuesta['stderr']['info'] = $validacion->getErrores();
314           } elseif ($validacion->esAdministracion($_POST['ipdestino'])) {
315             $respuesta['stderr']['error'] = 'validacion';
316             $respuesta['stderr']['info'] = array('ipdestino' => 'Valor no permitido');
317           } else {
318             $origen = $red->getPC($_POST['idorigen']);
319             if ($origen) {
320               $log->log('Realizando ping a la dirección '. $_POST['ipdestino']);
321               $respuesta = $origen->ping($_POST['ipdestino']);
322             } else {
323               $respuesta['stderr']['error'] = 'validacion';
324               $respuesta['stderr']['info'] = array('idorigen' => 'Origen desconocido');
325             }
326           }
327           $_SESSION['red'] = serialize($red);
328           break;
329       }
330     }
331
332     // Resultado de la operación
333     echo json_encode(array('stdout' => $respuesta['stdout'], 'stderr' => array('error' =>
           $respuesta['stderr']['error'], 'info' => $respuesta['stderr']['info'])));
334   }
335   catch (Exception $excepcion) {
336     if ($_POST['act'] == 'start')
337       Usuario::abandonaRed();
338     if (Usuario::esAdmin()) {
339       echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'excepcion', '
              info' => array((string) $excepcion))));
340     } else {
341       echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'excepcion', '
              info' => array('Se ha producido un error grave durante su petición. Pongase en
              contacto con el administrador de la aplicación.'))));
342     }
343     exit();
344   }
345 } else {
346   header('HTTP/1.1 403 Forbidden');
347 }
348 ?>
```

admin/conm_config.php

```
1  <?php
2  /**
3   * Script que devuelve el archivo de configuración indicado
4   *
5   * @author Jesús Algeciras <jes.algeciras@gmail.com>
```

```
 * @version 2010-10-24
 *
 */
session_start();
require_once('red.php');

$red = unserialize($_SESSION['red']);
$conmutador = $red->getConmutador($_GET['id']);
if ($conmutador instanceof Conmutador) {
  $file = $conmutador->getConfig();
  header('Content-Description: File Transfer');
  header('Content-Type: application/octet-stream');
  header('Content-Disposition: attachment; filename='.basename($file['archivo']));
  header('Content-Transfer-Encoding: binary');
  header('Expires: 0');
  header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
  header('Pragma: public');
  header('Content-Length: ' . filesize($file['archivo']));
  ob_clean();
  flush();
  readfile($file['archivo']);
} else {
  echo 'Archivo de configuración no accesible';
}
?>
```

admin/func_comunes.php

```
<?php
/**
 * Clases de uso común por el resto de los scripts
 *
 * @author Jesús Algeciras <jes.algeciras@gmail.com>
 * @version 2010-10-24
 *
 */

/**
 * Configuración
 */
require_once('configuracion.php');

/**
 * Decorador para la clase 'mysqli'
 *
 * Los métodos pertenecientes a 'mysqli' devuelven FALSE en caso
 * de que se produzca un error durante su ejecución. Esta clase
 * actua como decorador llamando a los métodos de la clase 'mysqli',
 * aadiendo la capacidad de lanzar excepciones del tipo DBException
 * en caso de error
 */
class Database {

  /**
   * Objeto Database
   * @var Database
   */
  private static $objeto;
```

```
31
32      /**
33       * Objeto mysqli
34       * @var mysqli
35       */
36      private $db;
37
38      /**
39       * Constructor. Obtiene la información necesaria del archivo de
40       * configuración e instancia un objeto de la clase mysqli. Es
41       * empleado el patron singleton para evitar crear una conexión
42       * distinta a la base de datos por cada objeto que quiera acceder
43       * a la misma
44       */
45      private function __construct() {
46        $this->db = @new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME, DB_PORT);
47        if (mysqli_connect_errno())
48          throw new DBException(mysqli_connect_error(), mysqli_connect_errno($this->db));
49      }
50
51      /**
52       * Se emplea el destructor para cerrar automaticamente la conexión con
53       * la base de datos
54       */
55      function __destruct() {
56        @self::$objeto->close();
57      }
58
59      /**
60       * Método estático que me devuelve la instancia de la clase
61       * @return Database
62       */
63      static function conectar() {
64        if (!self::$objeto instanceof self)
65          self::$objeto = new self;
66
67        return self::$objeto;
68      }
69
70      /**
71       * Método que permite capturar la invocación de métodos no existentes.
72       * Este método nos permitirá llamar a los métodos de la clase mysqli
73       * como si pertenecieran a la clase Database
74       * @param $metodo nombre del método que se invoca
75       * @param $argumentos matriz que contiene los argumentos pasados
76       * @return mixed
77       */
78      function __call($metodo, $argumentos) {
79        // Si es un método válido de la clase 'mysqli' lo ejecuta
80        if (method_exists($this->db, $metodo)) {
81          $resultado = call_user_func_array(array($this->db, $metodo), $argumentos);
82          if (@mysqli_errno($this->db)) {
83            throw new DBException(mysqli_error($this->db), mysqli_errno($this->db));
84          } else {
85            return $resultado;
86          }
87        }
88      }
89    }
90
91
92
```

```
93
94  /**
95   * Validación de formularios
96   *
97   * Contiene una serie de métodos que se usarán para validar
98   * si el contenido de los distintos campos de los formularios
99   * es el esperado
100  */
101 class Validacion {
102
103   /**
104    * Contiene los errores en el formulario: $errores['campo'] = 'error'
105    * @var array
106    */
107   private $errores;
108
109   function __construct() {
110     $this->errores = array();
111   }
112
113   function setErrores($input, $error) {
114     $this->errores[$input] = $error;
115   }
116
117   function getErrores() {
118     return $this->errores;
119   }
120
121   /**
122    * Comprueba si el campo esta vacio
123    * @param $input campo del formulario
124    * @param $value valor del campo
125    * @return boolean
126    */
127   function noNulo ($input, $value) {
128     if ($value == NULL) {
129       $this->errores[$input] = 'El campo esta vacio';
130       return FALSE;
131     }
132     return TRUE;
133   }
134
135   /**
136    * Comprueba si el valor del campo es 0 o 1
137    * @param $input campo del formulario
138    * @param $value valor del campo
139    * @return boolean
140    */
141   function esBinario($input, $value) {
142     if ($this->noNulo($input, $value)) {
143       if ($value == 0 || $value == 1) {
144         return TRUE;
145       } else {
146         $this->errores[$input] = 'Formato erroneo';
147         return FALSE;
148       }
149     }
150   }
151
152   /**
153    * Comprueba si el valor del campo es un entero
154    * @param $input campo del formulario
```

```
155      * @param $value valor del campo
156      * @return boolean
157      */
158     function esEntero($input, $value) {
159       if ($this->noNulo($input, $value)) {
160         if (preg_match('/^([0-9]+)$/', $value)) {
161           return TRUE;
162         } else {
163           $this->errores[$input] = 'Formato erroneo';
164           return FALSE;
165         }
166       }
167     }
168 
169     /**
170      * Comprueba si el valor del campo es una hora válida
171      * @param $input campo del formulario
172      * @param $value valor del campo
173      * @return boolean
174      */
175     function esHora($input, $value) {
176       if ($this->noNulo($input, $value)) {
177         $aux = array();
178         if (preg_match('/^([0-9]{1,2}):([0-9]{2})$/', $value, $aux) && $aux[1] < 24 && $aux
                [2] < 60) {
179           return TRUE;
180         } else {
181           $this->errores[$input] = 'Formato erroneo';
182           return FALSE;
183         }
184       }
185     }
186 
187     /**
188      * Comprueba si el valor del campo es una fecha válida
189      * @param $input campo del formulario
190      * @param $value valor del campo
191      * @return boolean
192      */
193     function esFecha($input, $value) {
194       if ($this->noNulo($input, $value)) {
195         $aux = array();
196         if (preg_match('/^([0-9]{4})\-([0-9]{1,2})\-([0-9]{1,2})$/', $value, $aux) &&
                checkdate($aux[2],$aux[3],$aux[1])) {
197           return TRUE;
198         } else {
199           $this->errores[$input] = 'Formato erroneo';
200           return FALSE;
201         }
202       }
203     }
204 
205     /**
206      * Comprueba si el valor del campo contiene caracteres no válidos
207      * @param $input campo del formulario
208      * @param $value valor del campo
209      * @return boolean
210      */
211     function esTexto($input, $value) {
212       if ($this->noNulo($input, $value)) {
213         if (preg_match('/^([a-zA-Z0-9áéíóú_]{4,})$/', $value)) {
214           return TRUE;
```

```
215         } else {
216           $this->errores[$input] = 'Formato erroneo';
217           return FALSE;
218         }
219       }
220     }
221
222     /**
223      * Comprueba si el valor del campo es una dirección de ip válida
224      * @param $input campo del formulario
225      * @param $value valor del campo
226      * @return boolean
227      */
228     function esIP($input, $value) {
229       if ($this->noNulo($input, $value)) {
230         if (preg_match('/^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.)
                {3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/', $value)) {
231           return TRUE;
232         } else {
233           $this->errores[$input] = 'Formato erroneo';
234           return FALSE;
235         }
236       }
237     }
238
239     /**
240      * Comprueba si el valor del campo es una mascara de red válida
241      * @param $input campo del formulario
242      * @param $value valor del campo
243      * @return boolean
244      */
245     function esMascara($input, $value) {
246       // Mascara en formato x.x.x.x
247       if ($this->esIP($input,$value) && preg_match('/^([1]{1,30})([0]{2,31})$/', sprintf("
              %032b",ip2long($value)))) {
248         return TRUE;
249         // Mascara en formato /xx
250       } elseif (preg_match('/(30|2[0-9]|1?[0-9])/', $value)) {
251         return TRUE;
252       } else {
253         $this->errores[$input] = 'Formato erroneo';
254         return FALSE;
255       }
256     }
257
258     /**
259      * Comprueba si la ip forma parte de la subred de administración
260      * @param $ip dirección IP
261      * @param $mascara mascara de red
262      * @return boolean
263      */
264     function esAdministracion($ip, $mascara = NULL) {
265       // Datos de la red de administración
266       global $cnxn_admin_red;
267       $fip = sprintf("%032b",ip2long($cnxn_admin_red['ip']));
268       $fmascara = preg_match_all('/1/',sprintf("%032b",ip2long($cnxn_admin_red['mascara']))
              , $bits);
269
270       $ip = sprintf("%032b",ip2long($ip));
271       // Obtenemos la mascara menor
272       if ($mascara) {
273         // Si la mascara esta en formato x.x.x.x la pasamos a un entero
```

```
274       if (preg_match('/(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.)
              {3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)/i', $mascara))
275         $mascara = preg_match_all('/1/',sprintf("%032b",ip2long($mascara)), $bits);
276       $mascara = $mascara > $fmascara ? $fmascara : $mascara;
277     } else {
278       $mascara = $fmascara;
279     }
280
281     if ($mascara != 0 && substr($ip, 0, $mascara) === substr($fip, 0, $mascara)) {
282       return TRUE;
283     } else {
284       return FALSE;
285     }
286   }
287
288   /**
289    * Comprueba si es una configuración válida de PC
290    * @param $config datos de la configuración
291    * @return boolean
292    */
293   function esPC($config) {
294     $error = FALSE;
295     $error = $error || !$this->esIP('ip', $config['ip']);
296     if ($this->esIP('mascara', $config['mascara'])) {
297       $error = $error || !$this->esMascara('mascara', $config['mascara']);
298     } else {
299       $error = TRUE;
300     }
301     $error = $error || !$this->esIP('puerta', $config['puerta']);
302     $error = $error || !$this->esIP('dns1', $config['dns1']);
303     $error = $error || !$this->esIP('dns2', $config['dns2']);
304     // Se comprueba que la puerta de enlace pertenezca a la misma red que el PC
305     if (!$error) {
306       $mascara = preg_match_all('/1/',sprintf("%032b",ip2long($config['mascara'])), $bits
              );
307       $ipb = sprintf("%032b",ip2long($config['ip']));
308       $gateway = sprintf("%032b",ip2long($config['puerta']));
309       if (!($mascara != 0 && substr($ipb, 0, $mascara) === substr($gateway, 0, $mascara))) {
310         $this->errores['puerta'] = 'No en la misma red del PC';
311         $error = TRUE;
312       }
313     }
314     // Se comprueba que las IPs no pertenecen a la red de administración
315     if (!$error) {
316       if ($this->esAdministracion($config['ip'], $config['mascara'])) {
317         $this->errores['ip'] = 'Valor no permitido';
318         $error = TRUE;
319       }
320       if ($this->esAdministracion($config['ip'])) {
321         $this->errores['puerta'] = 'Valor no permitido';
322         $error = TRUE;
323       }
324       if ($this->esAdministracion($config['dns1'])) {
325         $this->errores['dns1'] = 'Valor no permitido';
326         $error = TRUE;
327       }
328       if ($this->esAdministracion($config['dns2'])) {
329         $this->errores['dns2'] = 'Valor no permitido';
330         $error = TRUE;
331       }
332       if ($error) {
333         $log = LogFile::abrirLog();
```

```
334             $log->log('Intento de uso de valores restringidos a la red de administración',
                    LOG_WARNING);
335           }
336         }
337         return !$error;
338       }
339
340       /**
341        * Comprueba si es una configuración válida de conmutador
342        * @param $input campo del formulario
343        * @param $idred identificador de la red
344        * @param $idconmutador identificador del conmutador
345        * @param $config datos de la configuración
346        * @return boolean
347        */
348       function esConmutador($input, $idred, $idconmutador, $config) {
349         // Expresiones regulares usadas posteriormente
350         // -Listado de puertos o vlan ids
351         $listreg = '(?:(?:all|(?:(?:[0-9]+\s*)(?:-\s*[0-9]+)?))(?:\s*,(?:all|(?:(?:[0-9]+\s*)
                (?:-\s*[0-9]+)?)))*)';
352         // -Dirección IP
353         $ipreg = '(?:(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.)
                {3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?))';
354         // -Mascara de red
355         $maskreg = '(?:'.$ipreg.'|(?:\/(?:30|2[0-9]|1?[0-9])))';
356
357         // Datos de la subred de practicas
358         global $cnxn_admin_red;
359
360         // Lista de puertos prohibidos
361         $fports = $cnxn_admin_red['puertos_conm'];
362         $fports[] = 'all';
363
364         // VLAN ID prohibido
365         $fvlan[] = $cnxn_admin_red['vlan'];
366
367         // Lista de comandos prohibidos
368         $fcoms = array('aaa', 'arp-protect', 'auto-tftp', 'authorized-managers', 'banner', 'cdp
                ',
369                 'clock', 'console', 'crypto', 'dhcp', 'dhcp-relay', 'dhcp-snooping', 'fastboot',
370                 'fault-finder', 'front-panel-security', 'forward-protocol', 'helper-address',
371                 'igmp', 'monitor', 'key-chain', 'link-keepalive', 'lldp', 'lockout-mac',
372                 'logging', 'management-vlan', 'max-vlans', 'mirror-port', 'password',
373                 'port-security', 'radius-server', 'snmp-server', 'snmpv3', 'sntp', 'ssh',
374                 'stack', 'static-mac', 'tacacs-server', 'telnet-server', 'tftp', 'time',
375                 'source-binding', 'timep', 'timesync', 'web-management');
376
377
378         // Lista de comandos permitidos
379         // 'access-group', 'access-list', 'arp', 'arp-age', 'gvrp', 'hostname', 'interface', 'ip
                ',
380         // 'mac-age-time', 'qos', 'qos-passthrough-mode', 'spanning-tree', 'unknown-vlans', '
                vlan'
381
382
383         // Lista de comandos a comprobar (Puertos y VLAN Ids)
384         $ccoms = array('loop-protect' => $fports,
385                 'interface' => $fports,
386                 'spanning-tree(?:\s+instance\s+(?:1[0-6]|[1-9]|ist))?' => $fports,
387                 'trunk' => $fports,
388                 'auto' => $fports, //'vlan <VLAN-IDs> forbid'
389                        'forbid' => $fports, //'vlan <VLAN-IDs> auto'
```

```
390             'tagged' => $fports, //'vlan <VLAN-IDs> tagged'
391             'untagged' => $fports, //'vlan <VLAN-IDs> untagged'
392             'primary-vlan' => $fvlan,
393             'static' => $fvlan,
394             'vlan' => $fvlan);
395
396     // Lista de comandos a comprobar (Direcciones IP)
397     $cipcoms = array(
398         // 'qos device-priority <ip> [mask]'
399           'qos device-priority' => 'qos\s+device-priority\s+('.$ipreg.')\s*('.$maskreg.')?',
400         // ip address <ip> <mask>
401         'ip address' => 'ip\s+address\s+('.$ipreg.')\s*('.$maskreg.')',
402         // ip default-gateway <ip>
403         'ip default-gateway' => 'ip\s+default-gateway\s+('.$ipreg.')',
404         // ip route <ip> <mask> <ip>
405         'ip route' => 'ip\s+route\s+('.$ipreg.')\s*('.$maskreg.')\s+('.$ipreg.')');
406
407     $valido = TRUE;
408     $log = LogFile::abrirLog();
409
410     $texto = Archivo::leer($config);
411
412     // Se comprueba la aparición de los comandos no permitidos
413     foreach ($fcoms as $fcom) {
414       if (preg_match('/\s'.$fcom.'\s/', $texto)) {
415         $this->errores[$input][] = 'Uso del comando no permitido "'.$fcom.'"';
416         $valido = FALSE;
417       }
418     }
419
420     // Se comprueba que no se altere los puertos y VLANs no permitidas
421     foreach ($ccoms as $ccom => $ids) {
422       $aux = array();
423       $matches = array();
424       preg_match_all('/(?:\b('.$ccom.')\s+(?:(?:e|ethernet)\s+)?)('.$listreg.')\b/i',
              $texto, $matches);
425
426       foreach ($matches[2] as $match) {
427         preg_match_all('/all|(?:[0-9]+)(?:-[0-9]+)?\b/i', $match, $valores);
428         foreach ($valores[0] as $valor) {
429           if (preg_match('/([0-9]+)-([0-9]+)/', $valor, $rango)) {
430             $aux = array_merge($aux,range($rango[1], $rango[2]));
431           } else {
432             $aux[] = $valor;
433           }
434         }
435       }
436       $erroneos = array_intersect($ids, array_unique($aux));
437       if ($erroneos) {
438         $ccom = preg_replace('/spanning-tree.*/', 'spanning-tree', $ccom);
439         $this->errores[$input][] = 'Valores no permitidos para el comando "'.$ccom.'": '.
              implode(',', $erroneos);
440         $valido = FALSE;
441       }
442     }
443
444
445     // Se comprueba que no se empleen direcciones de la subred de administración
446     foreach ($cipcoms as $name => $cipcom) {
447       $aux = array();
448       $matches = array();
449       preg_match_all('/\b'.$cipcom.'\b/i', $texto, $matches, PREG_SET_ORDER);
```

