**Cambios Moodle (versión 3.5.2).**

En primer lugar, para instalar el plugin, hay que mover la carpeta con los ficheros del bloque al directorio /moodle/blocks y nombrarla como ‘tutor’. También habrá que añadir la siguiente línea en el fichero ‘version.php’:

*$plugin->component = 'block\_tutor'; // Full name of the plugin (used for diagnostics)*

Para añadir el bloque a un curso, hay que habilitar la edición de bloques de dicho curso y luego seleccionar ‘Agregar bloque’ y seleccionar ‘Tutor block’.

Hay que asegurarse de que cuando el agente introduzca a un alumno en un grupo para poder ver un recurso, si este actualiza la página, efectivamente se le permite acceder al recurso correspondiente. Si el agente añade al alumno al grupo necesario para ver un recurso pero actualizando la página del curso el usuario no consigue verlo, puede ser problema de la cache del curso, ya que cuando se modifica la base de datos directamente en vez de a través de moodle no se actualiza la cache.

Las caches se pueden eliminar manualmente por completo mediante Administración del sitio => Desarrollo => Purgar todas las caches.

La cache encargada de almacenar la información sobre los grupos a los que pertenece un usuario en un curso es ‘user\_group\_groupings’, por lo que cada vez que el agente introduzca un usuario en un grupo habrá que limpiarla para que ese cambio tenga efecto visible. Esta cache se puede purgar manualmente a través de Administración del sitio => Extensiones => Cache => Configuración => user\_group\_groupings – purgar.

Para que no haya que limpiarla manualmente tras cada inserción de un alumno en un grupo, se ha añadido un código al fichero /moodle/course/view.php que lo que hace es purgar la cache cada vez que el usuario carga a la página del curso para tener la última versión de los grupos a los que pertenece. Este código es el siguiente:

*$component = core*

*$area = user\_group\_groupings*

*cache\_helper::purge\_by\_definition($component, $area);*

La función checkTranslation() que se utiliza se explicará en el apartado donde se muestran las traducciones (7).

Para que el agente pueda crear los grupos utilizando los servicios web los pasos son los siguiente:

- Habilitar los servicios web y el protocolo que se vaya a utilizar (Administración del sitio => Extensiones => Servicios web => Habilitar servicios web/Habilitar los protocolos )

- Crear un usuario para utilizar los servicios web y matricularlo en el curso que se vaya a utilizar el bloque tutor.

- Crear un servicio externo con las funciones necesarias(core\_group\_add\_group\_members, core\_group\_create\_groups, core\_group\_delete\_grou\_memmbers, core\_group\_delete\_groups, core\_group\_get\_activity\_allowd\_groups, core\_group\_get\_group\_member, core\_group\_get\_groups) para crear los grupos e introducir alumnos autorizados (usuario creado en el punto anterior).

- Crear un rol en el sistema (Administración del sitio => Usuarios => Definir roles => Añadir un nuevo rol) con las capacidades necesarias para realizar las funciones indicadas y permitidiendo asignar el rol estudiante a las cuentas. (Administración del sitio => Usuarios => Definir roles => Permitir asignar roles)

- Asignarle al usuario creado el rol creado (Administración del sitio => Uusarios => Asignar roles globales).

- Crear un token para el usuario para el servicio. (Administración del sitio => Extensiones => Servicios web => Administrar tokens).

Para comprobar que funciona correctamente la creación de grupos mediante el servicio web se puede utilizar el siguiente código:

*package artifact;*

*/\**

*Main creado para comprobar el funcionamiento del servicio web creado para la creación de grupos de Moodle*

*\*/*

*public class Echo {*

*public static void main (String[] arg) {*

*WebServiceMoodle ws = new WebServiceMoodle();*

*try{*

*ws.cria\_grupos("grupo de prueba", 4);*

*System.out.println("Grupo creado");*

*}catch(Exception e){*

*System.out.println("criei grupo WS dentro da exception "+ e.getMessage()+ " para o curso "+ 4);*

*}*

*}*

*}*

Para que funcione el servicio web hay que cambiar el token y el domainName en el código de WebServiceMoodle.java

**Modificaciones realizadas**

**1)** Mostrar correctamente el árbol de dependencia.

En el fichero arvore.php hay que editar la creación del grafo, cambiando la línea:

