3. LAS HERRAMIENTAS

3.1. ECLISPE

Eclipse es un entorno integrado de desarrollo multiplataforma libre, distribuido gratuitamente por Internet, para crear aplicaciones clientes de cualquier tipo.

Eclipse fue creado originariamente por IBM, empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática. Ahora lo desarrolla la Fundación Eclipse, organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse utiliza módulos (en inglés plugin) para proporcionar sus diversas funcionalidades, a diferencia de otros entornos monolíticos dónde las funcionalidades están todas incluidas las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, existe un módulo por apoyar a C/C++. Existen módulos de todo tipo, desde una aplicación Telnet hasta soporte a diferentes sistemas de bases de datos.

Para nuestro proyecto hemos aprovechado las funcionalidades de TruStudio, en torno integrado de desarrollo Open Source con las herramientas necesarias por utilizar PHP.

Otra de las ventajas de haber utilizado Eclipse es que lleva incorporado un cliente de CVS, que nos ha permitido trabajar a todos los proyectistas conjuntamente, compartiendo de forma inmediata y segura el código desarrollado por cada uno.

3.2. CVS

Las siglas provienen del inglés y son un acrónimo de Concurrente Versions System (Sistema de Control de Versiones) o Concurrent Versioning System.

Se trata de un sistema que mantiene un registro de todo el trabajo realizado y los cambios en la implementación de un proyecto y que permite que diferentes desarrolladores, potencialmente separados por grandes distancias, colaboren juntos.

Los sistemas de control son utilizados principalmente en la industria del software para controlar las diferentes versiones del código fuente. Aún así, los mismos conceptos son aplicables a otros ámbitos (y no sólo para código fuente) como la generación de documentos, imágenes, etc. En el mundo del software libre se ha hecho muy popular entre los desarrolladores de código abierto, que trabajan con el CVS bajo la licencia GPL.

Haber tenido la oportunidad de trabajar con CVS nos ha permitido tener siempre la última versión del código y facilitar la colaboración y ayuda entre todos los proyectistas como por ejemplo en la solución de errores.

Los miembros del DFWikiteam hemos trabajado en dos CVS distintos. Inicialmente se trabajó en el CVS localizado en LaFarga (http://lafarga.cat/) pero debido a unos problemas nos pasamos a trabajar con el CVS de SourceForge (http://sourceforge.net/).

3.3. NAVEGADORES WEB

Obviamente, un navegador Web es una herramienta vital por trabajar en un proyecto Web. Entre todas las opciones disponibles los desarrolladores del DFWikiteam han utilizado mayoritariamente Mozilla Firefox. Este navegador ha sido de gran utilidad para hacer este proyecto sobretodo debido a sus extensiones.

- **Firebug**

Es una extensión para Mozilla Firefox de gran ayuda sobre todo para los que se dediquen al desarrollo de sitios Web. Con ella podremos editar y depurar código, además de monitorizar el sitio sobre el que trabajemos. Con esta herramienta se pueden utilizar características tan interesantes como:

- Inspeccionar y editar el HTML en tiempo real.
- Mejorar los estilos del sitio sin tener que acudir directamente a los ficheros CSS.
- Monitorizar los tiempos de carga de un sitio.
- Depurador de Javascript
- Gestor de errores en Javascript, CSS y XML.
- Regla para especificar las dimensiones en los estilos.
- Explorador del DOM

- **Web Developer**

Web Developer es una extensión diseñada por Chris Pederick para Firefox, Flock, Mozilla y Seamonkey. Agrega un menú y una barra de herramientas al navegador con funcionalidades aplicables al diseño y desarrollo de sitios Web.

Entre sus diversas funcionalidades, la extensión de Firefox Web Developer dedica un apartado especial a las herramientas de validación y adecuación respecto a los estándares Web, principalmente desde el punto de vista del desarrollo Web con CSS y HTML, y a las normas de accesibilidad de la "Web Accessibility Initiative", WAI, y el "World Wide Web Consortium", W3C.

