El gráfico anterior tiene en cuenta el coste, pero como cada perfil tiene un precio distinto por hora, no queda del todo claro la dedicación total. Gráficamente, la distribución del número de horas entre los diferentes perfiles es la siguiente:

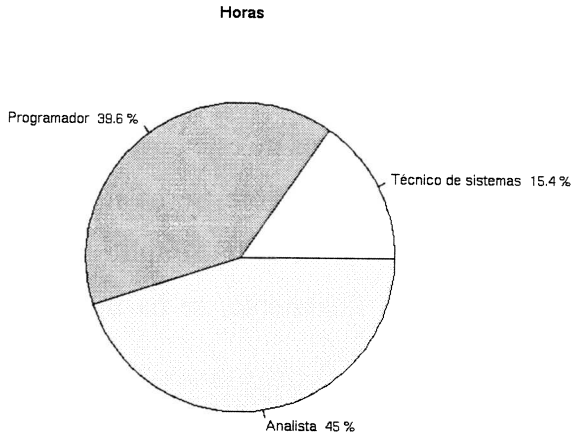

Figura 98. Distribución de las horas requeridas por el proyecto entre los diferentes perfiles

Aquí sí se aprecia claramente como el proyecto es principalmente el diseño y la programación de la nueva versión de e-status. Las horas del técnico de sistemas son únicamente el 15% del total de horas del proyecto, en concordancia con los objetivos de la parte de sistemas que son secundarios.

15. Conclusiones

Se puede decir que los múltiples objetivos de este proyecto han sido cumplidos totalmente. Se ha convertido todo el sistema a software libre, se ha remodelado la interfaz y ampliado las funcionalidades de la aplicación e-status, se ha simplificado la arquitectura, se han incluido nuevos servicios en el servidor, y finalmente, se ha mejorado la seguridad del sistema en general.

El sistema actualmente se trata de una plataforma completamente OpenSource. Se desea que en el futuro otras instituciones puedan usar este software, para ello entre otras tareas ha sido necesario reemplazar el sistema gestor de bases de datos y facilitar la instalación de los servicios.

Se han convertido todos los datos de la bases de datos en producción, Microsoft SQL Server bajo Windows, a un nuevo servidor MySQL bajo Linux. La conversión no ha sido una tarea sencilla, ya que la estructura de las tablas se ha cambiado por completo, y algunos campos no existían en la anterior. Tras la conversión, toda la información de asignaturas de la versión anterior de e-status están en la nueva versión y las estadísticas son consistentes entre ambas versiones.

La definición de los problemas de la versión anterior estaban en dos sistemas distintos, en el viejo lenguaje EST y en el nuevo lenguaje R. Como el antiguo motor EST estaba en Java y era muy poco potente, se ha desechado completamente. Por ello, los problemas definidos en el EST no pueden ser ejecutados en la nueva versión, pero en cambio los problemas que se han ido diseñando en R estos últimos meses en la versión anterior, son totalmente compatibles con e-status 2.0.

Para facilitar la instalación de los servicios de e-status, se ha integrado un sistema de paquetes CVS - RPM. Este sistema gestiona los ficheros de configuración del sistema que son necesarios modificar para tener un servidor e-status operativo y facilita la instalación de otros componentes, como la librería personalizada de R de e-status.

Para mejorar la seguridad, se ha diseñado un firewall local configurable de forma sencilla para los 3 servidores de e-status y se ha establecido un sistema de copias de seguridad automatizado en horas nocturnas. El sistema de backups se integra con el sistema de copias de la UPC a través de un servidor intermedio, gracias a la colaboración del responsable de sistemas del departamento de EIO, Toni Font.

La interfaz de la aplicación e-status ahora es más atractiva para el usuario. Los alumnos disponen de un historial mejorado, en el que se pueden comparar con sus compañeros para ver su progresión respecto el resto. Los profesores también disponen de nuevas herramientas como búsquedas más potentes, información contextual en el editor o visualizar ejecuciones pasadas de alumnos para revisar su corrección.

Este proyecto ha necesitado sincronizarse con el del proyectista Daniel Carrillo Carrillo. La programación paralela de ambos proyectos se ha realizado mediante CVS, diseñado en el entorno de desarrollo del proyecto de José María Cerdeira. Inicialmente el uso de esta herramienta añade complejidad al proceso de programación y pruebas, pero con un poco de uso se ha hecho bastante práctico y rápido.