```
450
451       foreach ($matches as $match) {
452         $erroneos = '';
453         $ip = $match[1];
454         $mask = $match[2] ? str_replace('/', '', $match[2]) : NULL;
455         $gate = $match[3] ? $match[3] : NULL;
456         if ($this->esAdministracion($ip, $mask))
457           $erroneos = $match[1].' '.$match[2];
458         if ($gate && $this->esAdministracion($gate))
459           $erroneos = $erroneos ? $erroneos.' '.$match[3] : $match[3];
460         if ($erroneos) {
461           $this->errores[$input][] = 'Valores no permitidos para el comando "'.$name.'": '.
                $erroneos;
462           $valido = FALSE;
463         }
464       }
465     }
466
467     if (!$valido) {
468       $log = LogFile::abrirLog();
469       $log->log('Intento de uso de valores restringidos a la red de administración',
              LOG_WARNING);
470       return FALSE;
471     } else {
472       return TRUE;
473     }
474   }
475 }
476
477
478
479
480 /**
481  * Control de acceso de los usuarios
482  *
483  * Contiene métodos para controlar el acceso de los usuarios
484  * a la página web, reserva de turnos y aplicación así como para
485  * la obtención de información acerca de estos desde la base de datos
486  */
487 class Usuario {
488
489   /**
490    * Valida los campos username y password y los comprueba en la base de datos
491    * Si son validos se inicia la sesión y se almacenan los datos que usarán
492    * posteriormente a lo largo de esta.
493    * @param $username
494    * @param $password
495    * @return boolean
496    */
497   static function login($usuario, $contrasena) {
498     $validacion = new Validacion;
499     $validacion->esTexto('usuario', $usuario);
500     $validacion->esTexto('contrasena', $contrasena);
501     if (count($validacion->getErrores())) {
502       return FALSE;
503     } else {
504       $db = Database::conectar();
505       $respuesta = $db->query('SELECT * FROM usuario NATURAL LEFT JOIN reserva
506                   WHERE usuario.usuario="'.$usuario.'" AND contrasena="'.$contrasena.'"
                          LIMIT 1');
507       if ($respuesta->num_rows > 0) {
508         $datos = $respuesta->fetch_assoc();
```

```
509         $respuesta->free();
510
511         // Se cambia el id de sesión por motivos de seguridad
512         session_regenerate_id();
513
514         $_SESSION['idusuario'] = $datos['idusuario'];
515         $_SESSION['usuario'] = $datos['usuario'];
516         $_SESSION['administrador'] = $datos['administrador'] ? TRUE : FALSE;
517
518         // Si el usuario ya tiene un turno asignado se almecena la información sobre este
519         if (isset($datos['idred']))
520           $_SESSION['idred'] = $datos['idred'];
521
522         if (isset($datos['idturno'])) {
523           $respuesta = $db->query('SELECT * FROM turno WHERE idturno='.$datos['idturno']);
524           $datos = $respuesta->fetch_assoc();
525           $respuesta->free();
526           $_SESSION['idturno'] = $datos['idturno'];
527           $_SESSION['turno_dia'] = $datos['dia'];
528           $_SESSION['turno_inicio'] = $datos['inicio'];
529           $_SESSION['turno_fin'] = $datos['fin'];
530           setcookie('turno_fin',$datos['fin'], time() + 3600*24, '/');
531         }
532         return TRUE;
533       } else {
534         $respuesta->free();
535         return FALSE;
536       }
537     }
538   }
539
540   /**
541    * Finaliza la sesión del usario y borra los datos de la misma
542    */
543   static function logout() {
544     $_SESSION = array();
545     setcookie('turno_fin','', 1, '/');
546     session_destroy();
547   }
548
549   /**
550    * Comprueba si el usuario ha iniciado la sesión
551    * @return boolean
552    */
553   static function logged() {
554     return isset($_SESSION['idusuario']);
555   }
556
557   /**
558    * Comprueba si el usuario tiene permisos de administración
559    * @return boolean
560    */
561   static function esAdmin() {
562     return (isset($_SESSION['administrador']) && $_SESSION['administrador']);
563   }
564
565   /**
566    * Comprueba que se esta dentro del periodo de reservas
567    * @return boolean
568    */
569   static function accesoReserva() {
570     $reservar = FALSE;
```

```
571       $db = Database::conectar();
572       $respuesta = $db->query('SELECT * FROM acceso WHERE idacceso="reservas"');
573       if ($respuesta->num_rows > 0) {
574         $datos = $respuesta->fetch_assoc();
575         $respuesta->free();
576         if(isset($_SESSION['idusuario']) && time() > strtotime($datos['inicio']) && time() <
              strtotime($datos['fin']))
577           $reservar = TRUE;
578       }
579       return $reservar;
580     }
581
582     /**
583      * Realiza la reserva de un turno
584      * @param $idturno
585      * @param $idred
586      * @return boolean
587      */
588     static function reservar($idturno, $idred) {
589       $validacion = new Validacion;
590
591       if (self::accesoReserva() && $validacion->esEntero('idturno', $idturno) && $validacion->
              esEntero('idred', $idred)) {
592         $db = Database::conectar();
593         // Se comprueba que el turno no este ya reservado
594         $respuesta = $db->query('SELECT * FROM reserva WHERE idturno='.$idturno.' AND idred='.
              $idred);
595         if ($respuesta->num_rows != 0) {
596           $respuesta->free();
597           return FALSE;
598         } else {
599           $respuesta->free();
600           // Se realiza una nueva reserva o se actualiza la ya realizada
601           $respuesta = $db->query('SELECT * FROM reserva WHERE idusuario='.$_SESSION['
                idusuario']);
602           if ($respuesta->num_rows == 0) {
603             $db->query('INSERT INTO reserva (idusuario,idred,idturno) VALUES ('.$_SESSION['
                  idusuario'].','.$idred.','.$idturno.')');
604           } else {
605             $db->query('UPDATE reserva SET idturno='.$idturno.', idred='.$idred.' WHERE
                  idusuario='.$_SESSION['idusuario']);
606           }
607           $respuesta->free();
608           // Se actualizan los datos de la sesión
609           $respuesta = $db->query('SELECT * FROM turno NATURAL LEFT JOIN reserva WHERE turno.
                idturno='.$idturno);
610           $datos = $respuesta->fetch_assoc();
611           $respuesta->free();
612           $_SESSION['idturno'] = $idturno;
613           $_SESSION['turno_dia'] = $datos['dia'];
614           $_SESSION['turno_inicio'] = $datos['inicio'];
615           $_SESSION['turno_fin'] = $datos['fin'];
616           $_SESSION['idred'] = $datos['idred'];
617           setcookie('turno_fin',$datos['fin'], time() + 3600*24, '/');
618           return TRUE;
619         }
620       }
621     }
622
623     /**
624      * Cancela la reserva del turno
625      */
```

```
626     static function cancelarReserva() {
627       $db = Database::conectar();
628       $db->query('DELETE FROM reserva WHERE idusuario='.$_SESSION['idusuario']);
629       unset($_SESSION['idturno']);
630       unset($_SESSION['turno_dia']);
631       unset($_SESSION['turno_inicio']);
632       unset($_SESSION['turno_fin']);
633       unset($_SESSION['idred']);
634       setcookie('turno_fin','', time() - 7200, '/');
635     }
636 
637     /**
638      * Comprueba si el usuario puede iniciar la aplicación, esto es,
639      * si este se encuentra dentro del periodo de acceso, si es su
640      * turno y si la aplicación no se encuentra ya en uso
641      * @return boolean
642      */
643     static function accesoAplicacion() {
644       $acceso = FALSE;
645 
646       if ($acceso = isset($_SESSION['idturno'])) {
647         $inicio = mktime(substr($_SESSION['turno_inicio'], 0, 2), substr($_SESSION['
              turno_inicio'], 3, 2), 0);
648         $fin = mktime(substr($_SESSION['turno_fin'], 0, 2), substr($_SESSION['turno_fin'],
              3, 2), 0);
649       }
650       // Se comprueba que el usuario halla iniciado la sesión
651       $acceso = $acceso && self::logged();
652 
653       // Se comprueba que se esté dentro de las fechas establecidas para el acceso a la
            aplicación
654       $db = Database::conectar();
655       $respuesta = $db->query('SELECT * FROM acceso WHERE idacceso="aplicacion"');
656       if ($respuesta->num_rows > 0) {
657         $fecha = $respuesta->fetch_assoc();
658         $respuesta->free();
659         $acceso = $acceso && (time() > strtotime($fecha['inicio'])) && (time() < strtotime(
              $fecha['fin']));
660       }
661       // Se comprueba que sea el turno del usuario
662       $acceso = $acceso && ($_SESSION['turno_dia'] == date('w')) && (time() > $inicio) && (
            $fin > time());
663 
664       if ($acceso) {
665         // Se comprueba que la aplicación no se encuentre actualmente ya en uso con ese id
666         // de usuario desde otra maquina.
667         $respuesta = $db->query('SELECT * FROM red WHERE idred='.$_SESSION['idred']);
668         $datos = $respuesta->fetch_assoc();
669         $respuesta->free();
670 
671         if ($datos['en_uso']) {
672           // Si el usuario activo no es el actual es a causa de que el prinero no cerró la
                aplicación correctamente.
673           if ($datos['ult_usuario'] != $_SESSION['idusuario']) {
674             $db->query('UPDATE red SET en_uso=0 WHERE idred='.$_SESSION['idred']);
675           } elseif($_SESSION['ult_acceso']) {
676             // Si el usuario activo es el actual se comprueba la fecha de ultimo acceso a la
                  red.
677             // Si es distinta a la suya es porque otro miembro del grupo ha accedido.
678             if (strtotime($datos['ult_acceso']) > strtotime($_SESSION['ult_acceso'])) {
679               $acceso = FALSE;
680               unset($_SESSION['red']);
```

```
681             unset($_SESSION['ult_acceso']);
682             foreach($_COOKIE as $name => $value)
683               setcookie($name, $value, 1, '/aplicacion/');
684           }
685         } else {
686           // Si un nuevo miembro del grupo accede se expulsa al anterior
687           $db->query('UPDATE red SET en_uso=0 WHERE idred='.$_SESSION['idred']);
688         }
689       }
690     }
691     return $acceso;
692   }
693
694   /**
695    * Actualiza el estado del usuario a activo
696    */
697   static function accedeRed() {
698     // Si se le concede el acceso a la aplicación se actualiza su estado a activo
699     $db = Database::conectar();
700     $db->query('UPDATE red SET en_uso=1,ult_acceso="'.date("Y-m-d H:i:s").'",ult_usuario='.
          $_SESSION['idusuario'].' WHERE idred='.$_SESSION['idred']);
701     $_SESSION['ult_acceso'] = date("Y-m-d H:i:s");
702   }
703
704   /**
705    * Actualiza el estado del usuario a no activo
706    */
707   static function abandonaRed() {
708     $db = Database::conectar();
709     $db->query('UPDATE red SET en_uso=0 WHERE idred='.$_SESSION['idred']);
710     unset($_SESSION['ult_acceso']);
711   }
712 }
713
714
715
716
717 /**
718  * Manejo de archivos
719  *
720  * Contiene métodos para el manejo de archivos con la capacidad
721  * de generar excepciones del tipo FileException en caso de error
722  */
723 class Archivo {
724
725   /**
726    * Devuelve el contenido del archivo leido en una cadena
727    * @param $archivo
728    * @return string
729    */
730   static function leer($archivo) {
731     $texto = '';
732     if (!($texto = @file_get_contents($archivo))) {
733       throw new FileException('Error al intentar leer el archivo "'.$archivo.'"');
734     } else {
735       return $texto;
736     }
737   }
738
739   /**
740    * Devuelve el contenido del archivo leido en un array
741    * @param $archivo
```

```
742      * @return array
743      */
744     static function lineas($archivo) {
745       $lineas = array();
746       if (!($lineas = @file($archivo))) {
747         throw new FileException('Error al intentar leer el archivo "'.$archivo.'"');
748       } else {
749         return $lineas;
750       }
751     }
752
753     /**
754      * Aade el texto pasado a la función al final
755      * del archivo indicado.
756      * @param $archivo
757      * @param $texto
758      * @return boolean
759      */
760     static function escribir($archivo, $texto) {
761       if(!($file = @fopen($archivo, 'a'))) {
762         throw new FileException('Error al intentar abrir el archivo "'.$archivo.'"');
763       } else {
764         if(!@fwrite($file, $texto))
765           throw new FileException('Error al intentar modificar el archivo "'.$archivo.'"');
766         @fclose($file);
767         return TRUE;
768       }
769       return FALSE;
770     }
771
772     /**
773      * Hace una copia de un archivo
774      * @param $origen
775      * @param $destino
776      * @return boolean
777      */
778     static function copiar($origen, $destino) {
779       if (!(@copy($origen, $destino))) {
780         throw new FileException('Error al intentar copiar el archivo "'.$origen.'" a "'.
               $destino.'"');
781       } else {
782         return TRUE;
783       }
784     }
785
786     /**
787      * Crea un archivo. Si este ya existe lo vacia.
788      * @param $archivo
789      * @return boolean
790      */
791     static function crea($archivo) {
792       if (!($file = @fopen($archivo, 'w+'))) {
793         throw new FileException('Error al intentar crear el archivo "'.$archivo.'"');
794       } else {
795         @fclose($file);
796       }
797       return TRUE;
798     }
799
800     /**
801      * Borra una archivo
802      * @param $archivo
```

```
803       * @return boolean
804       */
805      static function borrar($archivo) {
806        if (file_exists($archivo)) {
807          if (!(@unlink($archivo)))
808            throw new FileException('Error al intentar borrar el archivo "'.$archivo.'"');
809        }
810        return TRUE;
811      }
812
813      /**
814       * Copia un archivo subido a un directorio permanente
815       * @param $archivo
816       * @param $destino
817       * @return boolean
818       */
819      static function almacenar($archivo, $destino) {
820        if (!(@move_uploaded_file($archivo, $destino))) {
821          throw new FileException('Error al intentar almacenar el archivo "'.$archivo.'" a la
                 carpeta "'.$destino.'"');
822        } else {
823          return TRUE;
824        }
825      }
826    }
827
828
829    /**
830     * Log de la aplicación
831     *
832     * Clase que se emplea para el registro de eventos en el
833     * archivo de log
834     */
835    class LogFile {
836
837      /**
838       * Objeto LogFile
839       * @var LogFile
840       */
841      private static $objeto;
842
843      /**
844       * El archivo abierto
845       * @var mixed
846       */
847      protected $archivo;
848
849      /**
850       * Estado del archivo
851       * @var boolean
852       */
853      protected $abierto = FALSE;
854
855      /**
856       * Nivel de verbose
857       * @var int
858       */
859      protected $verbose;
860
861      /**
862       * Constructor. Obtiene el nombre del archivo de log
863       * del archivo de configuración
```

```
864      */
865     private function __construct($verbose) {
866       if (!file_exists(FILES_LOG)) {
867         if (!@touch(FILES_LOG))
868           throw new Exception('No ha sido posible la creación del archivo de log: "'.FILES_LOG
                .'"');
869         if (!@chmod(FILES_LOG, 0664))
870           throw new Exception('Error al intentar cambiar los permisos del archivo de log: "'.
                FILES_LOG.'"');
871       }
872
873       if (!$this->abierto) {
874         if (!($this->archivo = @fopen(FILES_LOG, 'a')))
875           throw new Exception('Error al intentar abrir el archivo "'.FILES_LOG.'"');
876         $this->abierto = TRUE;
877       }
878
879       $this->verbose = $verbose;
880     }
881
882     function __destruct(){
883       // Cierra el archivo si aun permanece abierto
884       if ($this->abierto)
885         $this->cerrarLog();
886     }
887
888     /**
889      * Método estático que me devuelve la instancia de la clase
890      * en caso de poder abrir correctamente el archivo de log
891      * @return LogFile
892      */
893     static function abrirLog($verbose = LOG_INFO) {
894       if (!self::$objeto instanceof self)
895         self::$objeto = new self($verbose);
896
897       return self::$objeto;
898     }
899
900     /**
901      * Cierra el archivo apuntado por 'archivo'
902      * @return boolean
903      */
904     function cerrarLog() {
905       if ($this->abierto)
906         @fclose($this->archivo);
907       $this->abierto = FALSE;
908       return ($this->abierto == FALSE);
909     }
910
911     /**
912      * Crea una entrada en el archivo de log
913      * @param string $mensaje
914      * @param string $prioridad Valores validos: LOG_EMERG, LOG_ALERT,
915      *        LOG_CRIT, LOG_ERR, LOG_WARNING,
916      *        LOG_NOTICE, LOG_INFO, LOG_DEBUG
917      * @return boolean
918      */
919     function log($mensaje, $prioridad = LOG_INFO) {
920       if ($prioridad <= $this->verbose) {
921         $tiempo = strftime('%Y-%m-%d %H:%M:%S');
922         $usuario = $_SESSION['usuario'].' (ID: '.$_SESSION['idusuario'].')';
923         $info = array(
```