Incluye la posibilidad de validar HTML, con sólo "clic" a través del "Markup Validation Service" del W3C. Además de la validación del sitio Web, facilita la validación de archivos concretos respecto a las CSS y el código HTML.

3.4. CROM

http://morfeo.upc.edu/crom Es un servidor con un Moodle instalado, y ha sido una de las herramientas básicas del proyecto.

En este servidor de Moodle existe un curso exclusivo para los desarrolladores del DFWikiteam dónde todos los miembros tienen permisos de profesor para tener libertad total.

Entre las actividades de más uso dentro de este curso destacamos el foro, que servía como canal de comunicación directa entre los miembros del DFWikiteam. Dentro del foro existen threads con dudas de programación, información sobre nuevos bugs descubiertos, estado del desarrollo y mensajes de coordinación del DFWikiteam.

También hace falta destacar varias Wikis que han servido, entre otras cosas, para compartir la documentación técnica de todos los módulos implementados o para escribir la memoria del proyecto que es común para todos los integrantes del DFWikiteam.

3.5. BLOG

Un Blog es un sitio Web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente.

Todo proyectista del DFWikiteam ha tenido un Blog donde ha realizado un seguimiento del proyecto para mantener informados al resto de compañeros.

4. LA TECNOLOGÍA

4.1. PHP

PHP es un acrónimo recursivo de PHP: Hipertext Preprocessor, aún cuando originalmente eran las siglas de Personal Home Page Tools. Se trata de un lenguaje de programación interpretado, libre y muy popular, utilizado por generar contenido dinámico en la Web.

Surgió hacia 1994 como un proyecto de un conjunto de scripts escritos en Perl por Rasmus Lerdof, al que se le añadieron Zeev Suraski y Andi Gutmans, dos programadores de Israel del Technion. El año 1997 salió PHP 3, la primera versión estable en la cual el lenguaje era parecido al actual, en mayo del 2000 salió la versión 4 y en julio de 2004, la 5.

Se trata de un lenguaje extremadamente modular, que lo hace útil para la instalación y uso en servidores Web. Es muy parecido, en tipo de datos, sintaxis y funciones a los lenguajes de programación C y C++. En relación con esto, hace falta tener presente que desde la versión 5 PHP incluye un mayor soporte a la programación orientada a objetos.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX (y de ese tipo, como Linux), Windows y Mac OS X, y puede interactuar con los servidores de Web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

PHP puede ser incluido dentro el código HTML indicando mediante unas etiquetas al intérprete, cuando debe empezar a interpretar el código como PHP y cuando puede dejar de hacerlo.

Además, PHP permite acceder a bases de datos Oracle, Sybase, PostgreSQL, Interbase, MySQL, SQLite, MSSQL y otras, y soporta bastantes protocolos entre los que se encuentran MAP, SNMP, NNTP, POP3, HTTP, LDAP, XMLRPC y SOAP.

Como principales ventajas de desarrollar en PHP, podemos comentar:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los sistemas de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial (http://www.php.net/manual/es/), entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es Software Libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de programación orientada a objetos.
- Permite crear los formularios para la Web.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

4.2. HTML

HTML (Acrónimo de Hyper Texto Markup Language, en castellano, lenguaje de marcas de hipertexto), es un lenguaje de marcas que deriva de SGML, diseñado para estructurar textos y relacionarlos en forma de hipertexto.

Gracias a Internet y a los navegadores Web, se ha convertido en uno de los formatos más populares que existen en la construcción de documentos en formato electrónico.

En su origen, el HTML era un lenguaje diseñado por compartir información científica entre científicos de todo el mundo. Era puramente un lenguaje estructural, en qué no había forma de describir la apariencia de las páginas (ni siquiera la posibilidad de poner un texto en negrita o cursiva). Más adelante se añadieron numerosas opciones para formatear y presentar texto y gráficos. Se desarrollaron las ampliaciones de HTML para conseguir la presentación deseada, pero siempre desde diferentes perspectivas de diferentes desarrolladores, que acabaron con

varias soluciones no estándares para diferentes navegadores. Esto provocó la aparición del consorcio que controla la evolución de HTML: el W3C (World Wide Web Consortium).