Gracias al proyecto de David Rodríguez, se integró R a través de Rserve en el motor de problemas de e-status. Esta integración se ha explotado notablemente con el nuevo motor de problemas y la introducción de las gráficas de las estadísticas en lenguaje R. Como yo no conocía R, tuve que aprender a programar en este lenguaje. No solo me ha servido para llevar a cabo con éxito este proyecto, sino también me ha sido de utilidad para realizar gráficos para proyectos de otras asignaturas de último curso.

Para futuros desarrollos en sistemas, podría estudiarse el uso de máquinas virtuales para los servidores de e-status. Las máquinas virtuales proporcionan compatibilidad directa entre diferentes plataformas, y facilitan enormemente la instalación de un nuevo servidor: todo el sistema operativo está en un único fichero.

Otro tema a investigar sería un sistema de monitorización como Nagios. Un servidor auxiliar comprobaría regularmente que todos los servicios de e-status se encuentran en marcha, y si alguno fallara, enviaría un correo electrónico al encargado de sistemas para que tome las medidas oportunas.

Finalmente, se podría analizar la posibilidad de generar un único fichero RPM que instale la totalidad de los servicios que requiere e-status para facilitar aún más la configuración del servidor.

Por último, he de decir que me ha parecido un proyecto muy interesante. La gran diversidad de objetivos ha hecho que ponga en práctica los conocimientos de multitud de asignaturas de toda la carrera, como estadística, sistemas operativos, bases de datos, ingeniería del software, programación y seguridad.

16. Bibliografía

16.1 e-status

- Cerdeira Rodríguez, José María. *e-Status bajo Linux.*
  Director: José Antonio González Alastrue.

- Rodríguez González, David. *Integración de R en e-status.*
  Director: José Antonio González Alastrue.

16.2 PHP

- Trigos García, Esteban. *PHP 4.*
  Anaya Multimedia.

- The PHP Group. Manual de PHP.
  http://www.php.net/manual/es/

16.3 CakePHP

- Cake Software Foundation. The CakePHP Manual.
  http://manual.cakephp.org/

16.4 Linux

- Marín Pérez, César; Pérez Crespo, Ismael. *Linux.*
  Anaya Multimedia.

- Wirzenius, Lars; Oja, Joanna; Stafford, Stephen; Weeks, Alex. *The Linux System Administrator's Guide*

- Anónimo. *Linux Máxima Seguridad.*
  Prentice Hall.

16.5 R

- Aluja, Tomàs. *Introducció a R.*

- Paradis, Emmanuel. *R para Principiantes.*

- Correa, Juan Carlos; González, Nelfi. *Gráficos estadísticos con R.*
  Universidad Nacional – Sede Medellín.

- R Development Core Team. *R Installation and Administration.*

- R Development Core Team. *R Data Import / Export.*

- R Development Core Team. *Writing R Extensions.*

- R Development Core Team. *An Introduction to R.*

- R Development Core Team. *R Language Definition*

- A. James, David. *The RMySQL Package.*

17. Agradecimientos

- Jose Antonio González Alastrue: por darme la posibilidad de realizar este proyecto y su ayuda durante el mismo.

- Toni Font Font: por su asistencia en la instalación de los servidores y servicios de e-status, como la gestión de las copias de seguridad.

- José María Cerdeira: por su colaboración cuando yo estaba de becario en el proyecto e-status.

- David Rodríguez González: por su contribución al integrar R en e-status.

- Daniel Carrillo Carrillo: por su cooperación en la nueva versión de e-status.

18. Apéndice 1: Problema complejo

Con el fin de ilustrar las nuevas características del motor de problemas, se ha creado un problema artificial que muestra prácticamente la totalidad de ellas.