```
924           0 => 'emergency',
925           1 => 'alert',
926           2 => 'critical',
927           3 => 'error',
928           4 => 'warning',
929           5 => 'notice',
930           6 => 'info',
931           7 => 'debug'
932         );
933         $linea = '['.$tiempo.'] ['.$usuario.'] ['.$info[$prioridad].'] '.$mensaje."\n";
934
935         flock($this->archivo, LOCK_EX);
936         if (!(@fwrite($this->archivo, $linea))) {
937           flock($this->archivo, LOCK_UN);
938           throw new Exception('Error al intentar modificar el archivo "'.$this->path.'"');
939         }
940         flock($this->archivo, LOCK_UN);
941       }
942       return TRUE;
943     }
944   }
945
946   /**
947    * Excepción que quedará resgistrada en el log
948    *
949    * Las excepciones que hereden de este clase serán
950    * almecenadas en un archivo de log
951    */
952   class LoggedException extends Exception {
953
954     function __construct($message = null, $code = 0) {
955       try {
956         $log = LogFile::abrirLog();
957         $log->log(get_class($this).': '.$message, LOG_ERR);
958       }
959       catch (Exception $excepcion) {
960         $message = $message."(No ha podido registrarse en el log)";
961       }
962       parent::__construct($message, $code);
963     }
964
965     /**
966      * Reperesentación de la excepción como cadena de texto
967      */
968     public function __toString() {
969       return get_class($this).': '.$this->message;
970     }
971   }
972
973
974   /**
975    * Excepción realacionada con la base de datos
976    *
977    * Excepción lanzada cuando se ha producido un error en la conexión
978    * o comunicación con la base de datos
979    */
980   class DBException extends LoggedException {}
981
982
983   /**
984    * Excepción realacionada con el manejo de archivos
985    *
```

```
986  * Excepción lanzada cuando se ha producido un error en el acceso,
987  * copia o modificación de un archivo
988  */
989 class FileException extends LoggedException {}
990
991
992 /**
993  * Excepción ralacionada con la conexión con nodos de la red
994  *
995  * Excepción lanzada cuando se ha producido un error al intentar
996  * cambiar la configuración de algun nodo de la red
997  */
998 class NetworkException extends LoggedException {}
```

admin/instala_db.php

```
1  <?php
2  /**
3   * Script de instalación de la base de la base de datos
4   *
5   * Este script crea las tablas necesarias en la base de datos e introduce los
6   * siguientes valores necesarios para el funcionamiento de la aplicación:
7   *
8   * - Un usuario con derechos de administración. Por simplicidad
9   *   se utilizan los mismos nombre de usuario y contrasea usados
10  *   para la base de datos. Estos valores podran ser editados
11  *   posteriormente desde la pagina web.
12  *
13  * - Los ids de las redes junto a su configuración por defecto.
14  *
15  * Tanto el usuario empleado para la conexión (y especificado en el archivo
16  * "configuracion.php" como la base de datos deben haber sido creadas
17  * previamente desde MySQL usando los siguientes comandos:
18  *
19  * >CREATE DATABASE redes;
20  * >GRANT ALL ON redes.* TO 'user'@'localhost' IDENTIFIED BY 'password';
21  *
22  * @author Jesús Algeciras <jes.algeciras@gmail.com>
23  * @version 2010-10-24
24  *
25  */
26
27 require_once('func_comunes.php');
28
29 try {
30   /**
31    * Funcion que retorna el último id introducido en un campo
32    * AUTO_INCREMENT de la tabla.
33    * @return int
34    */
35   function lastID () {
36     $database = Database::conectar();
37     $respuesta = $database->query('SELECT LAST_INSERT_ID() AS id');
38     $datos = $respuesta->fetch_assoc();
39     $respuesta->free();
40     return $datos['id'];
41   }
42
```

```
43      $database = Database::conectar();
44      $database->autocommit(FALSE);
45
46
47      // En caso de reinstalación se vacia primero la base de datos
48      $sql = 'DROP TABLE IF EXISTS acceso,reserva,turno,red,conmutador,pc,configuracion,usuario
            ';
49      $database->query($sql);
50
51
52      // Creación de las tablas
53      $sql = 'CREATE TABLE usuario (
54          idusuario  tinyint UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
55          usuario   varchar(10) NOT NULL UNIQUE,
56          contrasena  varchar(10) NOT NULL,
57          administrador boolean NOT NULL DEFAULT 0,
58
59          PRIMARY KEY (idusuario),
60          UNIQUE INDEX (usuario,contrasena)
61          )ENGINE=InnoDB;';
62      $database->query($sql);
63
64      $sql = 'CREATE TABLE configuracion (
65          idconfiguracion smallint UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
66          usuario   tinyint UNSIGNED NOT NULL,
67          nombre   varchar(20),
68          modificado  datetime NOT NULL,
69
70          PRIMARY KEY (idconfiguracion),
71          FOREIGN KEY (usuario) REFERENCES usuario (idusuario) ON DELETE CASCADE ON UPDATE
                CASCADE
72          )ENGINE=InnoDB;';
73      $database->query($sql);
74
75      $sql = 'CREATE TABLE pc (
76          idconfiguracion smallint UNSIGNED NOT NULL,
77          idpc   tinyint UNSIGNED NOT NULL,
78          ip    int UNSIGNED,
79          mascara   int UNSIGNED,
80          puerta   int UNSIGNED,
81          dns1   int UNSIGNED,
82          dns2   int UNSIGNED,
83
84          PRIMARY KEY (idconfiguracion,idpc),
85          FOREIGN KEY (idconfiguracion) REFERENCES configuracion (idconfiguracion) ON DELETE
                CASCADE ON UPDATE CASCADE
86          )ENGINE=InnoDB;';
87      $database->query($sql);
88
89      $sql = 'CREATE TABLE conmutador (
90          idconfiguracion smallint UNSIGNED NOT NULL,
91          idconmutador tinyint UNSIGNED NOT NULL,
92          archivo   varchar(255),
93
94          PRIMARY KEY (idconfiguracion,idconmutador),
95          FOREIGN KEY (idconfiguracion) REFERENCES configuracion (idconfiguracion) ON DELETE
                CASCADE ON UPDATE CASCADE
96          )ENGINE=InnoDB;';
97      $database->query($sql);
98
99      $sql = 'CREATE TABLE red (
100         idred   tinyint UNSIGNED NOT NULL UNIQUE,
```

```
101         default_config smallint UNSIGNED,
102         en_uso   boolean NOT NULL DEFAULT 0,
103         ult_acceso  datetime,
104         ult_usuario  tinyint UNSIGNED,
105 
106         PRIMARY KEY (idred),
107         FOREIGN KEY (default_config) REFERENCES configuracion (idconfiguracion) ON DELETE SET
                NULL ON UPDATE CASCADE,
108         FOREIGN KEY (ult_usuario) REFERENCES usuario (idusuario) ON DELETE SET NULL ON UPDATE
                CASCADE
109         )ENGINE=InnoDB;';
110     $database->query($sql);
111 
112     $sql = 'CREATE TABLE turno (
113         idturno  tinyint UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
114         dia   tinyint UNSIGNED NOT NULL,
115         inicio  time NOT NULL,
116         fin   time NOT NULL,
117 
118         PRIMARY KEY (idturno)
119         )ENGINE=InnoDB;';
120     $database->query($sql);
121 
122     $sql = 'CREATE TABLE reserva (
123         idusuario tinyint UNSIGNED NOT NULL UNIQUE,
124         idred  tinyint UNSIGNED NOT NULL,
125         idturno  tinyint UNSIGNED NOT NULL,
126 
127         PRIMARY KEY (idusuario),
128         FOREIGN KEY (idusuario) REFERENCES usuario (idusuario) ON DELETE CASCADE ON UPDATE
                CASCADE,
129         FOREIGN KEY (idred) REFERENCES red (idred) ON DELETE CASCADE ON UPDATE CASCADE,
130         FOREIGN KEY (idturno) REFERENCES turno (idturno) ON DELETE CASCADE ON UPDATE CASCADE,
131         UNIQUE INDEX (idred,idturno)
132         )ENGINE=InnoDB;';
133     $database->query($sql);
134 
135     $sql = 'CREATE TABLE acceso (
136         idacceso varchar(10) NOT NULL UNIQUE,
137         inicio  datetime NOT NULL,
138         fin   datetime NOT NULL,
139 
140         PRIMARY KEY (idacceso)
141         )ENGINE=InnoDB;';
142     $database->query($sql);
143 
144 
145     // Almacenamiento de los valores requeridos por la aplicación
146     $sql = 'INSERT INTO usuario (usuario,contrasena,administrador)
147          VALUES ("'.DB_USER.'","'.DB_PASS.'",1)';
148     $database->query($sql);
149     $idusuario = lastID();
150 
151 
152     // Configuración de las subredes
153     foreach ($defult_config as $idred => $red) {
154       // Creo la configuración
155       $sql = 'INSERT INTO configuracion (usuario,nombre,modificado)
156            VALUES ('.$idusuario.',"default_red_'.$idred.'","'.date('Y-m-d H:i:s').'")';
157       $database->query($sql);
158       $idconfig = lastID();
159 
```

```
160     // Guardo la configuración de PCs y Conmtadores
161     foreach ($defult_config[$idred]['pc'] as $pc => $config) {
162       $sql = 'INSERT INTO pc (idconfiguracion,idpc,ip,mascara,puerta,dns1,dns2)
163             VALUES ('.$idconfig.','.$pc.',INET_ATON("'.$config['ip'].'"),INET_ATON("'.
                    $config['mascara'].'"),
164             INET_ATON("'.$config['puerta'].'"),INET_ATON("'.$config['dns1'].'"),INET_ATON("'
                    .$config['dns2'].'"))';
165       $database->query($sql);
166     }
167     foreach ($defult_config[$idred]['conmutador'] as $conmutador => $config) {
168       $sql = 'INSERT INTO conmutador (idconfiguracion,idconmutador,archivo)
169             VALUES ('.$idconfig.','.$conmutador.',"'.$config['archivo'].'")';
170       $database->query($sql);
171     }
172     // Aado la red y sus valores a la base de datos
173     $sql = 'INSERT INTO red (idred,default_config,en_uso) VALUES('.$idred.','.$idconfig.',0)
            ';
174     $database->query($sql);
175   }
176
177   $database->commit();
178   $database->autocommit(TRUE);
179 }
180 catch (Exception $excepcion) {
181   echo (string) $excepcion;
182   exit();
183 }
```

admin/log.php

```
1  <?php
2  /**
3   * Script que devuelve el archivo de log
4   *
5   * @author Jesús Algeciras <jes.algeciras@gmail.com>
6   * @version 2010-10-24
7   *
8   */
9  session_start();
10 require_once('func_comunes.php');
11
12 if (Usuario::esAdmin()) {
13   if (file_exists(FILES_LOG)) {
14     header('Content-Description: File Transfer');
15     header('Content-Type: application/octet-stream');
16     header('Content-Disposition: attachment; filename=log.txt');
17     header('Content-Transfer-Encoding: binary');
18     header('Expires: 0');
19     header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
20     header('Pragma: public');
21     header('Content-Length: ' . filesize(FILES_LOG));
22     ob_clean();
23     flush();
24     readfile(FILES_LOG);
25   } else {
26     echo 'El archivo de log "'.FILES_LOG.'" no existe';
27   }
28 }
```

```
29 ?>
```

admin/proc_form.php

```
1  <?php
2  /**
3   * Script que procesa los formularios
4   *
5   * Para la comunicación entre el cliente y este script
6   * se emplea AJAX utilizando JSON como notación para la
7   * transferencia de información al cliente
8   *
9   * @author Jesús Algeciras <jes.algeciras@gmail.com>
10  * @version 2010-10-24
11  *
12  */
13
14 if (isset($_SERVER['HTTP_X_REQUESTED_WITH']) && $_SERVER['HTTP_X_REQUESTED_WITH'] == '
       XMLHttpRequest') {
15
16   session_start();
17   require_once('administrar.php');
18
19   // Se lleva a cabo la petición correspondiente y se devuelve el resultado en formato JSON
20   // La estructura de la respuesta es siempre la misma:
21   // array( stdout => (str), stderr => array(error => (str), info => (str)))
22   // Esta respuesta es tratada segun corresponda en el navegador mediante Javascript
23   if (Usuario::esAdmin()) {
24     if (($_POST['form'] == 'acceso') || ($_POST['form'] == 'turnos') || ($_POST['form'] == '
           usuarios') || ($_POST['form'] == 'reservas')) {
25       try {
26         $errores = array();
27         switch ($_POST['form']) {
28           case 'acceso':
29             $administrar = new AdministrarAcceso;
30             break;
31           case 'turnos':
32             $administrar = new AdministrarTurnos;
33             break;
34           case 'usuarios':
35             $administrar = new AdministrarUsuarios;
36             break;
37           case 'reservas':
38             $administrar = new AdministrarReservas;
39             break;
40         }
41         // Procesa la acción indicada en la variable $_POST['act']
42         if ($administrar != NULL) {
43           switch ($_POST['act']) {
44             case 'add':
45               $errores = $administrar->add($_POST);
46               break;
47             case 'edit':
48               $errores = $administrar->edit($_POST);
49               break;
50             case 'del':
51               $errores = $administrar->del($_POST);
```

```
52              break;
53            case 'delall':
54              $errores = $administrar->delall($_POST);
55              break;
56          }
57        }
58
59        // Devuelve la información de la ejecución al cliente
60        if (count($errores)) {
61          echo json_encode(array('stdout' => 'ERROR: Se ha producido un error durante su
                petición', 'stderr' => array('error' => 'validacion', 'info' => $errores)));
62        } else {
63          echo json_encode(array('stdout' => 'La base de datos se ha actualizado con éxito',
                'stderr' => array('error' => FALSE, 'info' => array())));
64        }
65      }
66      catch (Exception $excepcion) {
67        if (Usuario::esAdmin()) {
68          echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'excepcion',
                'info' => array((string) $excepcion))));
69        } else {
70          echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'excepcion',
                'info' => array('Se ha producido un error durante su petición'))));
71        }
72        exit();
73      }
74    }
75  } else {
76    $log = LogFile::abrirLog();
77    $log->log('Intento de acceso a la administración sin los permisos necesarios desde '.
          $_SERVER['REMOTE_ADDR'], LOG_WARNING);
78  }
79  } else {
80    header('HTTP/1.1 403 Forbidden');
81  }
82  ?>
```

admin/red.php

```
1   <?php
2   /**
3    * Clases que definen la red y los elementos que la componen
4    *
5    * @author Jesús Algeciras <jes.algeciras@gmail.com>
6    * @version 2010-10-24
7    *
8    */
9   require_once('func_comunes.php');
10  // Para la comunicación con los PCs
11  require_once('xmlrpc.inc');
12  // Para la comunicación con los Conmutadores
13  require_once ('PHPTelnet.php');
14
15  /**
16   * Clase para los elementos que conforman la red
17   *
18   * Clase abstracta que define los métodos y las propiedades
19   * de los elemnetos de la red (PC y Conmutadores).
```

```
20    */
21   abstract class ElementoRed {
22
23     /**
24      * Identificador
25      * @var int
26      */
27     protected $id;
28
29     /**
30      * Dirección IP de administración
31      * @var string
32      */
33     protected $ip;
34
35     /**
36      * Configuración actual
37      * @var mixed
38      */
39     protected $config;
40
41     /**
42      * Ultima configuración valida
43      * @var mixed
44      */
45     protected $lastConfig;
46
47     function __construct($id, $ip, $config = NULL) {
48       $this->id = $id;
49       $this->ip = $ip;
50       $this->config = $config;
51       $this->lastConfig = $config;
52     }
53
54     function getID() {
55       return $this->id;
56     }
57
58     function setConfig($config) {
59       $this->config = $config;
60     }
61
62     function getConfig() {
63       return $this->config;
64     }
65
66     /**
67      * Envia la configuración al PC/Conmutador
68      * @return array información sobre errores
69      */
70     abstract function sendConfig();
71
72     /**
73      * Carga la configuración de la base de datos
74      * @param $idconfig id de la configuración
75      * @return boolean
76      */
77     abstract function loadConfig($idconfig);
78
79     /**
80      * Almacena la configuración en la base de datos
81      * @param $idconfig id de la configuración
```