*$graph = $graph.$var[0]."->".$var[1].";";*

por:

*if($var[0] != ""){*

*$var[0]=checkTranslation($var[0]);*

*$var[1]=checkTranslation($var[1]);*

*$graph = $graph.$var[0]."->".$var[1].";"; //Esta línea era la única que había en el original, hay que añadir las demás*

*}*

*else{*

*$graph = $graph.$var[1].";";*

*}*

**2)** Lanzar automáticamente con el servidor y **3)** Lanzar el agente solo cuando deba trabajar.

Para saber cuando se ha de lanzar el proyecto Jason y cuando se ha de detener, se han creado dos scripts: ‘prueba.sh’ y ‘prueba\_matar.sh’. Estos scripts se ejecutarán mediante el cron cada minuto comprobando la existencia de un registro específico(‘lanzar’ y ‘matar’, respectivamente) en la tabla mdl\_tutor\_perfil y guardando el resultado de su ejecución en un log.

El fichero ‘prueba.sh’ buscará el registro con nombre ‘lanzar’. En caso de que no lo encuentre finalizará la ejecución del script, pero si dicho registro existe, lanzará el proyecto Jason y borrará el registro ‘lanzar’ . (también borrara el contenido de la tabla mdl\_tutor\_bedel\_curso, ya que para que el proyecto funcione correctamente hay que eliminar el registro del curso correspondiente). El contenido del script prueba.sh es el siguiente:

*#!/bin/bash*

*STR=$(mysql -u root -ptinteligentes00 -D moodle -e "select id from mdl\_tutor\_perfil where nome='lanzar' ")*

*fecha=$(date)*

*if [ -z "$STR" ];*

*then*

*echo $fecha*

*echo "Aun no hay que lanzar jason\n"*

*else*

*echo $(date)*

*echo "\nEliminando registro lanzar...."*

*echo "Eliminando registro de mdl\_tutor\_bedel\_curso"*

*var0=$(mysql -u root -ptinteligentes00 -D moodle -e "delete from mdl\_tutor\_perfil where nome='lanzar' ")*

*var1=$(mysql -u root -ptinteligentes00 -D moodle -e "delete from mdl\_tutor\_bedel\_curso")*

*echo "Lanzando agentes......\n"*

*cd /home/raul/Downloads/Tutores/TutoresInteligentes-master/bin && ant*

*fi*

(hay que cambiar la ruta en la que se encuentra el proyecto Jason)

El fichero ‘prueba\_matar.sh’ buscará el registro con nombre ‘matar’. En caso de que no lo encuentre finalizará la ejecución del script, pero si dicho registro existe, lo eliminará de tabla y envíará una señal al proceso asociado a la ejecución del script de lanzamiento. El contenido del script ‘prueba\_matar.sh’ es el siguiente:

*#!/bin/bash*

*STR=$(mysql -u root -ptinteligentes00 -D moodle -e "select id from mdl\_tutor\_perfil where nome='matar' ")*

*fecha=$(date)*

*if [ -z "$STR" ];*

*then*

*echo $fecha*

*echo "Aun no hay que matar jason\n"*

*else*

*echo $(date)*

*echo "Eliminando registro lanzar...."*

*var0=$(mysql -u root -ptinteligentes00 -D moodle -e "delete from mdl\_tutor\_perfil where nome='matar' ")*

*echo "Matar agentes......"*

*var=$(ps -ef | grep jasonTutor.mas2j | awk '{print $2}' | head -n 1)*

*echo $var*

*kill -9 $var*

*fi*

El registro ‘lanzar’ que indica que el agente debe trabajar se inserta en el evento de finalización del intento de un cuestionario: la función quiz\_attempt\_submitted\_handler() del fichero /moodle/mod/quiz/locallib.php. Para ello se ha añadido el código:

*$record = new stdClass();*

*$record->id = 0;*

*$record->nome = 'lanzar';*

*$record->nota\_min = -1;*

*$record->nota\_max = -1;*

*$DB->insert\_record('tutor\_perfil', $record);*

El registro ‘matar’ que indica que el agente ha finalizado su trabajo se inserta en el método purge\_course\_cache() del fichero Bedel\_Atc.java del proyecto. El código añadido es el siguiente:

*String string2= “INSERT INTO mdl\_tutor\_perfil VALUES(-1, ‘matar’, -1, -1)”;*

*System.out.println(“Finalizando, avisar...” );*

*this.conexaoBD(“matar jason”);*

*this.update(string2);*

**4)** Que al alumno se le refresque automáticamente la página cuando esté el contenido disponible.

Para que no sea el usuario quién tenga que refrescar la página del curso cada vez que tenga un nuevo recurso disponible si no que esto se realice automáticamente se ha hecho uso de los Eventos de servidor. Se utilizará la tabla mdl\_tutor\_nota\_perfil para comprobar los refrescos que son necesarios.

Se ha creado el fichero see.php en el bloque que comprobará los registros de dicha tabla y recogerá aquellos con el número de cálculo (nro\_calculo) igual a 0, es decir, que todavía no han sido procesados. Por cada uno de ellos realizará un envío de un evento indicando el curso y el usuario que debe refrescar su página. Posteriormente, actualizará el número de cálculo a 1 para indicar que esos registros ya han sido procesados. El contenido del fichero see.php es:

require\_once('../../config.php');

require\_once("$CFG->libdir/resourcelib.php");

header('Content-Type: text/event-stream');

header('Cache-Control: no-cache');

function sendMsg($id, $id\_alumno, $id\_curso) {

echo "id: $id" . PHP\_EOL;

echo "data: {\n";

echo "data: \"id\": \"$id\", \n";

echo "data: \"id\_alumno\": \"$id\_alumno\", \n";

echo "data: \"id\_curso\": $id\_curso\n";

echo "data: }\n";

echo PHP\_EOL;

ob\_flush();

flush();

}

$startedAt = time();

do {

if ((time() - $startedAt) > 10) {

die();

}

$result = $DB->get\_records\_sql('SELECT \* FROM {tutor\_nota\_perfil} where nro\_calculo = 0');

foreach ($result as $res){

$id = $res->id;

$aluno\_id = $res->aluno\_id;

$curso\_id = $res->curso\_id;

sendMsg($id, $aluno\_id, $curso\_id);

$record = new stdClass();

$record->id = $id;

$record->nro\_calculo = 1;

$DB->update\_record('tutor\_nota\_perfil', $record);

}

sleep(5);

} while(true);

Cuando un cliente reciba dicho evento, comprobará si el id del usuario y del curso actual

coinciden con los recibidos, y en ese caso refrescará la página principal del curso. Esto lo hará con el siguiente cdódigo Javascript añadido en el fichero /moodle/course/view.php

if(typeof(EventSource) !== "undefined") {

var curso = '<?php echo $id;?>';

var alumno = '<?php echo $USER->id;?>';

var source = new EventSource("../blocks/tutor/see.php");

source.onmessage = function(event) {

var datos = JSON.parse(event.data);

if(datos.id\_curso == curso && datos.id\_alumno == alumno ){

location.href ="http://localhost/moodle/course/view.php?id=" + curso;

return;

}

};

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support server-sent events...";

}

Para indicar que un alumno ha realizado un cuestionario y hay que refrescar su página se insertará un registro en la tabla indicando el id del curso y el alumno con nro\_calculo igual a -1 (para que haya únicamente uno por cada alumno) en el método insere\_mensagem() del fichero Tutor\_Atc.java del proyecto. El código añadido es el siguiente:

*String string2= “INSERT INTO mdl\_tutor\_nota\_perfil(id, curso\_id, id\_grade\_item, aluno\_id, nota, nro\_calculo) VALUES(”+datahoje+ “, ” + curso + “, 0 , “+ id + “, 0, -1 )”;*

*System.out.println(“Finalizando, avisar...” );*

*this.conexaoBD(“matar jason”);*

*this.update(string2);*

Cuando el agente bedel encargado de la gestión de los cursos haya finalizado su trabajo con todos los alumno cambiará el nro\_calculo de los registros con valor -1 a valor 0 para indicar que tienen que ser procesados. Esto se realiza en el método purge\_course\_cache() del fichero Bedel\_Atc.java del proyecto con el siguiente código:

*String string3= “UPDATE mdl\_tutor\_nota\_perfil SET nro\_calculo=0 where nro\_calculo=-1”;*

*System.out.println(“Refrescos necesarios, avisar...” );*

*this.conexaoBD(“refrescar”);*

*this.update(string3);*

**5)** Que al alumno no visualice el bloque tutor ni contenidos ocultos (solo el inicial).

Para que el alumno no puede ver el bloque hay que restringirle el acceso mediante Bloque tutor => Permisos => Ver bloque y eliminar el permiso de los alumnos.