Datos de problema

Nombre: Preguntas dinámicas

Descripción: Demo de preguntas dinámicas

Enunciado:

```
Preguntas dinámicas sobre una N($mu$,$sg$) y sobre una binomial($n$,$p$)
Gráfica de N($mu$,$sg$)
$grafi$
```

Expresiones:

```
mu=round(runif(1,1,100),0)
sg=round(runif(1,1,10),0)
x=seq(mu-3.5*sg,mu+3.5*sg,len=100)
y=dnorm(x,m=mu,sd=sg)
grafi=ini_imagen()
plot(x,y,type="l")
fin_imagen()
#Pregunta 1
#Entra un número que esté entre el percentil p1
#y el percentil p2 de una N(mu,sg) { qnorm(p1) <= resp1 <= qnorm(p2)}
p1<-round(runif(1,1,50),0)
p2<-round(runif(1,50,99),0)

z1<-qnorm( p1 / 100, mean=mu,sd=sg)
z2<-qnorm( p2 / 100, mean=mu,sd=sg)
z=c(z1,z2)
####################
#Pregunta 2
#si X es binomial($n$,$p$), entra un intervalo de valores que acumule
#una prob. mayor que $p_ini$ pero menor que $p_fin$ { $p_ini$<F(_q[2])-
F(_q[1]-1)< $p_fin$}
n<-sample(5:15,1)
p<-round(runif(1,0.1,1.9),1)/2

v<-sample(0:(n - 3), 2)
v1<-min(v)
v2<-max(v) + 2
v=c(v1,v2)

p_ini=round(pbinom(v2,n,p) - pbinom(v1,n,p),3 ) - 0.001;
p_fin=round(pbinom(v2+1,n,p) - pbinom(v1,n,p),3 ) - 0.005;
```

Pregunta 1

Esta pregunta tiene más de una respuesta correcta posible, ya que se comprueba que esté dentro de un rango de respuestas correctas. El rango se ha definido en las expresiones del enunciado del problema (valores z1 y z2), y se ha de comprobar la respuesta mediante un algoritmo de corrección complejo.

Además, en caso de ser incorrecta la respuesta, se genera un gráfico que indica el valor introducido, y el rango de valores correctos. Las nuevas variables creadas en las expresiones de respuesta pueden ser utilizadas en Mensaje correcto/regular/incorrecto, que se muestran en caso de que el resultado sea 1, (0,1) y 0, respectivamente. En este caso, se muestra la respuesta de nuevo (*respuesta_*), y el nuevo gráfico generado (*grafi_resp*).

Enunciado:

```
Entra un número que esté entre el percentil $p1$ y el percentil $p2$
```

Solución respuesta: *(vacío)*

Parámetro sintaxis: *(vacío)*

Sintaxis respuesta: Real

Plantilla exp. respuesta: *(vacío)*

Parámetro expresiones: *(vacío)*

Expresiones respuesta:

```
if (respuesta_ >=z1 && respuesta_ <= z2)
{
    resultado_ <- 1;
}else{
    resultado_ <- 0;
    #incorrecto se ha de generar el grafico marcando la respuesta del
usuario

    grafi_resp=ini_imagen(300,300)
    plot(x,y,type="l")

    #respuesta del usuario
    temp_x<-c(respuesta_,respuesta_)
    temp_y<-c(0,dnorm(respuesta_,mean=mu,sd=sg))
    lines(temp_x,temp_y,col="red")

    #rango ok
    temp_x<-c(z1,z1)
    temp_y<-c(0,dnorm(z1,mean=mu,sd=sg))
    lines(temp_x,temp_y,col="green")

    temp_x<-c(z2,z2)
    temp_y<-c(0,dnorm(z2,mean=mu,sd=sg))
```

```
    lines(temp_x,temp_y,col="green")

    fin_imagen()

}
```

Peso: 1

Intentos: 2

Mostrar solución: *(desmarcado)*

Mensaje correcto: *(vacío)*

Mensaje regular: *(vacío)*

Mensaje incorrecto:

```
El siguiente gráfico muestra su respuesta ($respuesta_$) en rojo. El rango
de posibles soluciones está delimitado por líneas verdes.

$grafi_resp$
```

Pregunta 2

Esta pregunta está encadenada con la pregunta anterior. El valor correcto de la respuesta va en función de la respuesta de la pregunta anterior. Para acceder a la respuesta anterior, se utiliza el código *respuestas_[[num_pregunta_ - 1]]*.

Como la respuesta anterior puede ser correcta o incorrecta, se puntuará la mitad en esta pregunta si el p-valor es correcto, pero la respuesta anterior era incorrecta, y la totalidad en caso de ser el p-valor correcto y la respuesta anterior correcta también. El resultado de la pregunta anterior se accede mediante *resultados_[num_pregunta_ - 1]*.
La respuesta es una probabilidad, y se desean 3 decimales correctos. El algoritmo de corrección se basa en la plantilla "Número de decimales correctos", aplicando los cambios indicados anteriormente.