```
82      * @return boolean
83      */
84     abstract function storeConfig($idconfig);
85   }
86
87   /**
88    * Clase que define propiedades y métodos para el PC
89    */
90   class PC extends ElementoRed {
91
92     /**
93      * Puerto de administración
94      * @var int
95      */
96     protected $port;
97
98     function __construct($id, $ip, $port, $config = NULL) {
99       $this->port = $port;
100      parent::__construct($id, $ip, $config = NULL);
101    }
102
103    function sendConfig() {
104      $error['error'] = FALSE;
105      $error['info'] = array();
106
107      // Construimos el mensaje XML-RPC y lo enviamos
108      $params = array(new xmlrpcval($this->config['ip']),
109      new xmlrpcval($this->config['mascara']),
110      new xmlrpcval($this->config['puerta']),
111      new xmlrpcval($this->config['dns1']),
112      new xmlrpcval($this->config['dns2']));
113
114      $msg = new xmlrpcmsg('config_red',$params);
115      $cliente = new xmlrpc_client('/', $this->ip, $this->port);
116      $respuesta = $cliente->send($msg);
117
118      if ($respuesta) {
119        if (!$respuesta->faultcode()) {
120          $resp = php_xmlrpc_decode($respuesta->value());
121          if (strlen($resp[1][0]) != 0)
122            throw new NetworkException('(PC '.$this->id.') '.implode($resp[1]));
123        } else {
124          throw new NetworkException('(PC '.$this->id.') '.$respuesta->faultString());
125        }
126        if ($error['error']) {
127          $this->config = $this->lastConfig;
128        } else {
129          $this->lastConfig = $this->config;
130        }
131      } else {
132        throw new NetworkException('(PC '.$this->id.') No ha sido posible establecer la conexi
             ón');
133      }
134      return $error;
135    }
136
137    function loadConfig($idconfig) {
138      $db = Database::conectar();
139      $respuesta = $db->query('SELECT INET_NTOA(ip) AS ip,
140                   INET_NTOA(mascara) AS mascara,
141                   INET_NTOA(puerta) AS puerta,
142                   INET_NTOA(dns1) AS dns1,
```

```
143                 INET_NTOA(dns2) AS dns2 FROM pc
144                 WHERE idconfiguracion="'.$idconfig.'" AND idpc='.$this->id);
145       if ($respuesta->num_rows > 0) {
146         $datos = $respuesta->fetch_assoc();
147         $respuesta->free();
148         $this->config = array('ip' => $datos['ip'],
149                 'mascara' => $datos['mascara'],
150                 'puerta' => $datos['puerta'],
151                 'dns1' => $datos['dns1'],
152                 'dns2' => $datos['dns2']);
153         return TRUE;
154       } else {
155         $respuesta->free();
156         return FALSE;
157       }
158     }
159
160     function storeConfig($idconfig) {
161       $db = Database::conectar();
162       if ($this->config) {
163         $respuesta = $db->query('SELECT idpc FROM configuracion NATURAL LEFT JOIN pc
164                 WHERE usuario='.$_SESSION["idusuario"].'
165                 AND idconfiguracion='.$idconfig.' AND idpc='.$this->id);
166         if ($respuesta->num_rows > 0) {
167           $db->query('UPDATE pc SET ip=INET_ATON("'.$this->config["ip"].'"),
168                 mascara=INET_ATON("'.$this->config["mascara"].'"),
169                 puerta=INET_ATON("'.$this->config["puerta"].'"),
170                 dns1=INET_ATON("'.$this->config["dns1"].'"),
171                 dns2=INET_ATON("'.$this->config["dns2"].'")
172                 WHERE idconfiguracion='.$idconfig.' AND idpc='.$this->id);
173         } elseif (!$this->default) {
174           $db->query('INSERT INTO pc VALUES ('.$idconfig.','.$this->id.',
175                 INET_ATON("'.$this->config["ip"].'"),
176                 INET_ATON("'.$this->config["mascara"].'"),
177                 INET_ATON("'.$this->config["puerta"].'"),
178                 INET_ATON("'.$this->config["dns1"].'"),
179                 INET_ATON("'.$this->config["dns2"].'"))');
180         }
181         $respuesta->free();
182       }
183       return TRUE;
184     }
185
186     /**
187      * Petición al servidor XML-RPC de un ping a la dirección indicada
188      * @param $destino ip destino del ping
189      * @return array
190      */
191     function ping($destino) {
192       $respuesta['stdout'] = array();
193       $respuesta['stderr']['error'] = FALSE;
194       $respuesta['stderr']['info'] = array();
195
196       // Construimos el mensaje XML-RPC y lo enviamos
197       $params = array(new xmlrpcval($destino));
198       $msg = new xmlrpcmsg('ping',$params);
199       $cliente = new xmlrpc_client('/', $this->ip, $this->port);
200       $resp = $cliente->send($msg);
201
202       // Se procesa la respuesta del servidor situado en el PC
203       if ($resp) {
204         if ($resp->faultcode() == 0) {
```

```
205         $resp = php_xmlrpc_decode($resp->value());
206         $respuesta['stdout'] = $resp[0];
207         if (strlen($resp[1][0]) != 0)
208           throw new NetworkException('(PC '.$this->id.') '.implode($resp[1]));
209       } else {
210         throw new NetworkException('(PC '.$this->id.') '.$respuesta->faultString());
211       }
212     } else {
213       throw new NetworkException('(PC '.$this->id.') No ha sido posible establecer la conexi
              ón');
214     }
215
216     return $respuesta;
217   }
218 }
219
220 /**
221  * Clase que define propiedades y métodos para el Conmutador
222  */
223 class Conmutador extends ElementoRed {
224
225   /**
226    * Contrasea
227    * @var string
228    */
229   protected $password;
230
231   function __construct($id, $ip, $password, $config = NULL) {
232     $this->password = $password;
233     parent::__construct($id, $ip, $config = NULL);
234   }
235
236   function sendConfig() {
237     $error['error'] = FALSE;
238     $error['info'] = array();
239     global $cnxn_admin_red;
240
241     $telnet = new PHPTelnet();
242     $telnet->show_connect_error = 0;
243
244     $respuesta = $telnet->Connect($this->ip, '', $this->password);
245     switch ($respuesta) {
246       case 0:
247         $this->generateConfig();
248         $telnet->DoCommand('copy tftp startup-config '.$cnxn_admin_red['ip'].' conm_config.
              txt', $respuesta);
249         $telnet->DoCommand('y', $respuesta);
250
251         // Se eliminan los caracteres de la secuencia de escape ASCII de la respuesta
252         $respuesta = preg_replace('/^.*24;1H([^\x1b]*)\r.*$/', "$1", $respuesta);
253         $respuesta = preg_replace('/\r/', '', $respuesta);
254         if (preg_match('/Corrupted download file/', $respuesta)) {
255           // Obtenermos los errores en la configuración del archivo de log del conmutador
256           $telnet->DoCommand('copy command-output "log -r" tftp '.$cnxn_admin_red['ip'].'
                log unix', $respuesta);
257           $conmlog = Archivo::lineas(FILES_TFTP.'log');
258           $warnings = array();
259           // Se obtienen todas las líneas desde la última transferencia
260           foreach ($conmlog as $line) {
261             if (preg_match('/Transfer completed/', $line))
262               break;
263             if (preg_match('/^W /', $line))
```

```
264               $warnings[] = $line;
265             }
266             $warnings = array_reverse($warnings);
267             foreach ($warnings as $i => $warn) {
268               $warn = preg_replace('/^.*update: (.*)$/', "$1", $warn);
269               $warn = preg_replace('/^line (\d+). /', '', $warn);
270               $warn = preg_replace('/Invalid input/', 'Comando / Valor erroneo', $warn);
271               $warnings[$i] = $warn;
272             }
273             Archivo::borrar(FILES_TFTP.'log');
274             $error['error'] = 'validacion';
275             $error['info']['archivo'] = $warnings;
276           }
277           $telnet->Disconnect();
278           Archivo::borrar(FILES_TFTP."conm_config.txt");
279           break;
280
281         case 1:
282           throw new NetworkException('(Conmutador '.$this->id.') No ha sido posible establecer
                la conexión');
283           break;
284         case 2:
285           throw new NetworkException('(Conmutador '.$this->id.') Host desconocido');
286           break;
287         case 3:
288           throw new NetworkException('(Conmutador '.$this->id.') Contrasea invalida');
289           break;
290         case 4:
291           throw new NetworkException('La versión de PHP no soporta la clase PHPTelnet');
292           break;
293       }
294
295       if ($error['error']) {
296         $this->config = $this->lastConfig;
297       } else {
298         $this->lastConfig = $this->config;
299       }
300       return $error;
301     }
302
303     function loadConfig($idconfig) {
304       $db = Database::conectar();
305       $respuesta = $db->query('SELECT archivo FROM conmutador WHERE idconfiguracion='.
              $idconfig.' AND idconmutador='.$this->id);
306       if ($respuesta->num_rows > 0) {
307         $datos = $respuesta->fetch_assoc();
308         $file = $datos['archivo'];
309         $respuesta->free();
310         if (is_file($file)) {
311           $this->config['archivo'] = $file;
312           return TRUE;
313         } else {
314           return FALSE;
315         }
316       } else {
317         $respuesta->free();
318         return FALSE;
319       }
320     }
321
322     function storeConfig($idconfig) {
323       $db = Database::conectar();
```

```
324       if ($this->config['archivo']) {
325         $origen = $this->config['archivo'];
326         $destino = FILES_STORAGE.'user'.sprintf('%02d',$_SESSION['idusuario']).'conf'.sprintf(
                '%d', $idconfig).'conm'.$this->id.'.txt';
327         if ($origen != $destino) {
328           Archivo::copiar($origen, $destino);
329           $respuesta = $db->query('SELECT idconmutador FROM configuracion
330                       NATURAL LEFT JOIN conmutador
331                       WHERE usuario='.$_SESSION["idusuario"].'
332                       AND idconfiguracion='.$idconfig.'
333                       AND idconmutador='.$this->id);
334           if ($respuesta->num_rows > 0) {
335             $datos = $respuesta->fetch_assoc();
336             $db->query('UPDATE conmutador SET archivo="'.$destino.'"
337                   WHERE idconfiguracion='.$idconfig.'
338                   AND idconmutador='.$this->id);
339           } else {
340             $db->query('INSERT INTO conmutador VALUES ('.$idconfig.','.$this->id.',"'.$destino
                      .'")');
341           }
342           $respuesta->free();
343           $this->config['archivo'] = $destino;
344         }
345       }
346       return TRUE;
347     }
348
349     /**
350      * Aade al archivo de configuración creado por el usuario
351      * la configuración de administración
352      * @return string ruta del archivo de configuración
353      */
354     private function generateConfig() {
355       global $cnxn_red;
356       $file = FILES_TFTP."conm_config.txt";
357       Archivo::crea($file);
358       Archivo::escribir($file, Archivo::leer($cnxn_red[$_SESSION['idred']]['conmutador'][$this
              ->id]['admin_config']));
359       Archivo::escribir($file, Archivo::leer($this->config['archivo']));
360     }
361   }
362
363   /**
364    * Clase que define al conjunto de toda la red
365    *
366    * Esta clase contendrá los objetos que representan los distintos elementos
367    * que conforman la red y los metodos para tratar con ellos de manera general.
368    */
369   class Red {
370     /**
371      * Objetos de la clase PC
372      * @var array
373      */
374     protected $pc;
375
376     /**
377      * Objetos de la clase Conmutador
378      * @var array
379      */
380     protected $conmutador;
381
382     /**
```

```
383      * Aade elementos a la red
384      * @param $elementoRed objeto a aadir
385      * @return boolean
386      */
387     function add($elementoRed) {
388       if ($elementoRed instanceof PC) {
389         $this->pc[] = $elementoRed;
390       } elseif ($elementoRed instanceof Conmutador) {
391         $this->conmutador[] = $elementoRed;
392       } else {
393         return FALSE;
394       }
395       return TRUE;
396     }
397
398     /**
399      * Devuelve la instancia de la clase PC indicada o todas
400      * @param $id id del elemento
401      * @return mixed
402      */
403     function getPC($id = NULL) {
404       if ($id) {
405         foreach ($this->pc as $pc) {
406           if ($pc->getID() == $id)
407             return $pc;
408         }
409       } else {
410         return $this->pc;
411       }
412       return FALSE;
413     }
414
415     /**
416      * Devuelve la instancia de la clase Conmutador indicada o todas
417      * @param $id id del elemento
418      * @return mixed
419      */
420     function getConmutador($id = NULL) {
421       if ($id) {
422         foreach ($this->conmutador as $conmutador) {
423           if ($conmutador->getID() == $id)
424             return $conmutador;
425         }
426       } else {
427         return $this->conmutador;
428       }
429       return FALSE;
430     }
431
432     /**
433      * Devuelve ID de la configuración por defecto de la red
434      * @return int ID de configuración
435      */
436     function defaultConfig() {
437       $database = Database::conectar();
438       $dbdata = $database->query('SELECT default_config FROM red WHERE idred='.$_SESSION['
              idred']);
439       $data = $dbdata->fetch_assoc();
440       $dbdata->free();
441       return $data['default_config'];
442     }
443
```

```
444     /**
445      * Invoca a la función sendConfig de todos los elementos de la red
446      * @return array información sobre errores
447      */
448     function sendConfig() {
449       $error['error'] = FALSE;
450       $error['info'] = array();
451       $nodos = array_merge($this->pc, $this->conmutador);
452       foreach ($nodos as $elementoRed) {
453         $errores == $elementoRed->sendConfig();
454         if ($errores['error']) {
455           $error['error'] = 'aplicacion';
456           $error['info'] = array_merge($error['info'], $errorres['info']);
457         }
458       }
459       return $error;
460     }
461 
462     /**
463      * Invoca a la función loadConfig de todos los elementos de la red
464      * @param $idconfig ID de de la confuración o 'default'
465      * @return array información sobre errores
466      */
467     function loadConfig($idconfig) {
468       $error['error'] = FALSE;
469       $error['info'] = array();
470       $loaded = TRUE;
471       $nodos = array_merge($this->pc, $this->conmutador);
472       foreach ($nodos as $elementoRed)
473         $loaded = $loaded && $elementoRed->loadConfig($idconfig);
474       if (!$loaded) {
475         $error['error'] = 'aplicacion';
476         $error['info'] = array('ERROR: La configuración no ha podido ser cargada con exito');
477       }
478       return $error;
479     }
480 
481     /**
482      * Almacena los datos de la configuracion e invoca a la
483      * función storeConfig de todos los elementos de la red
484      * @param $nombre Nombre de de la confuración
485      * @param $idconfig ID de de la confuración
486      * @return array información sobre errores
487      */
488     function storeConfig($idconfig, $nombre) {
489       $error['error'] = FALSE;
490       $error['info'] = array();
491       $nueva = ($idconfig == 'new') ? TRUE : FALSE;
492       $db = Database::conectar();
493       if (!$nueva) {
494         $db->autocommit(FALSE);
495         $db->query('UPDATE configuracion SET modificado="'.date('Y-m-d H:i:s').'",
496                nombre="'.$nombre.'" WHERE idconfiguracion='.$idconfig);
497       } else {
498         $respuesta = $db->query('SELECT COUNT(usuario) FROM configuracion
499                     WHERE usuario='.$_SESSION['idusuario'].' GROUP BY usuario');
500         $datos = $respuesta->fetch_assoc();
501         $respuesta->free();
502         if ($datos['COUNT(usuario)'] >= 20) {
503           $error['error'] = 'aplicacion';
504           $error['info'] = array('ERROR: El máximo número de configuraciones permitidas por
                usuario es 20');
```

```
505         return $error;
506       } else {
507         $db->autocommit(FALSE);
508         $db->query('INSERT INTO configuracion (usuario,nombre,modificado)
509                 VALUES ('.$_SESSION['idusuario'].',"'.$nombre.'","'.date('Y-m-d H:i:s').'")')
                    ;
510         $respuesta = $db->query('SELECT LAST_INSERT_ID() AS idconfig');
511         $datos = $respuesta->fetch_assoc();
512         $respuesta->free();
513         $idconfig = $datos['idconfig'];
514       }
515     }
516     $nodos = array_merge($this->pc, $this->conmutador);
517     try {
518       foreach ($nodos as $elementoRed)
519         $elementoRed->storeConfig($idconfig);
520     }
521     catch (Exception $excepcion) {
522       // Si se produce un error durante el gurdado de la configuración se borran los
              archivos
523       // creados para evitar archivos residuales
524       if ($nueva) {
525         $respuesta = $db->query('SELECT * FROM conmutador WHERE idconfiguracion='.$idconfig)
                ;
526         for ($i = 0; $i < $respuesta->num_rows; $i++) {
527           $config = $respuesta->fetch_assoc();
528           Archivo::borrar($config['archivo']);
529         }
530         $respuesta->free();
531       }
532       throw $excepcion;
533     }
534     $db->commit();
535     $db->autocommit(TRUE);
536     return $error;
537   }
538
539   /**
540    * Borra la configuracion indicada
541    * @param $idconfig ID de de la confuración
542    * @return boolean
543    */
544   function deleteConfig($idconfig) {
545     $db = Database::conectar();
546     $respuesta = $db->query('SELECT * FROM conmutador WHERE idconfiguracion='.$idconfig);
547     for ($i = 0; $i < $respuesta->num_rows; $i++) {
548       $config = $respuesta->fetch_assoc();
549       Archivo::borrar($config['archivo']);
550     }
551     $respuesta->free();
552     $db->query('DELETE FROM configuracion WHERE usuario='.$_SESSION['idusuario'].' AND
            idconfiguracion='.$idconfig);
553     return TRUE;
554   }
555 }
556
557 ?>
```

admin/usuario.php

```php
<?php
/**
 * Script encargado de la interacción con el usuario
 * (login, logout, reserva y cancelación de turnos).
 * Para la comunicación entre el cliente y este script
 * se emplea AJAX utilizando JSON como notación para la
 * transferencia de información al cliente
 * @author Jesús Algeciras <jes.algeciras@gmail.com>
 * @version 2010-10-24
 *
 */

if (isset($_SERVER['HTTP_X_REQUESTED_WITH']) && $_SERVER['HTTP_X_REQUESTED_WITH'] == '
    XMLHttpRequest') {

  session_start();
  require_once('func_comunes.php');