4.3. XML

XML (eXtensible Markup Language), es un lenguaje de etiquetas desarrollado por el World Wide Web Consortium.

Al igual que el HTML es una simplificación y adaptación del lenguaje SGML, y permite definir una gramática de lenguajes específicos (al igual que HTML). Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi en cualquier cosa imaginable. XML es por lo tanto, una tecnología sencilla que se complementa otras tecnologías ampliando enormemente sus posibilidades de tratar la información.

Tiene un papel muy importante en la actualidad ya que facilita la compatibilidad entre sistemas para compartir información de manera segura, fiable y fácil.

4.4. XHTML

XHTML (Lenguaje de Marcado de Hipertexto Extensible) es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML . XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos.

Ante la llegada al mercado de un gran número de dispositivos, XHTML surge como el lenguaje cuyo etiquetado, más estricto que HTML, va a permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella. XHTML puede incluir otros lenguajes como MathML , SMIL o SVG , al contrario que HTML.

XHTML, al estar orientado al uso de un etiquetado correcto, exige una serie de requisitos básicos a cumplir en lo que a código se refiere. Entre estos requisitos básicos se puede mencionar una estructuración coherente dentro del documento donde se incluirían elementos correctamente anidados, etiquetas en minúsculas, elementos cerrados correctamente, atributos de valores entrecomillados, etc.

A continuación se pueden ver algunos ejemplos de los aspectos más importantes a tener en cuenta a la hora de utilizar XHTML.

- Los documentos deben estar *bien formados*:

Un formato correcto en un documento XHTML es muy importante. Esto quiere decir que todos los elementos deben tener etiquetas de cierre, deben estar escritos de una forma determinada y además todos los elementos deben estar anidados correctamente.

Código de elementos anidados:

```
<p>Ejemplo de elementos bien <em>anidados</em>.</p>
<p>Ejemplo de elementos mal <em>anidados</p>.</em>
```

- Los nombres de atributos y elementos deben ir en minúsculas:

Tanto los elementos como los atributos deben ir en minúsculas para todos los elementos HTML y los nombres de atributos. Esto es importante ya que XML interpreta las mayúsculas y las minúsculas de forma diferente.

```
<body>Ejemplo correcto</body>
<BODY>Ejemplo incorrecto</BODY>
```

- Los elementos que no estén vacíos necesitan etiquetas de cierre:

Ejemplo correcto.
Ejemplo correcto.
Ejemplo incorrecto. Ejemplo incorrecto.

- Los valores de las etiquetas deben ir siempre entre comillas:

Todos los valores de los atributos deben ir entre comillas, incluso aquellos que sean numéricos.

<table rows="3">
<table rows=3> ejemplo incorrecto

4.5. CSS

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los *Estilos* definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

h1 {color: red;}

h1 es el selector

{color: red;} es la declaración

El selector funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto. En el ejemplo anterior, el selector h1 indica que todos los elementos h1 se verán afectados por la declaración donde se establece que la propiedad color va a tener el valor red (rojo) para todos los elementos h1 del documento o documentos que estén vinculados a esa hoja de estilos.

Las tres formas más conocidas de dar estilo a un documento son las siguientes:

1. Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento <link>, el cual debe ir situado en la sección <head>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN">
<html>
 <head>
  <title>Título</title>
  <link rel="stylesheet" type="text/css"
   href="http://www.w3.org/css/officeFloats.css" />
 </head>
 <body>
 .
 .
 .
 .
 </body>
</html>
```

2. Utilizando el elemento <style>, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección <head>. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN">
<html>
 <head>
  <title>hoja de estilo interna</title>
  <style type="text/css">

   body {
    padding-left: 11em;
    font-family: Georgia, "Times New Roman", serif;
    color: red;
    background-color: #d8da3d;
   }

   h1 {
    font-family: Helvetica, Geneva, Arial, sans-serif;
   }

  </style>
 </head>
 <body>
  <h1>Aquí se aplicará el estilo de letra para el Título</h1>
 </body>
</html>
```