Enunciado:

```
¿Cual es el p-valor de la respuesta anterior?
```

Solución respuesta: *(vacío)*

Parámetro sintaxis: *(vacío)*

Sintaxis respuesta: Probabilidad

Plantilla exp. respuesta: *(vacío)*

Parámetro expresiones: 3

Expresiones respuesta:

```
resp_anterior <- respuestas_[[num_pregunta_ - 1]];
solucion_ <- pnorm(resp_anterior ,mu,sg)

if(estatus.decimales_correctos(solucion_,respuesta_,parametro_))
{
    if (resultados_[num_pregunta_ - 1] != 1){     resultado_ <- '0.5'; }
    else {  resultado_<-'1'; }
}else{
     resultado_<-'0'
}
```

Peso: 1

Ayuda contextual: Entre $param_expresiones_$ decimales correctos

Intentos: 1

Mostrar solución: *(desmarcado)*

Mensaje correcto: *(vacío)*

Mensaje regular: Puntúa la mitad ya que es correcta pero la respuesta anterior no lo es.

Mensaje incorrecto: ¡Incorrecto! El p-valor de $resp_anterior$ es $solucion_$

Pregunta 3

Esta pregunta está relacionada con la siguiente. El alumno debe introducir un intervalo de valores, indicando el extremo inferior en esta pregunta, y el superior en la siguiente. Para corregir esta pregunta, se debe obtener la respuesta de esta pregunta (*respuesta_*) y la siguiente (*respuestas_[[num_pregunta_ + 1]]*), y comprobar si el intervalo formado acumula la probabilidad que indica el enunciado (valores *p_ini* y *p_fin* definidos en las expresiones del problema).

En caso de que el alumno fallara los 2 intentos, al estar marcado "mostrar solución", se mostrará la solución indicada (*v*). Esta solución ha sido generada en las expresiones del problema, y se trata de una solución posible (pero no tiene porqué ser la única), que se muestra como ejemplo.

Enunciado:

```
Si X es binomial($n$,$p$), entra un intervalo de valores que acumule una
prob. mayor que $p_ini$ pero menor que $p_fin$.
Extremo inferior
```

Solución respuesta: *v*

Parámetro sintaxis: *(vacío)*

Sintaxis respuesta: Natural

Plantilla exp. respuesta: *(vacío)*

Parámetro expresiones: *(vacío)*

Expresiones respuesta:

```
extremo_inf <- respuesta_
extremo_sup <- respuestas_[[ num_pregunta_ + 1 ]]

if (extremo_inf > extremo_sup)
{
    mensaje_ <- 'El extremo inferior debe ser menor que el superior';
    resultado_ <- 0;
}else{
    acum<-pbinom(extremo_sup,n,p)  - pbinom(extremo_inf,n,p)

    if (acum >p_ini && acum < p_fin)
    {
       resultado_ <- 1;
    } else {
       resultado_ <- 0;
    }
}
```

Peso: 1

Intentos:	2
Mostrar solución:	*(marcado)*
Mensaje correcto:	*(vacío)*
Mensaje regular:	*(vacío)*
Mensaje incorrecto:	*(vacío)*

Pregunta 4

La continuación de la pregunta anterior. El alumno debe introducir el extremo superior del intervalo de la pregunta anterior. Como la pregunta anterior ya tiene en cuenta este valor en la corrección, la respuesta a esta pregunta será correcta únicamente cuando la de la pregunta anterior lo sea. Para ello se asignará como resultado, el mismo que en la respuesta anterior (*resultados_[num_pregunta_ - 1]*).

La pregunta anterior tenía 2 intentos, así que para ser coherentes, esta pregunta también debe tener los mismos intentos. Así se puede hacer un segundo intervalo distinto en caso de ser incorrecto el primero introducido, de lo contrario en el segundo intento únicamente podríamos cambiar el extremo inferior y no el superior.

Aquí ya no es necesario mostrar la solución, porque en la pregunta anterior ya se mostrará un vector en caso de ser incorrecta.