  // Se lleva a cabo la petición correspondiente y se devuelve el resultado en formato JSON
      .
  // La estructura de la respuesta es siempre la misma:
  // array( stdout => (str), stderr => array(error => (str), info => (str)))
  // Esta respuesta es tratada segun corresponda en el navegador mediante Javascript
  try {
    switch ($_POST['act']) {
      case 'login':
        if (Usuario::login($_POST['usuario'], $_POST['contrasena'])) {
          echo json_encode(array('stdout' => 'Se ha identificado con éxito', 'stderr' =>
              array('error' => FALSE, 'info' => array())));
        } else {
          echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'validacion',
              'info' => array('ERROR: Usuario o contrasea incorrectos'))));
        }
        break;

      case 'logout':
        Usuario::logout();
        echo json_encode(array('stdout' => '', 'stderr' => array('error' => FALSE, 'info'
            => array())));
        break;

      case 'reservar':
        if (Usuario::reservar($_POST['idturno'],$_POST['idred'])) {
          echo json_encode(array('stdout' => 'La reserva ha sido realizada', 'stderr' =>
              array('error' => FALSE, 'info' => array())));
        } else {
          echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'validacion',
              'info' => array('ERROR: No ha sido posible realizar la reserva'))));
        }
        break;

      case 'cancelres':
        Usuario::cancelarReserva();
        echo json_encode(array('stdout' => 'La reserva ha sido cancelada', 'stderr' =>
            array('error' => FALSE, 'info' => array())));
        break;
    }
  }
  catch (Exception $excepcion) {
    if (Usuario::esAdmin()) {
```

```
53        echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'excepcion', '
              info' => array((string) $excepcion))));
54      } else {
55        echo json_encode(array('stdout' => '', 'stderr' => array('error' => 'excepcion', '
              info' => array('Se ha producido un error durante su petición'))));
56      }
57      exit();
58    }
59  } else {
60    header('HTTP/1.1 403 Forbidden');
61  }
62  ?>
```

include/administrar.php

```
1   <?php
2   if (Usuario::esAdmin()) {
3     if (isset($_GET['admin_acceso'])) {
4       include('include/admin_acceso.php');
5     } elseif (isset($_GET['admin_turnos'])) {
6       include('include/admin_turnos.php');
7     } elseif (isset($_GET['admin_reservas'])) {
8       include('include/admin_reservas.php');
9     } else {
10      include('include/admin_usuarios.php');
11    }
12  }
13  ?>
```

include/admin_acceso.php

```
1   <!-- Fechas -->
2   <ul>
3   <?php
4   $database = Database::conectar();
5
6   $respuesta = $database->query('SELECT * FROM acceso WHERE idacceso="reservas"');
7   if ($respuesta->num_rows > 0) {
8     $datos = $respuesta->fetch_assoc();
9     $fechas[] = $datos['idacceso'];
10    $respuesta->free();
11    echo '<li><h5>Reserva de turnos: </h5>Desde  '.date('G:i j-n-Y', strtotime(
          $datos['inicio'])).' hasta '.date('G:i j-n-Y', strtotime($datos['fin'])).""
          \n";
12  }
13
14  $respuesta = $database->query('SELECT * FROM acceso WHERE idacceso="aplicacion"');
15  if($respuesta->num_rows > 0) {
16    $datos = $respuesta->fetch_assoc();
17    $fechas[] = $datos['idacceso'];
18    $respuesta->free();
19    echo '<li><h5>Acceso a la aplicación: </h5>Desde  '.date('G:i j-n-Y',
          strtotime($datos['inicio'])).' hasta '.date('G:i j-n-Y', strtotime($datos[
          'fin'])).""\n";
```

```
20  }
21  ?>
22  </ul>
23  <br />
24  <ul id="edition">
25    <li><a href="" onclick="administrar.form.add(); return false;"><img alt="add" src="/
          images/add.png" />Nuevo/Editar</a></li>
26    <li><a href="" onclick="administrar.form.del(); return false;"><img alt="del" src="/
          images/del.png" />Borrar</a></li>
27    <li><a href="" onclick="administrar.form.delall(); return false;"><img alt="delall" src="
          /images/delall.png" />Borrar todos</a></li>
28  </ul>
29
30  <div id="admin_msg"></div>
31
32  <!-- Formualarios de administración -->
33  <div id="admin_forms">
34  <form id="add" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
        return false;">
35    <fieldset><legend>Fechas de acceso</legend> <input type="hidden" name="form" value="
          acceso" /> <input type="hidden" name="act" value="add" />
36      <div><p>Establecer fechas de acceso a la reserva de turnos y la aplicación:</p>
37      <label><span>Acceso a:</span><select name="idacceso">
38        <option value="" selected="selected"></option>
39        <option value="reservas">Reservas</option>
40        <option value="aplicacion">Aplicación</option>
41      </select></label>
42      <label><span>Fecha inicio:</span><input class="datepicker" type="text" name="
            fecha_inicio" maxlength="10" value="yyyy-mm-dd" /></label>
43      <label><span>Hora de inicio:</span><input type="text" name="hora_inicio" maxlength="5"
            value="hh:mm" /></label>
44      <label><span>Fecha final:</span><input class="datepicker" type="text" name="fecha_fin"
            maxlength="10" value="yyyy-mm-dd" /></label>
45      <label><span>Hora final:</span><input type="text" name="hora_fin" maxlength="5" value="
            hh:mm" /></label></div>
46      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
47      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
48    </fieldset>
49  </form>
50
51
52  <form id="del" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
        return false;">
53    <fieldset><legend>Fechas de acceso</legend>
54      <input type="hidden" name="form" value="acceso" /> <input type="hidden" name="act" value
            ="del" />
55      <div><p>Eliminar fecha de acceso:</p>
56      <label><span>Acceso a:</span><select name="idacceso">
57        <option value="" selected="selected"></option>
58  <?php
59  for ($i = 0; $i < count($fechas); $i++) {
60    echo '<option value="'.$fechas[$i].'">'.ucwords($fechas[$i]).'</option>'."\n";
61  }
62  ?>
63      </select></label></div>
64      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
65      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
66    </fieldset>
```

```
67  </form>
68
69
70  <form id="delall" action="" method="post" onsubmit="administrar.procesar($(this).serialize
        ()); return false;">
71    <fieldset><legend>Fechas de acceso</legend> <input type="hidden" name="form" value="
          acceso" />
72      <input type="hidden" name="act" value="delall" />
73      <div><p>Borrar todas las fechas de acceso</p></div>
74      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
75      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
76    </fieldset>
77  </form>
78  </div>
```

include/admin_reservas.php

```
1   <?php
2   $database = Database::conectar();
3   $dias = array(1 => 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes');
4   $reservas = array();
5   $turnos = array();
6   $usuarios = array();
7   $redes = array();
8
9   $respuesta = $database->query('SELECT turnos.*, usuario.usuario FROM ( SELECT * , (
10                SELECT idusuario FROM reserva WHERE reserva.idturno = turno.idturno AND
                      reserva.idred = red.idred
11                  ) AS idusuario FROM turno, red ) AS turnos NATURAL LEFT JOIN usuario ORDER
                        BY dia, inicio, idred');
12  for ($i = 0; $i < $respuesta->num_rows; $i++)
13    $reservas[] = $respuesta->fetch_assoc();
14  $respuesta->free();
15
16  $respuesta = $database->query('SELECT * FROM turno ORDER BY dia, inicio');
17  for ($i = 0; $i < $respuesta->num_rows; $i++)
18    $turnos[] = $respuesta->fetch_assoc();
19  $respuesta->free();
20
21  $respuesta = $database->query('SELECT * FROM usuario NATURAL LEFT JOIN reserva');
22  for ($i = 0; $i < $respuesta->num_rows; $i++)
23    $usuarios[] = $respuesta->fetch_assoc();
24  $respuesta->free();
25
26  $respuesta = $database->query('SELECT * FROM red');
27  for ($i = 0; $i < $respuesta->num_rows; $i++)
28    $redes[] = $respuesta->fetch_assoc();
29  $respuesta->free();
30  ?>
31
32  <table class="tabla" align="center" cellpadding="5" cellspacing="0">
33    <tr>
34      <th>TURNOS</th>
35      <th>HORARIO</th>
36      <th>GRUPOS</th>
37    </tr>
```

```
38  <?php
39  for ($i = 0, $j = 1; $i < count($reservas); $j++) {
40    echo '<tr><td>Turno '.($j).'</td>'."\n";
41    echo '<td>'.$dias[$reservas[$i]['dia']].' de '.substr($reservas[$i]['inicio'],0,5).' a '.
          substr($reservas[$i]['fin'],0,5).'</td>'."\n";
42    echo '<td><ul>'."\n";
43    $idturno = $reservas[$i]['idturno'];
44    while ($reservas[$i]['idturno'] == $idturno) {
45      echo '<li>Red '.$reservas[$i]['idred'].': '.($reservas[$i]['idusuario'] ? $reservas[$i
            ]['usuario'] : 'No reservada').'</li>'."\n";
46      $i++;
47    }
48    echo '</ul></td></tr>'."\n";
49  }
50  ?>
51  </table>
52
53  <ul id="edition">
54    <li><a href="" onclick="administrar.form.add(); return false;"><img alt="add" src="/
          images/add.png" />Nuevo/Editar</a></li>
55    <li><a href="" onclick="administrar.form.del(); return false;"><img alt="del" src="/
          images/del.png" />Borrar</a></li>
56    <li><a href="" onclick="administrar.form.delall(); return false;"><img alt="delall" src="
          /images/delall.png" />Borrar todos</a></li>
57  </ul>
58
59  <div id="admin_msg"></div>
60
61  <!-- Formualarios de administración -->
62  <div id="admin_forms">
63  <form id="add" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
        return false;">
64    <fieldset><legend>Reservas</legend>
65      <input type="hidden" name="form" value="reservas" /> <input type="hidden" name="act"
            value="add" />
66      <div><p>Establecer una reserva:</p>
67      <label><span>Usuario:</span> <select name="idusuario">
68        <option value="" selected="selected"></option>
69  <?php
70  for ($i = 0; $i < count($usuarios); $i++)
71    echo '<option value='.$usuarios[$i]['idusuario'].'>'.$usuarios[$i]['usuario'].'</option>'
          ."\n";
72  ?>
73      </select></label>
74      <label><span>Turno:</span> <select name="idturno">
75        <option value="" selected="selected"></option>
76  <?php
77  for ($i = 0; $i < count($turnos); $i++)
78    echo '<option value='.$turnos[$i]['idturno'].'>Turno '.($i + 1).'</option>'."\n";
79  ?>
80      </select></label>
81      <label><span>Red:</span> <select name="idred">
82        <option value="" selected="selected"></option>
83  <?php
84  for ($i = 0; $i < count($redes); $i++)
85    echo '<option value='.$redes[$i]['idred'].'> Red '.$redes[$i]['idred'].'</option>'."\n";
86  ?>
87      </select></label></div>
88      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
89      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
```

```
90    </fieldset>
91  </form>
92
93
94  <form id="del" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
        return false;">
95    <fieldset><legend>Reservas</legend>
96      <input type="hidden" name="form" value="reservas" /> <input type="hidden" name="act"
          value="del" />
97      <div><p>Eliminar reserva:</p>
98      <label><span>Usuario:</span><select name="idusuario">
99        <option value="" selected="selected"></option>
100 <?php
101 for ($i = 0; $i < count($usuarios); $i++) {
102   if ($usuarios[$i]['idturno'] != NULL)
103     echo '<option value='.$usuarios[$i]['idusuario'].'>'.$usuarios[$i]['usuario'].'</option
          >'."\n";
104 }
105 ?>
106     </select></label></div>
107     <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
          base de datos?</span></label></div>
108     <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
          Reiniciar" /></div>
109   </fieldset>
110 </form>
111
112
113 <form id="delall" action="" method="post" onsubmit="administrar.procesar($(this).serialize
      ()); return false;">
114   <fieldset><legend>Reservas</legend>
115   <input type="hidden" name="form" value="reservas" /> <input type="hidden" name="act"
        value="delall" />
116   <div><p>Eliminar todas las reservas</p></div>
117   <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la base
        de datos?</span></label></div>
118   <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
        Reiniciar" /></div>
119   </fieldset>
120 </form>
121 </div>
```

include/admin_turnos.php

```
1   <?php
2   $database = Database::conectar();
3   $dias = array(1 => 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes');
4   $reservas = array();
5   $turnos = array();
6
7   $respuesta = $database->query('SELECT turnos.*, usuario.usuario FROM (
8                 SELECT * , ( SELECT idusuario FROM reserva WHERE reserva.idturno = turno.
                      idturno AND reserva.idred = red.idred
9                 ) AS idusuario FROM turno, red ) AS turnos NATURAL LEFT JOIN usuario ORDER BY
                       dia, inicio, idred');
10  for ($i = 0; $i < $respuesta->num_rows; $i++)
11    $reservas[] = $respuesta->fetch_assoc();
12  $respuesta->free();
```

```
13
14  $respuesta = $database->query('SELECT * FROM turno ORDER BY dia, inicio');
15  for ($i = 0; $i < $respuesta->num_rows; $i++)
16    $turnos[] = $respuesta->fetch_assoc();
17  $respuesta->free();
18  ?>
19
20  <table class="tabla" align="center" cellpadding="5" cellspacing="0">
21    <tr>
22      <th>TURNOS</th>
23      <th>HORARIO</th>
24      <th>GRUPOS</th>
25    </tr>
26  <?php
27  for ($i = 0, $j = 1; $i < count($reservas); $j++) {
28    echo '<tr><td>Turno '.($j).'</td>'."\n";
29    echo '<td>'.$dias[$reservas[$i]['dia']].' de '.substr($reservas[$i]['inicio'],0,5).' a '.
          substr($reservas[$i]['fin'],0,5).'</td>'."\n";
30    echo '<td><ul>'."\n";
31    $idturno = $reservas[$i]['idturno'];
32    while ($reservas[$i]['idturno'] == $idturno) {
33      echo '<li>Red '.$reservas[$i]['idred'].': '.($reservas[$i]['idusuario'] ? $reservas[$i
            ]['usuario'] : 'No reservada').'</li>'."\n";
34      $i++;
35    }
36    echo '</ul></td></tr>'."\n";
37  }
38  ?>
39  </table>
40
41  <ul id="edition">
42    <li><a href="" onclick="administrar.form.add(); return false;"><img alt="add" src="/
          images/add.png" />Nuevo</a></li>
43    <li><a href="" onclick="administrar.form.edit(); return false;"><img alt="edit" src="/
          images/edit.png" />Editar</a></li>
44    <li><a href="" onclick="administrar.form.del(); return false;"><img alt="del" src="/
          images/del.png" />Borrar</a></li>
45    <li><a href="" onclick="administrar.form.delall(); return false;"><img alt="delall" src="
          /images/delall.png" />Borrar todos</a></li>
46  </ul>
47
48  <div id="admin_msg"></div>
49
50  <!-- Formualarios de administración -->
51  <div id="admin_forms">
52  <form id="add" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
        return false;">
53    <fieldset><legend>Turnos</legend>
54      <input type="hidden" name="form" value="turnos" /> <input type="hidden" name="act" value
            ="add" />
55      <div><p>Establecer nuevos turnos:</p>
56      <label><span>Día:</span> <select name="dia">
57        <option value="" selected="selected"></option>
58        <option value="1">Lunes</option>
59        <option value="2">Martes</option>
60        <option value="3">Miercoles</option>
61        <option value="4">Jueves</option>
62        <option value="5">Viernes</option>
63      </select></label>
64      <label><span>Hora de inicio:</span><input type="text" name="inicio" maxlength="5" value=
            "hh:mm" /></label>
```

```
65      <label><span>Hora final:</span><input type="text" name="fin" maxlength="5" value="hh:mm"
            /></label>
66      <label><span>N Sesiones:</span><input type="text" name="sesiones" maxlength="2" /></
            label></div>
67      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
68      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
69    </fieldset>
70  </form>
71
72
73  <form id="edit" action="" method="post" onsubmit="administrar.procesar($(this).serialize())
        ; return false;">
74    <fieldset><legend>Turnos</legend>
75      <input type="hidden" name="form" value="turnos" /> <input type="hidden" name="act" value
            ="edit" />
76      <div><p>Editar turno:</p>
77      <label><span>Turno:</span><select name="idturno">
78        <option value="" selected="selected"></option>
79  <?php
80  for ($i = 0; $i < count($turnos); $i++)
81    echo '<option value="'.$turnos[$i]['idturno'].'">Turno '.($i + 1).'</option>'."\n";
82  ?>
83      </select></label>
84      <label><span>Dia:</span><select name="dia">
85        <option value="" selected="selected"></option>
86        <option value="1">Lunes</option>
87        <option value="2">Martes</option>
88        <option value="3">Miercoles</option>
89        <option value="4">Jueves</option>
90        <option value="5">Viernes</option>
91      </select></label>
92      <label><span>Hora de inicio:</span><input type="text" name="inicio" maxlength="5" /></
            label>
93      <label><span>Hora final:</span><input type="text" name="fin" maxlength="5" /></label></
            div>
94      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
95      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
96    </fieldset>
97  </form>
98
99
100 <form id="del" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
        return false;">
101   <fieldset><legend>Turnos</legend>
102     <input type="hidden" name="form" value="turnos" /> <input type="hidden" name="act" value
            ="del" />
103     <div><p>Eliminar turno:</p>
104     <label><span>Turno:</span><select name="idturno">
105       <option value="" selected="selected"></option>
106 <?php
107 for ($i = 0; $i < count($turnos); $i++) {
108   if ($turnos[$i]['idturno'])
109     echo '<option value="'.$turnos[$i]['idturno'].'">Turno '.($i + 1).'</option>'."\n";
110 }
111 ?>
112     </select></label></div>
113     <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
```