3. Utilizando estilos directamente sobre aquellos elementos que lo permiten a través del atributo <style> dentro de <body>. Pero este tipo de estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

Algunas normas básicas a la hora de crear una CSS son las siguientes:

- En el siguiente ejemplo, h1{color: red;}, el *selector*, <h1>, le dice al navegador la parte del documento que se verá afectada por esa regla. Los selectores pueden aparecer individualmente o agrupados, separándolos con comas:

```
h1, h2, h3 {
  color: red;
}
```

o lo que es lo mismo

```
h1 {color: red;}
h2 {color: red;}
h3 {color: red;}
```

- La *propiedad*, que en este caso sería color, especifica qué aspecto se va a cambiar. En este ejemplo la propiedad cambiada será el color. Las propiedades que se desean modificar en una CSS para un mismo selector pueden agruparse, pero será necesario separar cada una de ellas con un punto y coma.

```
p {text-align:center;color:red}
```

Normalmente se describe una propiedad por línea, de la siguiente manera:

```
h1 {
  padding-left: 11em;
  font-family: Georgia, "Times New Roman",Times, serif;
  color: red;
  background-color: #d8da3d;
}
```

- El *valor*, en este caso red, establece el valor de la propiedad. Es importante recordar que si el valor está formado por más de una palabra, hay que ponerlo entre comillas.

```
p {font-family: "sans serif";}
```

4.6. JAVASCRIPT

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM. El Document Object Model (en español 'Modelo de Objetos de Documento'), frecuentemente abreviado DOM, es una forma de representar los elementos de un documento estructurado (tal como una página web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. El responsable del DOM es el World Wide Web Consortium (W3C). En resumen, el DOM es una API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como Javascript.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que fabricó los primeros navegadores web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se utilizaba en páginas Web HTML, para realizar tareas y operaciones en el marco de la aplicación cliente servidor.

Con la irrupción de Web 2.0, Javascript se ha convertido en un auténtico lenguaje de programación que aporta la potencia de cálculo del navegador para aumentar la usabilidad de aplicaciones Web.

Javascript se puede incluir en cualquier documento HTML, o todo aquel que termine traduciéndose en HTML en el navegador del cliente; ya sea PHP, ASP, SVG...

El código va inscrito dentro de los elementos HTML <script> y </script>:

```
<script type="text/javascript">
    // código JavaScript
</script>
```

Muchos incluyen comentarios HTML para que navegadores antiguos no interpreten el código JavaScript, sin embargo ningún navegador existente hoy en día necesita esta práctica.

Para incluir un archivo externo, bastará con escribir:

```
<script type="text/javascript" src="[URL]"></script>
```

Siendo [URL] el url relativo o absoluto apuntando a un archivo con código JavaScript.

5. LA METODOLOGÍA DE TRABAJO

Para llevar a cabo el proyecto se hacen reuniones semanales donde cada miembro del equipo DFWikiteam expone sus avances y sus dudas. Estas reuniones facilitan que todos estemos al corriente de la evolución de la NWiki y la coordinación de los miembros del grupo. Por otro lado los desarrolladores continúan en comunicación a través de foros, e-mail o Wikis.

Este modo de trabajo se basa en una metodología ágil. El desarrollo ágil es un marco de trabajo conceptual para llevar a cabo proyectos de ingeniería de software.

Los métodos ágiles enfatizan la comunicación en tiempo real, preferiblemente en persona en lugar de por escrito. Muchos de los equipos de trabajo ágiles están localizados en un solo espacio de trabajo e incluyen en toda la gente necesaria para terminar el software. Esto incluye, como mínimo, programadores y sus customers (que pueden ser tanto los clientes como los gestores del producto o analistas del negocio). Este espacio de trabajo podría incluir testers, diseñadores de la iteración, escritores técnicos y gestores.

Los métodos ágiles también enfatizan el software funcional como la medida primaria de progreso. Combinado con la preferencia de la comunicación en persona, los métodos ágiles producen poca documentación respeto a otros métodos. Esto ha sido criticado por los detractores de los métodos ágiles porque dicen que esto hace que el método no sea lo suficiente disciplinado.