Enunciado: Extremo superior
Solución respuesta: *(vacío)*
Parámetro sintaxis: *(vacío)*
Sintaxis respuesta: Natural
Plantilla exp. respuesta: *(vacío)*
Parámetro expresiones: *(vacío)*
Expresiones respuesta:

```
resultado_<- resultados_[ num_pregunta_ - 1]
```

Peso: 1
Ayuda contextual: *(vacío)*
Intentos: 2
Mostrar solución: *(desmarcado)*
Mensaje correcto: *(vacío)*
Mensaje regular: *(vacío)*
Mensaje incorrecto: *(vacío)*

Pregunta 5

Esta pregunta es similar a la 3, pero en este caso la respuesta es un vector con los 2 valores del intervalo, en lugar de 2 preguntas encadenadas. Para acceder a los dos valores del vector de respuesta, se usa *respuesta_[1]* y *respuesta_[2]*. No es necesario comprobar si el vector tiene estos 2 elementos, ya que de esto se encarga la sintaxis de respuesta Vector. Si el usuario no entra un vector con 2 elementos, devolverá el error al usuario, sin llegar a ejecutar el algoritmo de corrección de la pregunta.

El peso de esta pregunta es 2, para que tenga equivalencia con el peso de las preguntas 3-4 que forman una sola.

Enunciado:

```
Si X es binomial($n$,$p$), entra un intervalo de valores que acumule una
prob. mayor que $p_ini$ pero menor que $p_fin$.
Entre los dos números separados por un espacio.
```

Solución respuesta: *(vacío)*

Parámetro sintaxis: *2*

Sintaxis respuesta: Vector

Plantilla exp. respuesta: *(vacío)*

Parámetro expresiones: *5*

Expresiones respuesta:

```
extremo_inf <- respuesta_[1]
extremo_sup <- respuesta_[2]

if (extremo_inf > extremo_sup)
{
    mensaje_ <- 'El extremo inferior debe ser menor que el superior';
    resultado_ <- 0;
}else{
    acum<-pbinom(extremo_sup,n,p)  - pbinom(extremo_inf,n,p)
    if (acum >p_ini && acum < p_fin)
    {
        resultado_ <- 1
    } else {
        resultado_ <- 0
    }
}
```

Peso: 2

Ayuda contextual: *(vacío)*

Intentos: 1

Mostrar solución:	*(desmarcado)*
Mensaje correcto:	*(vacío)*
Mensaje regular:	*(vacío)*
Mensaje incorrecto:	*(vacío)*

Pregunta 6

El alumno debe introducir los 2 p-valores del vector indicado en la respuesta anterior. Este vector se accede mediante *respuestas_[[num_pregunta_ - 1]]*.

El algoritmo de corrección está basado en la plantilla de expresiones "Error relativo" con parámetro 5 (5% de error relativo).

Enunciado:

```
Entre los dos p-valores de la respuesta anterior separados por un espacio
```

Solución respuesta: *(vacío)*

Parámetro sintaxis: *2*

Sintaxis respuesta: Vector

Plantilla exp. respuesta: *(vacío)*

Parámetro expresiones: *5*

Expresiones respuesta:

```
extremo_inf <- respuesta_[1]
extremo_sup <- respuesta_[2]

if (extremo_inf > extremo_sup)
{
    mensaje_ <- 'El extremo inferior debe ser menor que el superior';
    resultado_ <- 0;
}else{
    acum<-pbinom(extremo_sup,n,p)  - pbinom(extremo_inf,n,p)
    if (acum >p_ini && acum < p_fin)
    {
        resultado_ <- 1
    } else {
        resultado_ <- 0
    }
}
```

Peso: 2

Ayuda contextual: *(vacío)*

Intentos: 1

Mostrar solución: *(desmarcado)*

Mensaje correcto: *(vacío)*

Mensaje regular: *(vacío)*

Mensaje incorrecto: *(vacío)*

Pregunta 7

Es la versión simplificada de la primera pregunta. Se utiliza una plantilla de expresiones de respuesta directamente, para comprobar que la respuesta está dentro de un intervalo de valores. Se ha de indicar como solución de respuesta un vector que contenga 2 elementos, el extremo inferior y el superior del intervalo de soluciones.