```
114     <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
115   </fieldset>
116 </form>
117
118
119 <form id="delall" action="" method="post" onsubmit="administrar.procesar($(this).serialize
        ()); return false;">
120   <fieldset><legend>Turnos</legend>
121     <input type="hidden" name="form" value="turnos" /> <input type="hidden" name="act" value
            ="delall" />
122     <div><p>Borrar todos los turnos</p></div>
123     <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
124     <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
125   </fieldset>
126 </form>
127 </div>
```

include/admin_usuarios.php

```
1  <?php
2  $database = Database::conectar();
3  $dias = array(1 => 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes');
4  $usuarios = array();
5
6  $respuesta = $database->query('SELECT * FROM usuario');
7  for ($i = 0; $i < $respuesta->num_rows; $i++)
8    $usuarios[$i] = $respuesta->fetch_assoc();
9  $respuesta->free();
10 ?>
11 <!-- Cuentas de usuarios -->
12 <table class="tabla" align="center" cellpadding="5" cellspacing="0">
13   <tr>
14     <th>USUARIOS</th>
15     <th>CONTRASEAS</th>
16     <th>ADMINISTRADOR</th>
17   </tr>
18 <?php
19 for ($i = 0; $i < count($usuarios); $i++) {
20   echo '<tr>'."\n";
21   echo '<td>'.$usuarios[$i]['usuario'].'</td>'."\n";
22   echo '<td>'.$usuarios[$i]['contrasena'].'</td>'."\n";
23   echo '<td>'.($usuarios[$i]['administrador'] ? 'Si' : 'No').'</td>'."\n";
24   echo '</tr>'."\n";
25 }
26 ?>
27 </table>
28
29 <ul id="edition">
30   <li><a href="" onclick="administrar.form.add(); return false;"><img alt="add" src="/
         images/add.png" />Nuevo</a></li>
31   <li><a href="" onclick="administrar.form.edit(); return false;"><img alt="edit" src="/
         images/edit.png" />Editar</a></li>
32   <li><a href="" onclick="administrar.form.del(); return false;"><img alt="del" src="/
         images/del.png" />Borrar</a></li>
```

```
33    <li><a href="" onclick="administrar.form.delall(); return false;"><img alt="delall" src="
          /images/delall.png" />Borrar todos</a></li>
34  </ul>
35
36  <div id="admin_msg"></div>
37
38  <!-- Formualarios de administración -->
39  <div id="admin_forms">
40  <form id="add" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
          return false;">
41    <fieldset><legend>Usuarios</legend> <input type="hidden" name="form" value="usuarios" />
          <input type="hidden" name="act" value="add" />
42      <div><p>Generar nuevas cuentas de usuarios:</p>
43      <label><span>N de usuarios:</span><input type="text" name="num_usuarios" maxlength="2"
          /></label></div>
44      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
          base de datos?</span></label></div>
45      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
          Reiniciar" /></div>
46    </fieldset>
47  </form>
48
49
50  <form id="edit" action="" method="post" onsubmit="administrar.procesar($(this).serialize())
        ; return false;">
51    <fieldset><legend>Usuarios</legend> <input type="hidden" name="form" value="usuarios" />
          <input type="hidden" name="act" value="edit" />
52      <div><p>Editar usuario:</p>
53      <label><span>Usuario:</span><select name="idusuario">
54        <option value="" selected="selected"></option>
55  <?php
56  for ($i = 0; $i < count($usuarios); $i++) {
57    echo '<option value="'.$usuarios[$i]['idusuario'].'">'.$usuarios[$i]['usuario'].'</option
          >'."\n";
58  }
59  ?>
60      </select></label>
61      <label><span>Nombre:</span><input type="text" name="usuario" maxlength="10" /></label>
62      <label><span>Contrasea:</span><input type="text" name="contrasena" maxlength="10" /></
          label>
63      <label><span>Administrador:</span><select name="administrador">
64        <option value="" selected="selected"></option>
65        <option value="0">No</option>
66        <option value="1">Si</option>
67      </select></label></div>
68      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
          base de datos?</span></label></div>
69      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
          Reiniciar" /></div>
70    </fieldset>
71  </form>
72
73
74  <form id="del" action="" method="post" onsubmit="administrar.procesar($(this).serialize());
          return false;">
75    <fieldset><legend>Usuarios</legend> <input type="hidden" name="form" value="usuarios" />
          <input type="hidden" name="act" value="del" />
76      <div><p>Eliminar usuario:</p>
77      <label><span>Usuario:</span><select name="idusuario">
78        <option value="" selected="selected"></option>
79  <?php
80  for ($i = 0; $i < count($usuarios); $i++) {
```

```
81    echo '<option value="'.$usuarios[$i]['idusuario'].'">'.$usuarios[$i]['usuario'].'</option
          >'."\n";
82  }
83  ?>
84      </select></label></div>
85      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
86      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
87    </fieldset>
88  </form>
89
90
91  <form id="delall" action="" method="post" onsubmit="administrar.procesar($(this).serialize
        ()); return false;">
92    <fieldset><legend>Usuarios</legend> <input type="hidden" name="form" value="usuarios" />
          <input type="hidden" name="act" value="delall" />
93      <div><p>Borrar todos los usuarios sin derechos de administración</p></div>
94      <div><label><input type="checkbox" name="confirmado" /><span>Confirmar cambios en la
            base de datos?</span></label></div>
95      <div class="buttons"><input type="submit" value="Enviar" /><input type="reset" value="
            Reiniciar" /></div>
96    </fieldset>
97  </form>
98  </div>
```

include/inicio.php

```
1   <?php
2   $database = Database::conectar();
3   ?>
4   <!-- Fechas -->
5   <?php
6   $msg = '';
7   $respuesta = $database->query('SELECT * FROM acceso WHERE idacceso="reservas"');
8   if ($respuesta->num_rows > 0) {
9     $fecha = $respuesta->fetch_assoc();
10    $msg .= '<li><h5>Reserva de turnos: </h5>Desde  '.date('G:i j-n-Y', strtotime(
          $fecha['inicio'])).' hasta '.date('G:i j-n-Y', strtotime($fecha['fin'])).'
          \n";
11  }
12  $respuesta->free();
13
14  $respuesta = $database->query('SELECT * FROM acceso WHERE idacceso="aplicacion"');
15  if($respuesta->num_rows > 0) {
16    $fecha = $respuesta->fetch_assoc();
17    $msg .= '<li><h5>Acceso a la aplicación: </h5>Desde  '.date('G:i j-n-Y',
          strtotime($fecha['inicio'])).' hasta '.date('G:i j-n-Y', strtotime($fecha[
          'fin'])).'\n";
18  }
19  $respuesta->free();
20
21  if($msg != '')
22    echo '<ul id="fechas">'.$msg.'</ul>';
23  ?>
24
25  <h4>Bienvenido,</h4>
26  <p>A través de esta web podrá acceder tanto a la reserva de turnos para
```

```
27  la realización de las prácticas como a la aplicación a través de la cual
28  se llevarán a cabo:</p>
29  <br />
30  <ul>
31  <li><b>Reserva de turnos:</b> Para acceder a la reserva de turnos haga click en el enlace
32  correspondiente en el menú de navegación. Una vez establecido el horario
33  de turnos (en caso contrario se indicará) podrá realizar la reserva de
34  una red durante un turno determinado.</li>
35  <li><b>Prácticas:</b> Al hacer click en el enlace correspondiente a la aplicación en el
36  menú, se abrirá la ventana externa donde se encuentra la aplicación. En
37  esta ventana se proporcionan los enlaces a la documentación necesaria
38  para el manejo de la misma.</li>
39  </ul>
40  <br />
41  <p>Tanto la reserva de turnos como el acceso a la aplicación estarán sujetos
42  a unas determinadas fechas de acceso. Dichas fechas se indicarán en esta
43  misma página una vez hayan sido establecidas.</p>
44  <br />
45  <br />
```

include/turnos.php

```
1   <?php
2   //Se obtiene la información de los turnos de la base de datos
3   $database = Database::conectar();
4   $reservar = Usuario::accesoReserva();
5   $dias = array(1 => 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes');
6   $reservas = array();
7
8   $respuesta = $database->query('SELECT turnos.*, usuario.usuario FROM ( SELECT * , (
9                 SELECT idusuario FROM reserva WHERE reserva.idturno = turno.idturno AND
                       reserva.idred = red.idred
10                ) AS idusuario FROM turno, red ) AS turnos NATURAL LEFT JOIN usuario ORDER BY
                       dia, inicio, idred');
11  if ($respuesta->num_rows == 0) {
12    echo '<div class="alert">El horario de los turnos aun no ha sido establecido</div>';
13  } else {
14    for ($i = 0; $i < $respuesta->num_rows; $i++)
15      $reservas[] = $respuesta->fetch_assoc();
16    $respuesta->free();
17
18    if (Usuario::logged()) {
19      $respuesta = $database->query('SELECT * FROM reserva WHERE idusuario='.$_SESSION['
            idusuario'].' LIMIT 1');
20      $turno = $respuesta->num_rows == 0 ? array() : $respuesta->fetch_assoc();
21      $respuesta->free();
22    }
23  ?>
24  <!-- Presentación y reserva de turnos -->
25  <table class="tabla" align="center" cellpadding="5" cellspacing="0">
26    <tr>
27      <th>TURNOS</th>
28      <th>HORARIO</th>
29      <th>GRUPOS</th>
30    </tr>
31  <?php
32    for ($i = 0, $j = 1; $i < count($reservas); $j++) {
33      echo '<tr><td>Turno '.($j).'</td>'."\n";
```

```
34      echo '<td>'.$dias[$reservas[$i]['dia']].' de '.substr($reservas[$i]['inicio'],0,5).' a
            '.substr($reservas[$i]['fin'],0,5).'</td>'."\n";
35      echo '<td><ul>'."\n";
36      $idturno = $reservas[$i]['idturno'];
37      while ($reservas[$i]['idturno'] == $idturno) {
38        echo '<li>Red '.$reservas[$i]['idred'].': ';
39        // Fuera del plazo de reserva o el turno ya ha sido reservado por otro usuario
40        if (!$reservar) {
41          echo $reservas[$i]['idusuario'] ? $reservas[$i]['usuario'] : 'No reservada';
42        } else {
43          // Es posible cancelar la reserva
44          if ($turno['idturno'] == $reservas[$i]['idturno'] && $turno['idred'] == $reservas[$i
                ]['idred']) {
45            echo $reservas[$i]['usuario'].' <a href="" onclick="usuario.cancelres(); return
                  false;">(Cancelar reserva)</a>';
46          // Es posible reservar el turno
47          } elseif (count($turno) == 0 && !$reservas[$i]['idusuario']) {
48            echo '<a href="" onclick="usuario.reservar('.$reservas[$i]['idturno'].', '.
                  $reservas[$i]['idred'].'); return false;">Reservar</a>';
49          } else {
50            echo $reservas[$i]['idusuario'] ? $reservas[$i]['usuario'] : 'No reservada';
51          }
52        }
53        echo '</li>'."\n";
54        $i++;
55      }
56      echo '</ul></td></tr>'."\n";
57    }
58  ?>
59  </table>
60  <?php
61  }
62  ?>
```

css/interfaz.css

```
1   @charset "utf-8";
2
3   @import "reset.css";
4
5   @import "jquery-ui-1.7.2.custom.css";
6
7   html,body {
8     height: 100%;
9   }
10
11  body {
12    font-family: "Trebuchet MS", Arial, Helvetica, Tahoma, sans-serif;
13    font-size: 12px;
14    letter-spacing: 1px;
15    background: #EEE;
16  }
17
18  a,span.disabled {
19    text-decoration: none;
20    color: #888;
21  }
22
```

```
23  a:hover,a:active {
24    color: #FFB700;
25  }
26
27  h1,h2,h3,h4,h5,h6 {
28    color: #FFB700;
29    margin: 10px 0;
30  }
31
32  h1 {
33    font-size: 20px;
34  }
35
36  h2 {
37    font-size: 18px;
38  }
39
40  h3 {
41    font-size: 16px;
42  }
43
44  h4 {
45    font-size: 14px;
46  }
47
48  h5 {
49    font-size: 14px;
50    display: inline;
51  }
52
53  p {
54    line-height: 18px;
55    text-indent: 36px;
56  }
57
58  li {
59    list-style: disc;
60    line-height: 18px;
61    margin-left: 30px;
62  }
63
64  #main {
65    margin: 0 auto;
66    width: 960px;
67    position: relative;
68    background: #FFF url(../images/app_header.jpg) no-repeat top;
69  }
70
71  /* Menu */
72  .menu {
73    font-size: 16px;
74    font-weight: bold;
75    letter-spacing: 2px;
76  }
77
78  .menu ul {
79    height: 35px;
80  }
81
82  .menu li {
83    float: left;
84    list-style: none;
```

```
85  }
86
87  .menu li a,.menu li span.disabled {
88    display: block;
89    line-height: 30px;
90    padding: 0 25px;
91    text-align: center;
92  }
93
94  .menu li a:hover,.menu ul li:hover a {
95    color: #FFB700;
96  }
97
98  .menu li ul {
99    background: #000;
100   display: none;
101   height: auto;
102   position: absolute;
103   width: 225px;
104   z-index: 200;
105 }
106
107 .menu li:hover ul {
108   display: block;
109 }
110
111 .menu li li {
112   display: block;
113   float: none;
114   width: 225px;
115   margin: 0px;
116   padding: 0px;
117 }
118
119 .menu li:hover li a {
120   background: none;
121   color: #FFF;
122 }
123
124 .menu li ul a {
125   display: block;
126   height: 35px;
127   font-size: 14px;
128   font-weight: normal;
129   padding: 0 10px 0 15px;
130   text-align: left;
131 }
132
133 .menu li ul a:hover,.menu li ul li:hover a {
134   background: #FFB700;
135   color: #FFF;
136 }
137
138 .menu p {
139   clear: left;
140 }
141
142 .menu .active a {
143   background: #FFB700;
144   color: #FFF;
145 }
146
```

```
147 .menu .active a:hover,.menu .active:hover a {
148   color: #FFF;
149 }
150
151 /* Barra de Navegación */
152 #navigation {
153   position: absolute;
154   top: 80px;
155 }
156
157 /* Boton de login */
158 #login_button {
159   position: absolute;
160   top: 125px;
161   right: 20px;
162   font-size: 12px;
163   letter-spacing: 1px;
164 }
165
166 #login_button li {
167   display: inline;
168 }
169
170 /* Contenido pagina */
171 #main_content {
172   padding: 115px 50px 0 50px;
173   height: 600px;
174   overflow-y: hidden;
175 }
176
177 #instrucciones {
178   height: 500px;
179   overflow-y: auto;
180 }
181
182 .panel {
183   border: 2px solid #444;
184 }
185
186 .title {
187   display: block;
188   height: 20px;
189   padding: 0 10px;
190   background: #444;
191   color: #FFF;
192   line-height: 20px;
193   z-index: 10;
194 }
195
196 .title a {
197   float: right;
198   font-style: italic;
199   font-size: 10px;
200   color: #EEE;
201 }
202
203 #instrucciones {
204   padding: 50px;
205 }
206
207 #scheme {
208   height: 346px;
```

```
209     width: 556px;
210     position: absolute;
211     top: 115px;
212     left: 0;
213     overflow: hidden;
214   }
215
216   #scheme #diagrama {
217     display: block;
218     height: 300px;
219     width: 500px;
220     position: absolute;
221     top: 33px;
222     left: 28px;
223   }
224
225   #console {
226     width: 556px;
227     height: 248px;
228     position: absolute;
229     top: 463px;
230     left: 0;
231   }
232
233   #console div {
234     height: 185px;
235     padding: 10px 20px;
236     overflow-y: auto;
237   }
238
239   #console p {
240     text-indent: 0;
241   }
242
243   #console #info {
244     height: 20px;
245     padding: 0 10px 0 0;
246     overflow: hidden;
247     border-bottom: 1px solid #888;
248     text-align: right;
249   }
250
251   #console #info span {
252     font-style: italic;
253   }
254
255   #console #cursor {
256     text-decoration: blink;
257   }
258
259   #console p.error_msg {
260     color: #8A1F11;
261   }
262
263   #config {
264     float: right;
265     height: 596px;
266     width: 398px;
267     position: absolute;
268     top: 115px;
269     right: 0;
270     overflow: hidden;
```

```
271  }
272
273  #config h2 {
274    padding: 30px 0 0 0;
275    text-align: center;
276  }
277
278  #config ul {
279    margin-bottom: 20px;
280  }
281
282  #config form {
283    margin-top: 20px;
284  }
285
286  #config form,#config iframe {
287    display: none;
288  }
289
290  /* Tooltips */
291  .jqm0verlay {
292    background: #27799D;
293  }
294
295  .tooltip {
296    padding: 10px 5px;
297  }
298
299  .tooltip h5 {
300    padding: 10px;
301    text-transform: uppercase;
302    color: #FFB700;
303  }
304
305  .tooltip a {
306    font-size: 10px;
307  }
308
309  .tooltip div {
310    padding: 10px 0 10px 10px;
311    border-top: 2px solid #444;
312    margin-top: 5px;
313  }
314
315  .tooltip li p {
316    text-indent : 0px;
317  }
318
319  .tooltip div a {
320    font-size: 12px;
321  }
322
323  /* Formularios */
324  form {
325    width: 640px;
326    text-align: left;
327    margin: 0 auto;
328  }
329
330  form.narrow {
331    width: 300px;
332  }
```