Haciendo solo esto, el alumno no podría acceder mediante la página del curso a la configuración del bloque, pero si que podría mediante la URL. Para evitar esto, se ha añadido un fragmento de código en todos los ficheros del bloque que comprueba si quién intenta acceder a la página está matriculado únicamente como alumno o tiene más roles. Si tiene más roles se le permite el acceso, pero si solo está matriculado como alumno se le redirigirá a la página del curso mostrándole un mensaje. El código utilizado para esto es el siguiente:

*$context = get\_context\_instance(CONTEXT\_COURSE,$course->id);*

*$can\_access=false;*

*if ($roles = get\_user\_roles($context, $USER->id)) {*

*foreach ($roles as $role) {*

*if( $role->roleid != 5){ //El 5 es el id del rol estudiante*

*$can\_access=true;*

*}*

*}*

*}*

*if(!$can\_access){*

*$message= "No puedes acceder a esta página. Se te redirigirá de nuevo a la página del curso en 5 segundos.";*

*redirect("$CFG->wwwroot/course/view.php?id=$course->id", $message, 5) ;*

*}else{*

*//Ejecución del código*

*}*

**6)** La nota está puesta siempre sobre 100.

En el fichero de configuración del bloque tutor ‘tutor\_form.php’ se permitirá modificar las notas mínima y máxima de cada uno de los perfiles (Básico, Medio y Avanzado), teniendo que añadir dos nuevos campos int: ‘nota\_min’ y ‘nota\_max’ en la tabla mdl\_tutor\_perfil. Ahora se podrán modifcar estas notas mediante el siguiente formulario en forma de tabla:

*<form name="guardar" method="post" action="<?php echo "tutor\_form.php?id=".$id."&amp;acao=guardar"?>" value="guardar">*

*<table>*

*<tr>*

*<td></td>*

*<td style="text-align:center;"><?php echo $mensajes[7] ?></td>*

*<td style="text-align:center;"><?php echo $mensajes[8] ?></td>*

*</tr>*

*<tr>*

*<td><b><?php echo $mensajes[9] ?> </b> </td>*

*<td><input type="number" id ="basico\_minima" step="0.1" placeholder="<?php echo $basico->nota\_min/10; ?>" name="basico\_minima"></td>*

*<td><input type="number" id ="basico\_maxima" step="0.1" placeholder="<?php echo $basico->nota\_max/10; ?>" name="basico\_maxima"></td>*

*</tr>*

*<tr>*

*<td><b><?php echo $mensajes[10] ?> </b> </td>*

*<td><input type="number" id ="medio\_minima" step="0.1" placeholder="<?php echo $medio->nota\_min/10; ?>" name="medio\_minima"></td>*

*<td><input type="number" id ="medio\_maxima" step="0.1" placeholder="<?php echo $medio->nota\_max/10; ?>" name="medio\_maxima"></td>*

*</tr>*

*<tr>*

*<td><b><?php echo $mensajes[11] ?> &nbsp;</b> </td>*

*<td><input type="number" id ="avanzado\_minima" step="0.1" placeholder="<?php echo $avanzado->nota\_min/10; ?>" name="avanzado\_minima"></td>*

*<td><input type="number" id ="avanzado\_maxima" value="10" disabled></td>*

*</tr>*

*</table>*

*</br><input type="button" value="<?php echo $mensajes[13]?>" onClick="GuardarValores()"/>*

*</form>*

Para rellenarlo, se utilizarán los valores obtenidos de la base de datos mediante:

$*basico = $DB->get\_record('tutor\_perfil', array('id'=>'1'));*

*$medio = $DB->get\_record('tutor\_perfil', array('id'=>'2'));*

*$avanzado = $DB->get\_record('tutor\_perfil', array('id'=>'3'));*

Para poder modificar las notas en la base de datos se utilizará el script GuardarValores() que se ejecutará tras pulsar el botón del formulario. El script es este:

var checkBox = document.getElementById("myCheck");

var basicom = document.getElementById("basico\_minima");

var basicoM = document.getElementById("basico\_maxima");

var mediom = document.getElementById("medio\_minima");

var medioM = document.getElementById("medio\_maxima");

var avanzadom = document.getElementById("avanzado\_minima");

var avanzadoM = document.getElementById("avanzado\_maxima");

if( basicom.value.length == 0 || basicoM.value.length == 0 ||

mediom.value.length == 0 || medioM.value.length == 0 ||

avanzadom.value.lenth == 0){

window.alert("Rellena todos los campos para cambiar los valores.\nAhora se están mostrando los actuales");

return;

}

if (checkBox.checked == true){

var basico\_minima = parseInt(basicom.value);

var basico\_maxima = parseInt(basicoM.value);

var medio\_minima = parseInt(mediom.value);

var medio\_maxima = parseInt(medioM.value);

var avanzado\_minima = parseInt(avanzadom.value);

var avanzado\_maxima = parseInt(avanzadoM.value);

if(basico\_minima < 0 || basico\_minima > 100 || basico\_maxima < 0 || basico\_maxima > 100 || medio\_minima < 0 || medio\_minima > 100 || medio\_maxima < 0 || medio\_maxima > 100 ||avanzado\_minima < 0 || avanzado\_minima > 100){

window.alert("Los valores deben estar entre 0 y 100");

return;

}

}else {

var basico\_minima = parseFloat(basicom.value);

var basico\_maxima = parseFloat(basicoM.value);

var medio\_minima = parseFloat(mediom.value);

var medio\_maxima = parseFloat(medioM.value);

var avanzado\_minima = parseFloat(avanzadom.value);

var avanzado\_maxima = parseFloat(avanzadoM.value);

if(basico\_minima < 0 || basico\_minima > 10 || basico\_maxima < 0 || basico\_maxima > 10 || medio\_minima < 0 || medio\_minima > 10 || medio\_maxima < 0 || medio\_maxima > 10 ||avanzado\_minima < 0 || avanzado\_minima > 10){

window.alert("Los valores deben estar entre 0 y 10");

return;

}

}

//La nota minima de los perfiles debe ser inferior a la maxima

if(basico\_minima >= basico\_maxima ){

window.alert("El rango del perfil Basico no es valido, revísalo. ");

}

else if(medio\_minima >= medio\_maxima){

window.alert("El rango del perfil Medio no es valido, revísalo. ");

}

else if(avanzado\_minima >= avanzado\_maxima){

window.alert("El rango del perfil Avanzado no es valido, revísalo. ");

}

//La nota maxima de los perfiles debe ser menor que la minima de los siguientes

else if(basico\_maxima != medio\_minima){

window.alert("La maxima del basico y la minima del medio deben ser iguales.");

}

else if(medio\_maxima != avanzado\_minima){

window.alert("La maxima del medio y la minima del avanzado deben ser iguales.");

}else {

var c = window.confirm("Todo OK. Confirma para guardar los nuevos valores.");

if(c== true){

document.guardar.action="tutor\_form.php?id=<?php echo $id ?>&acao=guardar";

document.forms.guardar.submit();

}

}

Este script comprobará la validez de los datos y en caso de que esten correctos llamará de nuevo al fichero form.php con el parámetro ‘acao’=guardar.

Cuando se detecte este parámetro, se ejecutará el siguiente fragemento de código, que permite actualizar los registros de cada uno de los perfiles en la base de datos.

if ($acao == 'guardar'){

if (isset($\_POST['basico\_minima']) && isset($\_POST['basico\_maxima'])){

$record = new stdClass();

$record->id = 1;

$record->nome = 'Básico';

if($\_POST['basico\_minima'] < 10){

$\_POST['basico\_minima'] = $\_POST['basico\_minima'] \* 10;

}

$record->nota\_min = $\_POST['basico\_minima'];

if($\_POST['basico\_maxima'] < 10){

$\_POST['basico\_maxima'] = $\_POST['basico\_maxima'] \* 10;

}

$record->nota\_max = $\_POST['basico\_maxima'];

$DB->update\_record('tutor\_perfil', $record);