Enunciado:

```
Entra un número que esté entre el percentil $p1$ y el percentil $p2$
```

Solución respuesta:	*z*
Parámetro sintaxis:	*(vacío)*
Sintaxis respuesta:	Real
Plantilla exp. respuesta:	*Intervalo*
Parámetro expresiones:	*(vacío)*
Peso:	1
Ayuda contextual:	*(vacío)*
Intentos:	1
Mostrar solución:	*(desmarcado)*
Mensaje correcto:	*(vacío)*
Mensaje regular:	*(vacío)*
Mensaje incorrecto:	*(vacío)*

19. Apéndice 2: Convenios de CakePHP

El desarrollo con CakePHP proporciona muchas ventajas siempre y cuando se siga un convenio de desarrollo unificado entre todos los desarrolladores. Siguiendo estas normas se consigue que el código no vaya perdiendo calidad conforme nuevos desarrolladores se incorporan al proyecto.

19.1 Archivos

Los nombres de todos los archivos deben ser separados por carácter de subrayado y en minúsculas siempre. Ejemplo: *mi_nombre_de_fichero.php.* Además, según el tipo de clase (controlador, modelo, componente...) puede requerir un sufijo tipo "*_controller*". A continuación se detalla el nombre exacto de los ficheros según el tipo:

Tipo de clase	Nombre de clase	Nombre del Fichero
Controlador	SintaxisRespuestasController	app/controllers/ sintaxis_respuestas_controller.php
Modelo	CarpetasProblemas	app/models/ carpetas_problemas.php
Component	MyPaginationComponent	app/components/ my_pagination.php
Helper	DebugWindowHelper	app/views/helpers/ debug_window.php
Form	EncuestasAsignatura*Index*Form	app/forms/ encuestas_asignatura.php
Element	-	app/views/elements/ sidebar.thtml
View	ProblemasController::index()	app/views/problemas/index.thtml

19.2 Controladores

- Los nombres de los controladores siempre están en plural como *SintaxisRespuestasController*.
- Las acciones (funciones) de los controladores siempre en singular y subrayado como *index(), edit(), delete(), ver_respuestas()*.
- Las funciones que tienen el simbolo '_' como prefijo en su nombre, son funciones privadas del controlador. Es decir, son funciones auxiliares que no pueden ser ejecutadas desde una petición por URL.
- Si varias acciones del mismo controlador repiten código:
  - Utilizar una función auxiliar si intervienen varios modelos.
  - Si interviene un único modelo, estudiar su incorporación como función en el modelo.
- Si varios controladores repiten código:
  - Crear un componente si el código es suficientemente genérico y no interviene ningún modelo.
  - Si interviene un único modelo, estudiar su incorporación como método del modelo.
  - Si interviene más de un modelo, se puede incorporar a uno de los modelos, si hay una relación entre los modelos, o bien utilizar *$this->requestAction* para llamar la función de un controlador desde otra acción.

19.3 Modelos

- Las tablas de la base de datos creadas a partir de un modelo, siempre están en minúsculas, separando las palabras por el carácter subrayado. Por ejemplo, para el modelo EncuestasAsignatura, la tabla es encuestas_asignaturas.
- Si CakePHP no es capaz de traducir el nombre de la tabla de la base de datos pasándolo a mayúsculas, se ha de indicar explícitamente en el modelo en la variable *useTable*:

```
<?php
class EncuestasAsignatura extends AppModel
{
      var $name = 'EncuestasAsignatura';
      var $useTable = 'encuestas_asignaturas';
}
```

- Los modelos son el mejor sitio donde añadir el código necesario para manipular los datos del modelo y sus modelos relacionados.
- En lugar de crear una consulta SQL compleja, crear una MySQL view y a partir de ella, un modelo. Preferiblemente, los nombres de las views deben tener el prefijo *view_*.

19.4 Componentes

Son el mejor lugar donde incorporar código que no depende de ningún modelo en concreto, como el componente que gestiona la paginación de los datos.

- Se ha de evitar establecer nombres fijos de controladores o modelos. Deben ser totalmente genéricos.
- Si necesitan utilizar una variable en un controlador, crear la variable en el controlador con un nombre preestablecido, y usarla desde el componente: por ejemplo, la variable *$access* en los controladores es usada por el componente *rdAuth*.
- Habilitar el auto-arranque de los componentes. La función *startup* siempre es llamada justo antes de la acción del controlador, y la instancia del objeto del controlador actual es pasada por parámetro:

```
class TestComponent extends Object
{
	var $controller;
	function startup(&$controller)
	{
		$this->controller = & $controller;
	}
}
```

En esta función se pueden inicializar estructuras de datos, y se ha de intentar incorporar en ella la máxima funcionalidad posible del componente para que su uso sea automático.