```
333
334   form p {
335     clear: both;
336     margin-bottom: 10px;
337   }
338
339   form div {
340     width: 290px;
341     margin: 0 5px;;
342     float: left;
343   }
344
345   form .buttons {
346     width: 290px;
347     padding-top: 20px;
348     float: right;
349   }
350
351   form.narrow .buttons {
352     width: 290px;
353   }
354
355   form .buttons input {
356     float: right;
357     margin-left: 20px;
358   }
359
360   form fieldset {
361     border: 2px solid #CCC;
362     padding: 10px;
363   }
364
365   form.narrow fieldset {
366     border: none;
367   }
368
369   form legend {
370     font-size: 14px;
371     font-weight: bold;
372     color: #FFB700;
373     padding: 10px;
374   }
375
376   form label {
377     display: block;
378     position: relative;
379     vertical-align: middle;
380     margin: 0 0 20px 0;
381   }
382
383   form label span {
384     width: 260px;
385     text-align: left;
386     position: absolute;
387     left: 20px;
388     padding: 0 0 0 10px;
389     text-align: left;
390   }
391
392   form label span:first-child {
393     width: 120px;
394     line-height: 24px;
```

```
395     text-align: right;
396     padding: 0 20px 0 0;
397     position: absolute;
398     left: 0;
399     right: 50%;
400   }
401
402   form input,form select {
403     width: 100px;
404     padding: 2px 10px;
405     margin: 0 0 0 50%;
406   }
407
408   form select {
409     width: 124px;
410   }
411
412   form input[type="checkbox"],form input[type="radio"] {
413     width: 5px;
414     margin: 0;
415     padding: 0;
416   }
417
418   form input[type="file"] {
419     padding: 2px 10px;
420     margin: 20px 0 0 20%;
421   }
422
423   form select[name="idconfig"] {
424     width: 200px;
425     padding: 2px 10px;
426     margin: 20px 0 0 20%;
427   }
428
429   form input.invalid,form select.invalid {
430     border: solid 1px #FF6440;
431   }
432
433   form div.input_error,form div.input_info {
434     width: 240px;
435     position: absolute;
436     left: 50%;
437     top: 25px;
438     font-size: 10px;
439     overflow: hidden;
440   }
441
442   form div.input_error {
443     color: #FF6440;
444   }
445
446   form div.input_info {
447     color: #264409;
448   }
449
450   form input[type="checkbox"]+div.input_error,form input[type="radio"]+div.input_error
451     {
452     left: 0;
453     top: 20px;
454   }
455
```

```
456 form input[type="file"]+div.input_error,form select[name="idconfig"]+div.input_error,form
        select[name="idconfig"]+div.input_info
457   {
458   left: 20%;
459   top: 60px;
460 }
461
462 /* Tablas */
463 .tabla {
464   width: 640px;
465   text-align: center;
466   margin: 20px auto;
467   border: 2px solid #444;
468 }
469
470 .tabla th,.tabla td {
471   padding: 10px 0;
472   min-width: 100px;
473 }
474
475 .tabla th {
476   background: #444;
477   color: #FFF;
478 }
479
480 /* Mensajes */
481 .info,.alert,.error {
482   width: 400px;
483   padding: 10px 10px 10px 40px;
484   margin: 40px auto 20px auto;
485   border: 2px solid #000000;
486   line-height: 18px;
487 }
488
489 .info {
490   background: #E6EFC2 url(../images/info.png) 5px center no-repeat;
491   color: #264409;
492   border-color: #C6D880;
493 }
494
495 .alert {
496   background: #FFF6BF url(../images/alert.png) 5px center no-repeat;
497   color: #A67700;
498   border-color: #FFD324;
499 }
500
501 .error {
502   background: #FBE3E4 url(../images/error.png) 5px center no-repeat;
503   color: #8A1F11;
504   border-color: #FBC2C4;
505 }
506
507 .blocked {
508   width: 480px;
509   padding: 10px 20px;
510   position: relative;
511 }
512
513 .blocked p {
514   text-align: left;
515   padding-bottom: 20px;
516 }
```

css/redes.css

```
@charset "utf-8";

@import "reset.css";

@import "jquery-ui-1.7.2.custom.css";

html,body {
  height: 100%;
}

body {
  font-family: "Trebuchet MS", Arial, Helvetica, Tahoma, sans-serif;
  font-size: 12px;
  letter-spacing: 1px;
  background: #EEE;
}

a {
  text-decoration: none;
  color: #888;
}

a:hover,a:active {
  color: #FFB700;
}

h1,h2,h3,h4,h5,h6 {
  color: #FFB700;
  margin: 10px 0;
}

h1 {
  font-size: 20px;
}

h2 {
  font-size: 18px;
}

h3 {
  font-size: 16px;
}

h4 {
  font-size: 14px;
}

h5 {
  font-size: 14px;
  display: inline;
}

p {
  line-height: 18px;
  text-indent: 36px;
```

```
56  }
57
58  li {
59    list-style: disc;
60    line-height: 18px;
61    margin-left: 30px;
62  }
63
64  #main {
65    margin: 0 auto -100px auto;
66    width: 960px;
67    height: auto !important;
68    height: 100%;
69    min-height: 100%;
70    position: relative;
71    background: #FFF url(../images/header.jpg) no-repeat top;
72  }
73
74  /* Menu */
75  .menu {
76    font-size: 16px;
77    font-weight: bold;
78    letter-spacing: 2px;
79  }
80
81  .menu ul {
82    height: 35px;
83  }
84
85  .menu li {
86    float: left;
87    list-style: none;
88  }
89
90  .menu li a {
91    display: block;
92    line-height: 30px;
93    padding: 0 25px;
94    text-align: center;
95  }
96
97  .menu li a:hover,.menu ul li:hover a {
98    color: #FFB700;
99  }
100
101 .menu li ul {
102   background: #000;
103   display: none;
104   height: auto;
105   position: absolute;
106   width: 225px;
107   z-index: 200;
108 }
109
110 .menu li:hover ul {
111   display: block;
112 }
113
114 .menu li li {
115   display: block;
116   float: none;
117   width: 225px;
```

```
118     margin: 0px;
119     padding: 0px;
120   }
121
122   .menu li:hover li a {
123     background: none;
124     color: #FFF;
125   }
126
127   .menu li ul a {
128     display: block;
129     height: 35px;
130     font-size: 14px;
131     font-weight: normal;
132     padding: 0 10px 0 15px;
133     text-align: left;
134   }
135
136   .menu li ul a:hover,.menu li ul li:hover a {
137     background: #FFB700;
138     color: #FFF;
139   }
140
141   .menu p {
142     clear: left;
143   }
144
145   .menu .active a {
146     background: #FFB700;
147     color: #FFF;
148   }
149
150   .menu .active a:hover,.menu .active:hover a {
151     color: #FFF;
152   }
153
154   /* Barra de Navegación */
155   #navigation {
156     position: absolute;
157     top: 110px;
158   }
159
160   /* Boton de login */
161   #login_button {
162     position: absolute;
163     top: 170px;
164     right: 20px;
165     font-size: 12px;
166     letter-spacing: 1px;
167   }
168
169   #login_button li {
170     display: inline;
171   }
172
173   /* Contenido pagina */
174   #main_content {
175     padding: 200px 100px 120px 100px;
176     height: auto !important;
177     height: 200px;
178     min-height: 200px;
179   }
```

```
180
181   #fechas {
182     border: 2px solid #CCC;
183     padding: 10px;
184     margin: 0 0 40px 0;
185   }
186
187   /* Footer */
188   #footer {
189     margin: 0 auto -20px auto;
190     width: 960px;
191     height: 100px;
192     background: #000;
193     overflow: hidden;
194     position: relative;
195     color: #FFF;
196   }
197
198   #footer a:hover,#footer a:active {
199     color: #FFF;
200   }
201
202   #links {
203     position: absolute;
204     top: 15px;
205     left: 15px;
206   }
207
208   #links p {
209     margin-right: 10px;
210     display: inline;
211   }
212
213   #contact_info {
214     position: absolute;
215     bottom: 20px;
216     right: 20px;
217     text-align: right;
218   }
219
220   #contact_info div {
221     float: right;
222   }
223
224   /* Menu edición */
225   #edition {
226     float: right;
227   }
228
229   #edition li {
230     list-style: none;
231     display: inline;
232     margin: 10px;
233   }
234
235   #edition li img {
236     width: 16px;
237     height: 16px;
238     vertical-align: sub;
239     margin: 0 5px;
240   }
241
```

```
242  /* Formularios */
243  form {
244    width: 640px;
245    text-align: left;
246    margin: 0 auto;
247  }
248
249  form.narrow {
250    width: 300px;
251  }
252
253  form p {
254    clear: both;
255    margin-bottom: 10px;
256  }
257
258  form div {
259    width: 290px;
260    margin: 0 5px;;
261    float: left;
262  }
263
264  form .buttons {
265    width: 290px;
266    padding-top: 20px;
267    float: right;
268  }
269
270  form.narrow .buttons {
271    width: 290px;
272  }
273
274  form .buttons input {
275    float: right;
276    margin-left: 20px;
277  }
278
279  form fieldset {
280    border: 2px solid #CCC;
281    padding: 10px;
282  }
283
284  form.narrow fieldset {
285    border: none;
286  }
287
288  form legend {
289    font-size: 14px;
290    font-weight: bold;
291    color: #FFB700;
292    padding: 10px;
293  }
294
295  form label {
296    display: block;
297    position: relative;
298    vertical-align: middle;
299    margin: 0 0 20px 0;
300  }
301
302  form label span {
303    width: 260px;
```

```
304     text-align: left;
305     position: absolute;
306     left: 20px;
307     padding: 0 0 0 10px;
308     text-align: left;
309   }
310
311   form label span:first-child {
312     width: 120px;
313     line-height: 24px;
314     text-align: right;
315     padding: 0 20px 0 0;
316     position: absolute;
317     left: 0;
318     right: 50%;
319   }
320
321   form input,form select {
322     width: 100px;
323     padding: 2px 10px;
324     margin: 0 0 0 50%;
325   }
326
327   form select {
328     width: 124px;
329   }
330
331   form input[type="checkbox"],form input[type="radio"] {
332     width: 5px;
333     margin: 0;
334     padding: 0;
335   }
336
337   form input[type="file"] {
338     width: 100px;
339     padding: 2px 10px;
340     margin: 20px 0 0 20%;
341   }
342
343   form input.invalid,form select.invalid {
344     border: solid 1px #FF6440;
345   }
346
347   form div.input_error {
348     width: 240px;
349     position: absolute;
350     left: 50%;
351     top: 25px;
352     color: #FF6440;
353     font-size: 10px;
354     overflow: hidden;
355   }
356
357   form input[type="checkbox"]+div.input_error,form input[type="radio"]+div.input_error
358     {
359     left: 0;
360     top: 20px;
361   }
362
363   form input[type="file"]+div.input_error {
364     left: 20%;
365     top: 60px;
```

```
366  }
367
368  #admin_forms form {
369    display: none;
370    margin-top: 20px;
371  }
372
373  /* Tablas */
374  .tabla {
375    width: 640px;
376    text-align: center;
377    margin: 20px auto;
378    border: 2px solid #444;
379  }
380
381  .tabla th,.tabla td {
382    padding: 10px 0;
383    min-width: 100px;
384  }
385
386  .tabla th {
387    background: #444;
388    color: #FFF;
389  }
390
391  /* Mensajes */
392  .info,.alert,.error {
393    width: 400px;
394    padding: 10px 10px 10px 40px;
395    margin: 0 auto 20px auto;
396    border: 2px solid #000000;
397    line-height: 18px;
398  }
399
400  .info {
401    background: #E6EFC2 url(../images/info.png) 5px center no-repeat;
402    color: #264409;
403    border-color: #C6D880;
404  }
405
406  .alert {
407    background: #FFF6BF url(../images/alert.png) 5px center no-repeat;
408    color: #A67700;
409    border-color: #FFD324;
410  }
411
412  .error {
413    background: #FBE3E4 url(../images/error.png) 5px center no-repeat;
414    color: #8A1F11;
415    border-color: #FBC2C4;
416  }
417
418  .dialog p {
419    margin: 5px 10px;
420    text-indent: 0;
421  }
422
423  .dialog #admin_msg {
424    width: 260px;
425    margin: 0 auto 20px auto;
426  }
427
```

```
#admin_msg {
  margin: 30px auto;
  text-align: center;
}
```

js/administrar.js

```
// Administración
var administrar = {
  // Funciones para mostrar por pantalla los diferentes formularios
  form : {
    add : function() {
      this.mostrar('add');
    },
    edit : function() {
      this.mostrar('edit');
    },
    del : function() {
      this.mostrar('del');
    },
    delall : function() {
      this.mostrar('delall');
    },
    mostrar : function(formulario) {
      $('#admin_forms input, #admin_forms select').removeClass('invalid');
      $('#admin_forms div.input_error').remove();
      $('#admin_forms #' + formulario).siblings().hide();
      $('#admin_forms #' + formulario).toggle();
    }
  },

  // Función encargada de mosrtrar los errores producidos tras la última
  // petición del administrador
  mostrarErrores : function(data) {
    if (data.stderr.error == 'validacion') {
      $.each(data.stderr.info, function(i, value) {
        $('input[name=' + i + '], select[name=' + i + ']').addClass('invalid').after('<div
            class=input_error>\'--> ' + value + '</div>');
      });
    } else {
      var msg = '';
      $.each(data.stderr.info, function(i, value) {
        msg = msg + '<p>' + value + '</p>';
      });
      mensaje.error('#admin_msg', msg);
    }
  },

  // Función para el envio del formulario mediante AJAX y el tratamieto de la
  // respuesta del servidor
  procesar : function(inputs) {
    $.post('/admin/proc_form.php', inputs, function(data) {
      if (data.stderr.error) {
        administrar.mostrarErrores(data);
      } else {
        var msg = 'La base de datos se ha actualizado con éxito, haga <a href="" onclick="
            window.location.reload(); return false;">click aquí</a> para ver los cambios.';
        mensaje.info('#admin_msg', msg);
```

```
50        }
51      }, 'json');
52    }
53  };
```

js/func_varias.js

```
1   $(document).ready(function() {
2     $('tr:even').css('background-color', '#FFE7AC');
3     $('.datepicker').datepicker({
4       dateFormat : 'yy-mm-dd',
5       firstDay : 1
6     });
7     $('input[type=reset], input[type=submit]').click(function() {
8       $(this).parents('form').find('.input_info').remove();
9       $(this).parents('form').find('.input_error').remove();
10      $(this).parents('form').find('input').removeClass('invalid');
11      $(this).parents('form').find('select').removeClass('invalid');
12    });
13    $('input[name="default"]').click(function() {
14      if ($(this).attr('checked') == true) {
15        $(this).parent().prevAll().children('input').attr('disabled', 'disabled');
16      } else {
17        $(this).parent().prevAll().children('input').removeAttr('disabled');
18      }
19    });
20    $.cookie('cookies', 'TRUE');
21    if (!$.cookie('cookies')) {
22      $('#msg').addClass('alert').html('La página que estás viendo requiere para su
          funcionamiento el uso de cookies. Es necesario que las habilite.');
23    } else {
24      $.cookie('cookies', null);
25    }
26  });
27
28  // Ventanas externas
29  var ventanas = {
30    ventana : null,
31    abrir : function(url, titulo, opciones) {
32      this.ventana = window.open(url, titulo, opciones);
33      if (window.focus) {
34        this.ventana.focus();
35      }
36    }
37  };
38
39  // Mensajes
40  var mensaje = {
41    info : function(div, msg) {
42      this.mostrar(div, msg, 'info');
43    },
44    alert : function(div, msg) {
45      this.mostrar(div, msg, 'alert');
46    },
47    error : function(div, msg) {
48      this.mostrar(div, msg, 'error');
49    },
50    mostrar : function(div, msg, clase) {
```

```
51      $(div).fadeTo(200, 0.1, function() {
52        $(this).removeClass().addClass(clase).html(msg).fadeTo(900, 1);
53      });
54    }
55  };
56
57  // Acciones del usuario
58  var usuario = {
59    form : '<div id="login" class="dialog"><div id="admin_msg"></div>'
60        + '<form id="login_form" class="narrow" action=""><fieldset>'
61        + '<div><label><span>Nombre</span><input id="usuario" type="text" name="usuario"
              maxlength="10" /></label>'
62        + '<label><span>Contrasea</span><input id="contrasena" type="password" name="
              contrasena" maxlength="10" /></label></div>'
63        + '</fieldset></form></div>',
64
65    login : function() {
66      $(this.form).dialog({
67        bgiframe : true,
68        minHeight : 200,
69        width : 340,
70        resizable : false,
71        modal : true,
72        title : 'Login',
73        buttons : {
74          'Enviar' : function() {
75            $.post('/admin/usuario.php', {act : 'login', usuario : $('#usuario').val(),
                  contrasena : $('#contrasena').val()}, function(data) {
76              if (data.stderr.error) {
77                var msg = '';
78                $.each(data.stderr.info, function(i, value) {
79                  msg = msg + '<p>' + value + '</p>';
80                });
81                mensaje.error('#admin_msg', msg);
82              } else {
83                mensaje.info('#admin_msg', '<p>' + data.stdout + '</p>');
84                setTimeout("window.location.reload()");
85              }
86            }, 'json');
87          }
88        }
89      });
90    },
91
92    logout : function() {
93      $.post('/admin/usuario.php', {act : 'logout'}, function() {
94        document.location = '';
95      });
96    },
97
98    reservar : function(idturno, idred) {
99      var msg = '<div class="dialog"><div id="admin_msg"></div><p>Desea reservar este turno?</
            p></div>';
100     $(msg).dialog( {
101       bgiframe : true,
102       minHeight : 120,
103       width : 340,
104       resizable : false,
105       modal : true,
106       title : 'Reservar',
107       buttons : {
108         'Ok' : function() {
```