Algunos métodos ágiles, como el que utilizamos en nuestro caso, el "extrem Programming", tratan de minimizar riesgos desarrollando software en pequeños espacios de tiempos, denominados iteraciones, que típicamente duran entre una y cuatro semanas. Al final de cada iteración, el equipo re-evalúa las prioridades del proyecto. Es un sistema de trabajo muy enriquecedor ya que estás en constante contacto con tus compañeros y minimiza los errores de coordinación.

5.1. DINÁMICA EN GRUPO

Como la metodología de trabajo del DFWikiteam se basa en el trabajo coordinado de dos de sus miembros en el mismo proyecto, son constantes las interacciones e influencias entre ellos. Fruto de estas influencias y comunicaciones, el grupo desarrolla un número de dinámicas que los distingue de un colectivo de personas escogidas al azar.

La primera que podemos destacar, y que englobaría a todo el equipo de desarrolladores, es la mesa redonda que se crea en cada reunión semanal del DFWikiteam. En esta reunión Marc Alier, con el papel de coordinador, empieza haciendo una pequeña introducción del tema que se tratará (novedades en Moodle de aquella semana, peticiones de usuarios, reporte de nuevos bugs, etc.). Tras este "briefing" introductorio se pasa el turno a los proyectistas, que exponen el trabajo realizado. El orden de intervención de los expositores surge de manera espontánea según como se van relacionando los temas que exponen ellos mismos. Tras la intervención de cada expositor se decide si el trabajo está finalizado, en cuyo caso se le asigna una nueva tarea o por el contrario todavía se debe seguir desarrollando. Una vez todos los miembros han expuesto su trabajo semanal se hace un pequeño resumen de las conclusiones de la reunión de manera muy informal por asegurar que todo el mundo ha entendido el nuevo trabajo que se le ha encargado.

Un tipo de dinámica diferente se establece entre la pareja de proyectistas. Tal y como hemos dicho anteriormente, el modelo de desarrollo del DFWikiteam se basa en una metodología ágil. Dentro de las metodologías ágiles que existen, algunos de los miembros del DFWikiteam han estando utilizando la denominada "extrem Programming", y otras parejas han seguido algunas de las características de la misma. Una de las características fundamentales de este método, que han aplicado todos los miembros del DFWikiteam, es la programación por parejas puesto que se supone que el código escrito de este modo, es de mayor calidad aún cuando se escriba menos código, ya que el código se revisado y discutido mientras se escribe. En consecuencia, la dinámica que existe dentro de este grupo de personas es de total comunicación y conocimiento de todo lo que se hace. Aunque hayan momentos en que el trabajo se reparta y cada cual haga lo que buenamente pueda con la parte que le toca, la comunicación "intragrup" continúa haciendo informes de todo lo hecho vía e-mail o haciendo

una pequeña reunión impersonal, así todo el mundo siempre se consciente de cómo ha sido realizado el trabajo.

5.2. ASPECTOS POSITIVOS Y NEGATIVOS

La metodología seguida por el DFWikiteam es muy parecida al "extrem Programming". Podríamos decir que la única diferencia relevante es que muchas de las veces el trabajo no se hacía en presencia de los dos miembros del grupo, pero esto quedaba solucionado con la información que se transmitían los miembros una vez acabada la parte hecha individualmente. Por lo tanto, la mayoría de ventajas e inconvenientes son similares a los propios del "extrem Programming".

Ventajas:

- La frecuente interacción con el cliente garantiza que se está haciendo aquello que realmente se quiere: Las reuniones semanales del DFWikiteam garantizan este punto.
- Las pequeñas releases y el continuo proceso de diseño permito la frecuente revisión del software por parte de los usuarios y de los desarrolladores.
- La simplicidad del código facilita su posterior modificación.
- La gran complementación entre simplicidad y comunicación: Con más comunicación resulta más fácil identificar lo que se debe hacer y lo que no. Mientras más simple sea el sistema, menos se tendrá que informar, lo que conlleva a una comunicación más completa.
- La refactorització de código aporta facilidad de ampliación: Rescribir partes del código para aumentar su legibilidad y mantenibilidad facilita el trabajo de futuros desarrolladores.
- Mejor calidad en el código compartido: Las pruebas por parte del equipo de desarrolladores sobre un mismo código garantizan que los posibles errores serán detectados.