19.5 Helpers

Su creación se recomienda cuando lógica de presentación compleja se repita entre diferentes vistas.

- Un helper puede usar a otro helper. Siempre que se pueda, se han de crear los helpers

encima de los ya existentes para no repetir código. Para incluir los helpers, se utiliza la variable *$helpers:*

```
var $helpers=array('Head','Javascript','Html');
```

Para usar uno de estos helpers, en una de las funciones del helper en desarrollo, la sintaxis es:

```
echo $this->Html->input()
```

- Si un helper está relacionado con ficheros CSS, o Javascript, el helper debe incluirlos automáticamente. En la función *afterRender()*, llamada automáticamente, es el mejor lugar para referenciar los ficheros:

```
class TreeHelper extends Helper
{
	var $helpers=array('Head');
	function afterRender()
	{
		$this->Head->js('dtree/dtree');
		$this->Head->css('dtree/dtree');

	}
}
```

19.6 Elements

Si código de presentación en html se repite entre varias vistas, se han de crear elements.

- Si está relacionado directamente con un controlador, la carpeta de creación del element será app/views/elements/*nombre_controlador/nombre.thtml.*
- Si se usa en más de un controlador, ha de ser totalmente genérico (mediante parámetros) y que pueda ser usado por todos los controladores. En ese caso, se creará en la carpeta app/views/elements/*generic/nombre.thtml.*

19.7 Views

Las vistas deben tener la mínima lógica de presentación posible. Para la lógica, ya existen los helpers, y para bloques de html repetidos entre vistas, los elements.

20. Apéndice 3: Conversión de los datos

Para convertir los datos de la versión anterior de e-status a la nueva, se han creado dos scripts semi automáticos. A continuación se detallan los pasos para pasar los datos de *ka.upc.es* (en MS SQL Server) a *estatus.upc.es* (MySQL).

1. Realizar una copia de seguridad local de los problemas nuevos en *estatus.upc.es* a formato XML desde el Editor.

2. Como usuario root en *estatus.upc.es*, ejecutar lo siguiente:

```
cd /srv/www/htdocs/estatus_cake/app/webroot

cp
/srv/www/htdocs/estatus_cake/vendors/estatus/otros/conversion/convert
ir2.php .
```

3. Copiar a ka.upc.es el script de conversión. Desde estatus.upc.es:

```
cd /srv/www/htdocs/estatus_cake/vendors/estatus/otros/conversion

scp convertir.php root@ka.upc.es:/wwwroot/htdocs/convertir.php
```

4. Acceder mediante el navegador a la dirección *http://ka.upc.es/convertir.php*. Se ha de esperar hasta que el proceso termine, mostrará "*FIN*". Este script copia la base de datos MS SQL Server en producción a una base de datos MySQL temporal en *ka.upc.es*.

5. Eliminar el script de conversión de *ka.upc.es*. Como usuario root en ka, ejecutar:

```
rm /wwwroot/htdocs/convertir.php
```

6. Generar una copia de seguridad de la base de datos MySQL de ka a un fichero .sql.

```
cd /root
mysqldump -u root estatus_import >estatus_import.sql
```

7. Importar el fichero .sql anterior. Desde *estatus.upc.es*, como usuario root:

```
cd /root
scp root@ka.upc.es:/root/estatus_import.sql ./copia.sql
```

```
mysql -u root --password=password_root estatus_import < copia.sql
```

8. Acceder mediante el navegador a la dirección *http://estatus.upc.es/convertir2.php*. Se ha de esperar hasta que el proceso termine. Este script convierte la estructura de la base de datos anterior a la nueva estructura. Es un proceso muy lento, puede tardar varios minutos.

9. Eliminar el script de conversión de *estatus.upc.es*. Como usuario root, ejecutar:

```
rm /srv/www/htdocs/estatus_cake/app/webroot/convertir2.php
```

10. Finalmente, acceder a *http://estatus.upc.es* y restaurar los problemas en formato XML desde el Editor.