```
109           if ($('#admin_msg').hasClass('error')) {
110             setTimeout("window.location.reload()");
111           } else {
112             $.post('/admin/usuario.php', {act : 'reservar', idturno : idturno, idred : idred
                    }, function(data) {
113               if (data.stderr.error) {
114                 var msg = '';
115                 $.each(data.stderr.info, function(i, value) {
116                   msg = msg + '<p>' + value + '</p>';
117                 });
118                 mensaje.error('#admin_msg', msg);
119               } else {
120                 setTimeout("window.location.reload()");
121               }
122             }, 'json');
123           }
124         }
125       }
126     });
127   },
128
129   cancelres : function() {
130     var msg = '<div class="dialog"><div id="admin_msg"></div><p>Desea cancelar la reserva?</
          p></div>';
131     $(msg).dialog( {
132       bgiframe : true,
133       minHeight : 120,
134       width : 340,
135       resizable : false,
136       modal : true,
137       title : 'Cancelar la reserva',
138       buttons : {
139         'Ok' : function() {
140           $.post('/admin/usuario.php', {act : 'cancelres'}, function(data) {
141             if (data.stderr.error) {
142               var msg = '';
143               $.each(data.stderr.info, function(i, value) {
144                 msg = msg + '<p>' + value + '</p>';
145               });
146               mensaje.error('#admin_msg', msg);
147             } else {
148               setTimeout("window.location.reload()");
149             }
150           }, 'json');
151         }
152       }
153     });
154   }
155 };
```

js/interfaz.js

```
1  $(document).ajaxStart($.blockUI).ajaxStop($.unblockUI);
2
3  // Aplicación
4  var aplicacion = {
5    // Función encargada de mostrar la interfaz gráfica de la aplicación
6    gui : function() {
```

```
7       aplicacion.listarConfiguraciones();
8       aplicacion.mostrarConfiguracion();
9       // Al hacer click en el menu principal aparecerán los formularios
10      // correspondientes
11      $('#navadmin').click(function() {
12        $('#config .title').html('Administrar');
13        $('#config h2').html('Administrar');
14        $('#config form:visible').hide();
15        $('#config .input_error, #config .input_info').remove();
16        $('#config input, #config select').removeClass('invalid');
17        $('#config #adminconfig').show();
18      });
19      $('#navload').click(function() {
20        $('#config .title').html('Cargar');
21        $('#config h2').html('Cargar');
22        $('#config form:visible').hide();
23        $('#config .input_error, #config .input_info').remove();
24        $('#config input, #config select').removeClass('invalid');
25        $('#config #loadconfig').show();
26      });
27      $('#navstore').click(function() {
28        $('#config .title').html('Guardar');
29        $('#config h2').html('Guardar');
30        $('#config form:visible').hide();
31        $('#config .input_error, #config .input_info').remove();
32        $('#config input, #config select').removeClass('invalid');
33        $('#config #storeconfig').show();
34      });
35      $('#navping').click(function() {
36        $('#config .title').html('Ping');
37        $('#config h2').html('Ping');
38        $('#config form:visible').hide();
39        $('#config .input_error, #config .input_info').remove();
40        $('#config input, #config select').removeClass('invalid');
41        $('#config #ping').show();
42      });
43      $('select[name="idconfig"]').change(function() {
44        $('#config .input_error, #config .input_info').remove();
45        $('#config input, #config select').removeClass('invalid');
46        $(this).parents('form').find('input[name="nombre"]').val('');
47        if ($(this).val() && $(this).val() != "new" && $(this).val() != "default") {
48          var config = eval('(' + unescape($.cookie('config' + $(this).val())) + ')');
49          $(this).after('<div class=input_info>\'--> Modificado: ' + config['modificado'].
                replace('+', ' a las ') + '</div>');
50          $(this).parents('form').find('input[name="nombre"]').val(config['nombre']);
51        }
52      });
53
54      // Información para el usuario del tiempo disponible antes de finalizar
55      // su turno
56      var turno_fin = $.cookie('turno_fin');
57      var fecha = new Date();
58      var fin = parseInt(turno_fin.substr(0, 2)) * 3600 + parseInt(turno_fin.substr(3, 2)) *
            60;
59      var ahora = fecha.getHours() * 3600 + fecha.getMinutes() * 60 + fecha.getSeconds();
60      $('#time').countdown({
61        until : (fin - ahora - 5 * 60),
62        onExpiry : function() {
63          $('#time').countdown('change',{
64            until : (5 * 60),
65            onExpiry : function() {$('#main_content').block({
```

```
66            message : '<div class="blocked"><h4>Su turno ha finalizado</h4><p>Ya no puede
                  acceder a la aplicación hasta su próximo turno. Los cambios no guardados han
                   sido descartados.</p><a href="" onclick="window.close(); return false;"><
                  button>OK</button></a></div>',
67            css : {width : '500px'}});
68          }
69        });
70        $('#main_content').block({
71          message : '<div class="blocked"><h4>Atención!: Restan 5 minutos para finalizar su
                  turno</h4><p>Se recomienda que proceda a almacenar los cambios realizados y
                  cierre la aplicación antes de que expire dicho tiempo.</p><a href="" onclick="
                  $(\'#main_content\').unblock(); return false;"><button>OK</button></a></div>',
72          css : {width : '500px'}});
73      },
74      compact : true
75    });
76  },
77
78
79  // Función que devuelve la hora actual
80  time : function() {
81    var time = new Date();
82    time = time.toTimeString().substring(0, 8);
83    return time;
84  },
85
86
87  // Función para listar las configuraciones existentes
88  listarConfiguraciones : function(formid) {
89    var lista = '';
90    if (document.cookie && document.cookie != '') {
91      var split = document.cookie.split(';');
92      for ( var i = 0; i < split.length; i++) {
93        var name_value = split[i].split("=");
94        name_value[0] = name_value[0].replace(/^ /, '');
95        if (name_value[0].substring(0, 6) == 'config') {
96          var config = eval('(' + unescape(decodeURIComponent(name_value[1])) + ')');
97          lista = lista + '<option value="' + config['idconfiguracion'] + '">' + config['
                  nombre'] + '</option>' + "\n";
98        }
99      }
100   }
101   $('#config .configslist').html(lista);
102 },
103
104
105 // Función encargada de mosrtrar la configuración actual de un elemento
106 mostrarConfiguracion : function() {
107   var idred = eval('(' + unescape($.cookie('idred')) + ')');
108   var svg_url = '/images/red' + idred + '.svg';
109   $('#diagrama').svg('destroy');
110   $('#diagrama').svg({
111     loadURL : svg_url,
112     onLoad : function() {
113       $('g').each(function() {
114         var nodo = true;
115         var id = $(this).attr('id').split(':');
116         var cid = id[0] + id[1];
117         if (id[0] == 'pc') {
118           var config = eval('(' + unescape($.cookie(cid)) + ')');
119           if (config) {
120             var msg = '<div class="tooltip"><h5>PC ' + id[1] + '</h5><div><ul>';
```

```
121               $.each(config, function(i,value) {
122                 msg = msg + '<li>' + i + ': ' + value + '</li>';
123               });
124               msg = msg + '</ul></div></div>';
125             } else {
126               var msg = '<div class="tooltip">Configuración no definida</div>';
127             }
128           } else if (id[0] == 'conmutador') {
129             var topologia = eval('(' + unescape($.cookie(cid)) + ')');
130             if (topologia) {
131               var msg = '<div class="tooltip"><h5>Conmutador ' + id[1] + '</h5><div><ul><li>
                      Modelo : ' + topologia.modelo + '</li>';
132               msg = msg + '<li><a href="" onclick="ventanas.abrir(\'/admin/conm_config.php?id
                      =' + id[1] + '\', \'Configuración\', \'location=no,toolbar=no,status=no\')
                      ; return false;">-> Mostrar archivo</a></li>';
133               msg = msg + '<li>Conexiones : ';
134               $.each(topologia.conexiones, function(puerto, nodo) {
135                 msg = msg + '<p>puerto ' + puerto + ' -> ' + nodo[0].toLowerCase() + ' ' +
                        nodo[1] + '</p>';
136               });
137               msg = msg + '</li></ul></div></div>';
138             } else {
139               var msg = '<div class="tooltip">Configuración no definida</div>';
140             }
141           } else {
142             nodo = false;
143           }
144           if (nodo) {
145             $(this).qtip({
146               content : msg,
147               position : {
148                 target : 'mouse',
149                 corner : {target : 'rightMiddle', tooltip : 'leftMiddle'}
150               },
151               style : {
152                 tip : 'leftMiddle',
153                 border : {width : 1, radius : 2, color : '#444'}
154               },
155               hide : {
156                 when : 'mouseout',
157                 fixed : true
158               }
159             });
160           }
161         });
162       }
163     });
164   },
165
166
167   // Función encargada de mosrtrar los formularios en el panel derecho
168   mostrarFormulario : function(nodo) {
169     var id = $(nodo).attr('id').split(':');
170     var elemento = id[0];
171     id = id[1];
172     $('#config form:visible').hide();
173     $('#config .input_error, #config .input_info').remove();
174     $('#config input, #config select').removeClass('invalid');
175     if (elemento == 'pc') {
176       $('#config .title').html('PC ' + id);
177       $('#config h2').html('PC ' + id);
178       if ($.cookie(elemento + id)) {
```

```
179         var config = eval('(' + unescape($.cookie('pc' + id)) + ')');
180         $('#config #pc input[name="id"]').attr('value', id);
181         $('#config #pc input[name="ip"]').attr('value', config.ip);
182         $('#config #pc input[name="mascara"]').attr('value', config.mascara);
183         $('#config #pc input[name="puerta"]').attr('value', config.puerta);
184         $('#config #pc input[name="dns1"]').attr('value', config.dns1);
185         $('#config #pc input[name="dns2"]').attr('value', config.dns2);
186         $('#config #pc').show();
187       }
188     } else if (elemento == 'conmutador') {
189       $('#config .title').html('Conmutador ' + id);
190       $('#config h2').html('Conmutador ' + id);
191       if ($.cookie(elemento + id)) {
192         $('#config #conmutador input[name="id"]').attr('value', id);
193         $('#config #conmutador').show();
194       }
195     }
196   },
197
198
199   // Función encargada de mosrtrar los errores producidos tras la última petición al
        servidor
200   mostrarErrores : function(data) {
201     if (data.stderr.error == 'validacion') {
202       var msg = '<p class="error_msg">>> (' + aplicacion.time() + ') El formulario contiene
            campos no validos</p>';
203       $.each(data.stderr.info, function(i, value) {
204         if (i == 'archivo') {
205           $('input[name=' + i + '], select[name=' + i + ']').addClass('invalid').after('<div
                class=input_error>\'--> Configuración no válida</div>');
206           msg = msg + '<br />';
207           $.each(value, function(i, error) {
208             msg = msg + '<p class="error_msg">' + error + '</p>';
209           });
210           msg = msg + '<br />';
211         } else {
212           $('input[name=' + i + '], select[name=' + i + ']').addClass('invalid').after('<div
                class=input_error>\'--> ' + value + '</div>');
213         }
214       });
215       $('#resultados').prepend(msg);
216     } else if (data.stderr.error == 'aplicacion') {
217       $.each(data.stderr.info, function(i, value) {
218         $('#resultados').prepend('<p class="error_msg">>> (' + aplicacion.time() + ') ' +
              value + '</p>');
219       });
220     } else {
221       var msg = '';
222       $.each(data.stderr.info, function(i, value) {
223         msg = msg + '<p>' + value + '</p>';
224       });
225       $('#main_content').block({
226         message : '<div class="blocked">' + msg + '<a href="" onclick="window.close();
              return false;"><button>OK</button></a></div>',
227         css : {
228           width : '500px',
229           border : '2px solid #FBC2C4',
230           background : '#FBE3E4',
231           color : '#8A1F11'
232         }
233       });
234     }
```

```
235     },
236
237
238     // Iniciar la aplicación
239     start : function() {
240       $.post('/admin/aplicacion.php', {act : 'start'}, function(data) {
241         if (data.stderr.error) {
242           var msg = '';
243           $.each(data.stderr.info, function(i, value) {
244             msg = msg + '<p>' + value + '</p>';
245           });
246           $('#main_content').block({
247             message : '<div class="blocked"><h4>Imposible iniciar!</h4>' + msg + '<a href=""
                    onclick="window.close(); return false;"><button>OK</button></a></div>',
248             css : {width : '500px'}
249           });
250         } else {
251           document.location = '/aplicacion/';
252         }
253       }, 'json');
254     },
255
256
257     // Finalizar la aplicación
258     end : function() {
259       $.post('/admin/aplicacion.php', {act : 'end'}, function(data) {
260         if (data.stderr.error) {
261           aplicacion.mostrarErrores(data);
262         } else {
263           document.location = '/aplicacion/';
264         }
265       }, 'json');
266     },
267
268
269     // Enviar configuración de un PC
270     setPC : function(config) {
271       $.post('/admin/aplicacion.php', config, function(data) {
272         if (data.stderr.error) {
273           aplicacion.mostrarErrores(data);
274         } else {
275           $('#resultados').prepend('<p>>> (' + aplicacion.time() + ') ' + data.stdout + '</p>'
                    );
276           // Se actualiza la ventana flotante que contiene la configuración del PC
277           aplicacion.mostrarConfiguracion();
278         }
279       }, 'json');
280     },
281
282
283     // Enviar configuración de un Conmutador
284     setConmutador : function() {
285       var data = eval("(" + $('#upload').contents().text() + ")");
286       if (data) {
287         if (data.stderr.error) {
288           aplicacion.mostrarErrores(data);
289         } else {
290           $('#resultados').prepend('<p>>> (' + aplicacion.time() + ') ' + data.stdout + '</p>'
                    );
291         }
292       }
293       $.unblockUI();
```

```
294   },
295
296
297   // Realizar ping a la dirección IP especificada
298   ping : function(info) {
299     $.post('/admin/aplicacion.php', info, function(data) {
300       if (data.stderr.error) {
301         aplicacion.mostrarErrores(data);
302       } else {
303         var msg = '<p>>> (' + aplicacion.time() + ') Resultado del comando ping:</p><br />';
304         $.each(data.stdout, function(i, value) {
305           msg = msg + '<p>' + value + '</p>';
306         });
307         msg = msg + '<br />';
308         $('#resultados').prepend(msg);
309       }
310     }, 'json');
311   },
312
313
314   // Guardar/Cargar/Borrar la configuración especificada
315   configs : function(inputs) {
316     $.post('/admin/aplicacion.php', inputs, function(data) {
317       if (data.stderr.error) {
318         aplicacion.mostrarErrores(data);
319       } else {
320         $('#resultados').prepend('<p>>> (' + aplicacion.time() + ') ' + data.stdout + '</p>'
                );
321         aplicacion.listarConfiguraciones();
322       }
323     }, 'json');
324   }
325 };
```

B.3. Módulo para el servidor XML-RPC

```
1  #!/usr/bin/python
2
3  # Modulos a importar
4  import SocketServer
5  from SimpleXMLRPCServer import SimpleXMLRPCServer,SimpleXMLRPCRequestHandler
6  import time
7  import sys
8  import os
9
10 # Ruta del archivo commonfunc.py
11 sys.path.insert(0, "/home/centro/scripts/cdcrpc")
12 import commonfunc
13 #definimos parametros
14 ficherologs = '/var/log/cdcrpc'
15 interfaz = 'eth0'
16
17 class Exportando:
18     """
19     Clase que contiene los metodos que se ofrecen como servicios remotos.
20     """
21     def config_red(self, ip, mascara, puerta, dns1, dns2):
```

```
22          """
23          Configura diferentes parametros de red en el servidor
24          """
25          err = []
26          comando = 'ifconfig ' + interfaz + ' ' + str(ip) + ' netmask ' + str(mascara)
27          salida, salidaerr = commonfunc.invoca(comando)
28          err = err + salidaerr
29      comando = 'route del default dev ' + interfaz
30          commonfunc.invoca(comando)
31      comando = 'route add default gw ' + str(puerta) + ' dev ' + interfaz
32          salida, salidaerr = commonfunc.invoca(comando)
33          err = err + salidaerr
34          comando = 'cat /dev/null > /etc/resolv.conf'
35          salida, salidaerr = commonfunc.invoca(comando)
36          err = err + salidaerr
37          comando = 'echo "nameserver ' + str(dns1) + '" >> /etc/resolv.conf'
38          salida, salidaerr = commonfunc.invoca(comando)
39          err = err + salidaerr
40          comando = 'echo "nameserver ' + str(dns2) + '" >> /etc/resolv.conf'
41          salida, salidaerr = commonfunc.invoca(comando)
42          err = err + salidaerr
43          return salida, err
44
45      def ping(self, destino):
46          """
47          Realiza un ping a la IP destino especificada
48          """
49          comando = 'ping -c 5 ' + str(destino)
50          salida, salidaerr = commonfunc.invoca(comando)
51          return salida, salidaerr
52
53
54  def initServer(host, Port, cerrojo, encoding, allowips):
55      """
56      Funcion que utiliza el metaservidor para lanzar el servicio.
57      """
58      commonfunc.inicia(host, Port, cerrojo, encoding, Exportando, ficherologs, allowips)
```