Inconvenientes:

- Esta metodología resulta difícil de adaptar a proyectos grandes: La falta de una mayor planificación y el centrarse rápidamente en los primeros resultados hacen difícil su uso en proyectos a gran escala.
- A veces el coste de tener una pareja de programadores en lugar de uno solo es demasiado grande.
- La falta de planificación inicial a gran escala puede provocar desconfianza.
- Es común la falta de documentación.

6. EL DESARROLLO DEL PROYECTO

En este apartado se va a explicar el desarrollo que ha tenido el proyecto y las diferentes etapas por las que ha pasado.

6.1. ETAPA 1: Formación sobre Moodle

Como primer paso previo para poder iniciar nuestro proyecto, tuvimos que familiarizarnos con la plataforma Moodle. Inicialmente nos instalamos la versión 1.7.

La forma más sencilla es utilizar uno de los paquetes completos de instalación disponibles en la página de descargas de download.moodle.org que incluyen todo lo necesario para instalar Moodle correctamente cumpliendo los requerimientos.

Contenido del paquete:

- Moodle - programa para crear y gestionar cursos.
- Apache - servidor Web.
- PHP – lenguaje de programación
- MySQL - base de datos.

Una vez instalado procedí a utilizar la plataforma a nivel de usuario viendo todas sus funcionalidades y familiarizándome con ellas. Después analicé y busqué información para saber como estaba estructurado internamente Moodle a nivel de carpetas y archivos.

Breve resumen de los contenidos del directorio Moodle:

config.php - contiene la configuración fundamental.
install.php - el script que ejecutará para crear el archivo config.php.
version.php - define la versión actual del código de Moodle.
index.php - la página principal del sitio.
admin/ - Código para administrar todo el servidor.
auth/ - Módulos para la autenticación de usuarios.

blocks/ - Módulos para los pequeños bloques laterales contenidos en muchas páginas.
calendar/ - Código para manejar y mostrar eventos de calendario.
course/ - Código para presentar y gestionar los cursos.
doc/ - Documentación de ayuda de Moodle. (Por ejemplo esta página).
files/ - Código para presentar y gestionar los archivos cargados.
lang/ - Textos en diferentes idiomas, un directorio por idioma.
lib/ - Librerías del código fundamental de Moodle.
login/ - Código para manejar las entradas y creación de cuentas.
mod/ - Todos los módulos de los cursos de Moodle.
pix/ - Gráficos genéricos del sitio.
theme/ - Paquetes de temas/pieles para cambiar la apariencia del sitio.
user/ - Código para mostrar y gestionar los usuarios.

Después de analizar el funcionamiento de Moodle procedí a instalar la Wiki sustituyéndola por el módulo oficial. Luego, seguí el mismo procedimiento que había seguido con Moodle, primero me familiaricé a nivel usuario y luego procedí a analizar la estructura interna.
La Wiki sigue la misma estructura que Moodle, por lo tanto una vez conoces Moodle internamente es fácil conocer la Wiki.

Breve resumen de la situación de la Nwiki dentro del directorio Moodle:

- La Wiki es un módulo y todos sus ficheros están en moodle/mod/wiki.
- Los bloques de nuestra Wiki están definidos en los ficheros moodle/blocks/wiki_XXX, que son las cajas que puedes añadir y quitar de los laterales de las páginas, son accesos rápidos o información.
- Además la Wiki también es un formato de curso, es decir que en vez de tener semanas, tiene una Wiki. El código que define el curso en formato Wiki está en moodle/course/format/wiki
- Las funcionalidades están definidas en moodle/mod/wiki, concretamente en los ficheros locallib.php y lib.php, éstas son comunes al módulo Wiki y al curso Wiki.
- También tenemos datos en la carpeta moodle/lang, porque en los ficheros de los módulos no se escribe directamente lo que quieres que aparezca en html, sino que lo haces mediante la función: print_string("constant","nomModul"). Para cada lenguaje

creas un fichero .php con el nombre del módulo, y al mostrarse la página se cambian las constantes por sus respectivos strings, en el lenguaje en el que se está ejecutando Moodle.