}

if (isset($\_POST['medio\_minima']) && isset($\_POST['medio\_maxima'])){

$record = new stdClass();

$record->id = 2;

$record->nome = 'Médio';

if($\_POST['medio\_minima'] < 10){

$\_POST['medio\_minima'] = $\_POST['medio\_minima'] \* 10;

}

$record->nota\_min = $\_POST['medio\_minima'];

if ($\_POST['medio\_maxima'] < 10){

$\_POST['medio\_maxima'] = $\_POST['medio\_maxima'] \* 10;

}

$record->nota\_max = $\_POST['medio\_maxima'];

$DB->update\_record('tutor\_perfil', $record);

}

if (isset($\_POST['avanzado\_minima']) ){

$record = new stdClass();

$record->id = 3;

$record->nome = 'Avançado';

if($\_POST['avanzado\_minima'] < 10){

$\_POST['avanzado\_minima'] = $\_POST['avanzado\_minima'] \* 10;

}

$record->nota\_min = $\_POST['avanzado\_minima'];

$DB->update\_record('tutor\_perfil', $record);

}

}

Para que estas modificaciones tengan efecto, hay que modificar parte del código en el fichero Bedel\_Atc.java del proyecto, para que en vez de calcular los perfiles en el método calcula\_perfil\_nota\_aluno() con notas estáticas lo haga con aquellas almacenadas en la base de datos.

ArrayList<String> basico = newArrayList<>();

ArrayList<String> medio = newArrayList<>();

ArrayList<String> avanzado = newArrayList<>();

try{

basico = get\_notas\_perfil(**"**Básico**"**);

medio = get\_notas\_perfil(**"**Médio**"**);

avanzado = get\_notas\_perfil(**"**Avançado**"**);

} catch(SQLException e) {

System.*out*.println(**"**ExcepcionSQL**"**);

} catch(ClassNotFoundException e) {

System.*out*.println(**"**ExcepcionClassNotFoundException**"**);

}

{……………}

if (media > Integer.*parseInt*(basico.get(0)) && media <= Integer.*parseInt*(basico.get(1))) {

id\_perfil = "1";

} else if (media > Integer.*parseInt*(medio.get(0)) && media < Integer.*parseInt*(medio.get(1))) {

id\_perfil = "2";

} else if (media >= Integer.*parseInt*(avanzado.get(0))) {

id\_perfil = "3";

}

El método get\_notas\_perfil() es el encargado de obtener las notas de la base de datos y almacenarlas en el perfil correspondiente. El método contiene el siguiente código:

String string = "SELECT nome, nota\_min, nota\_max FROM mdl\_tutor\_perfil where nome='" + nome\_perfil + "'";

ArrayList<String> notas = new ArrayList<>();

try {

ResultSet rs = bedel.select(string);

while (rs.next()) {

notas.add(rs.getString("nota\_min"));

notas.add(rs.getString("nota\_max"));

}

} catch (SQLException e) {

System.*out*.println("Excepcion SQL");

} catch (ClassNotFoundException e) {

System.*out*.println("Excepcion ClassNotFound");

}

if (Integer.*parseInt*(notas.get(0)) > Integer.*parseInt*(notas.get(1))) {

System.*out*.println("El rango no es valido: " + Integer.*parseInt*(notas.get(0)) + "~" + Integer.*parseInt*(notas.get(1)));

String temp = notas.get(0);

notas.set(0, notas.get(1));

notas.set(1, temp);

System.*out*.println("Cambio: " + Integer.*parseInt*(notas.get(0)) + "~" + Integer.*parseInt*(notas.get(1)));

}

return notas;

Se ha añadido la opción de poder ver y modificar las notas sobre un total de 100 o un total de 10. Para ello se ha creado el script cambiar() que actualizará los step y los valores de los diferentes campos number del formulario cuando se marque o se desmarque el chechbox:

*<input type="checkbox" id="myCheck" onclick="cambiar()"><?php echo $mensajes[5] ?></br>*

El script cambiar() es este:

var checkBox = document.getElementById("myCheck");

var basicom = document.getElementById("basico\_minima");

var basicoM = document.getElementById("basico\_maxima");

var mediom = document.getElementById("medio\_minima");

var medioM = document.getElementById("medio\_maxima");

var avanzadom = document.getElementById("avanzado\_minima");

var avanzadoM = document.getElementById("avanzado\_maxima");

if (checkBox.checked == true){

var basico\_minima = parseFloat(basicom.value);

var basico\_maxima = parseFloat(basicoM.value);

var medio\_minima = parseFloat(mediom.value);

var medio\_maxima = parseFloat(medioM.value);

var avanzado\_minima = parseFloat(avanzadom.value);

var avanzado\_maxima = parseFloat(avanzadoM.value);

basicom.value = parseInt(basico\_minima\*10);

basicoM.value = parseInt(basico\_maxima\*10);

mediom.value = parseInt(medio\_minima\*10);

medioM.value = parseInt(medio\_maxima\*10);

avanzadom.value = parseInt(avanzado\_minima\*10);

avanzadoM.value = parseInt(avanzado\_maxima\*10);

basicom.placeholder = basicom.placeholder \* 10;

basicoM.placeholder = basicoM.placeholder \* 10;

mediom.placeholder = mediom.placeholder \* 10;

medioM.placeholder = medioM.placeholder \* 10;

avanzadom.placeholder = avanzadom.placeholder \* 10;

basicom.step = 1;

basicoM.step = 1;

mediom.step = 1;

medioM.step = 1;

avanzadom.step = 1;

avanzadoM.step = 1;

}else {

var basico\_minima = parseInt(basicom.value);

var basico\_maxima = parseInt(basicoM.value);

var medio\_minima = parseInt(mediom.value);

var medio\_maxima = parseInt(medioM.value);

var avanzado\_minima = parseInt(avanzadom.value);

var avanzado\_maxima = parseInt(avanzadoM.value);

basicom.value=basico\_minima/10;

basicoM.value=basico\_maxima/10;

mediom.value=medio\_minima/10;

medioM.value=medio\_maxima/10;

avanzadom.value=avanzado\_minima/10;

avanzadoM.value=avanzado\_maxima/10;

basicom.placeholder = basicom.placeholder / 10;

basicoM.placeholder = basicoM.placeholder / 10;

mediom.placeholder = mediom.placeholder / 10;

medioM.placeholder = medioM.placeholder / 10;

avanzadom.placeholder = avanzadom.placeholder / 10;

basicom.step=0.1;

basicoM.step=0.1;

mediom.step=0.1;

medioM.step=0.1;

avanzadom.step=0.1;

avanzadoM.step=0.1;

}

**7)** Traducciones.

Se han diferenciado dos tipos de traducciones: De mensajes que se imprimian directamente en los ficheros del bloque como html y del contenido de Moodle.

Para realizar las traducciones del contenido de Moodle hay que descargar el plugin filter-multilang2, descomprimirlo en la carpeta filter de moodle e instalarlo desde el navegador. Luego hay que acceder a Administración del sitio => Extensiones => Vista general de extensiones => Editar filtros => habilitar filtro multilang. Hay que habilitarlo tanto para contenido como para cabeceras, para poder así permitir la traducción de los nombres de los cuestionarios, por ejemplo. Las traducciones se harán de la siguiente forma:

*{mlang XX}contenido en lenguaje XX{mlang}*

*{mlang YY}contenido en lenguaje YY{mlang}*

Tras realizar esto, si traducimos los nombres de los cuestionarios que forman parte del árbol de dependencia del curso, este se muestra mal. Para solucionarlo se ha creado la función checkTranslation(), que se ha incluido en un nuevo fichero ‘lib.php’ creado en la carpeta del bloque. Para utilizar este fichero en los demás del bloque hay que añadirles la siguiente línea:

*require\_once($CFG->dirroot . '/blocks/tutor/lib.php');*

*La* función checkTranslation() que se ha utiliado a lo largo del documento, se queda con el primer nombre introducido en la traducción, suponiendo que el primero introducido siempre será en el idioma principal del curso, para que de esta forma sea este el que se muestre en el árbol de dependencia. La función es:

*$findme = '{mlang}';*

*$pos = strpos($var, $findme);*

*if ($pos !== false) {*

*$cad= explode("}", $var);*

*$res = explode("{", $cad[1]);*

*return $res[0];*

*}*

*else{*

*return $var;*

*}*