El siguiente paso tras aprender el funcionamiento tanto de Moodle como de la NWiki a nivel usuario y conocer su estructura interna, era familiarizarnos con la tecnología utilizada por esta plataforma.

Como la mayor parte de Moodle está escrito en PHP, me leí un manual para familiarizarme con el lenguaje y poder interpretar mejor las funciones implementadas de Moodle.

Adicionalmente al manual de PHP que se encuentra en su sitio oficial, Moodle pone a disposición de los desarrolladores una herramienta de documentación del código ubicada en (http://xref.moodle.org/nav.html?index.html), donde disponemos de la documentación de las funciones propias de Moodle, y donde podemos colaborar añadiendo la documentación de aquellas funciones que generemos utilizando la sintaxis phpdoc muy similar a javadoc para los comentarios en el código fuente.

Finalmente, para escribir el código había que instalar un editor. Después de probar algún que otro programa escogí, aconsejada por miembros del DFWikiteam el Eclipse, ya que integra un cliente de CVS (control de versiones) y dispone de un conjunto de plugins para trabajar cómodamente con PHP.

Una vez leído un manual sobre PHP y habiéndome bajado el Eclipse, ya estaba preparada para participar en el desarrollo de la NWiki.

6.2. ETAPA 2: Accesibilidad Web

En la primera reunión del DFWikiteam se nos encomendó la tarea de hacer accesible la NWiki. Esta labor era muy necesaria ya que es uno de los requisitos indispensables para que la NWiki sea el módulo oficial de Moodle.

La accesibilidad Web era un tema desconocido para nosotros así que nos pusimos a buscar información en seguida. Tras la búsqueda nos dimos cuenta que para que una página Web fuera accesible tenía que cumplir con los estándares de la W3C.

Estuvimos mirando en los foros de Moodle. Tenían muchos debates que hablaban del tema. Descubrimos que el principal objetivo de Moodle para que la aplicación Web fuese más accesible era que cumpliese con el estándar XHTML.

6.3. ETAPA 3: XHTML Transitional

Tras informarnos de lo que era el estándar XHTML supimos que como primera regla todo documento XHTML tiene que tener definido su tipo de documento.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

En la definición del Doctype de la versión de Moodle 1.7 vimos que se tenía que cumplir con el estándar XHTML Transitional. Ese fue, por lo tanto, nuestro primer objetivo.

Como la NWiki tiene cientos de archivos, nos dividimos el trabajo en dos. En un inicio nos pusimos a tocar el código directamente sin saber como visualizarlo. En seguida comprendimos que era un error actuar de esa manera porque no sabíamos si estábamos cambiando la forma de visualización. En ese momento empezamos a trabajar desde la Web de la Wiki y de cada página, buscábamos donde se hallaba el código correspondiente y corregíamos no sin antes comprobar que seguía teniendo la misma presentación.

Este hecho ha sido uno de los problemas más costos ya que al no haber sido nosotros los que habíamos creado la NWiki en ocasiones se hacia realmente difícil saber donde se hallaban según que trozos de código. Por ese motivo, hemos estado en pleno contacto con el becario, Jordi Piguillem para que nos ayudase en esa tarea. Él nos ha guiado, a lo largo del proyecto,

cuando no encontrábamos el código correspondiente de tablas, imágenes, etc. que se visualizaban en la página de la Wiki y no cumplían el estándar.

La herramienta que más hemos utilizado para que la NWiki cumpla el estándar XHTML ha sido los validador W3C. Esta herramienta se encarga de ver que la programación de la página en cuestión sea válida, es decir, que no existan errores de sintaxis como tags abiertos, etc. De forma automática nos indicaba el resultado de su análisis de la codificación y proponía las soluciones oportunas a aquellos errores que haya detectado. No siempre las soluciones propuestas eran correctas pero, en ese caso te aportaba ideas sobre que podía ser.

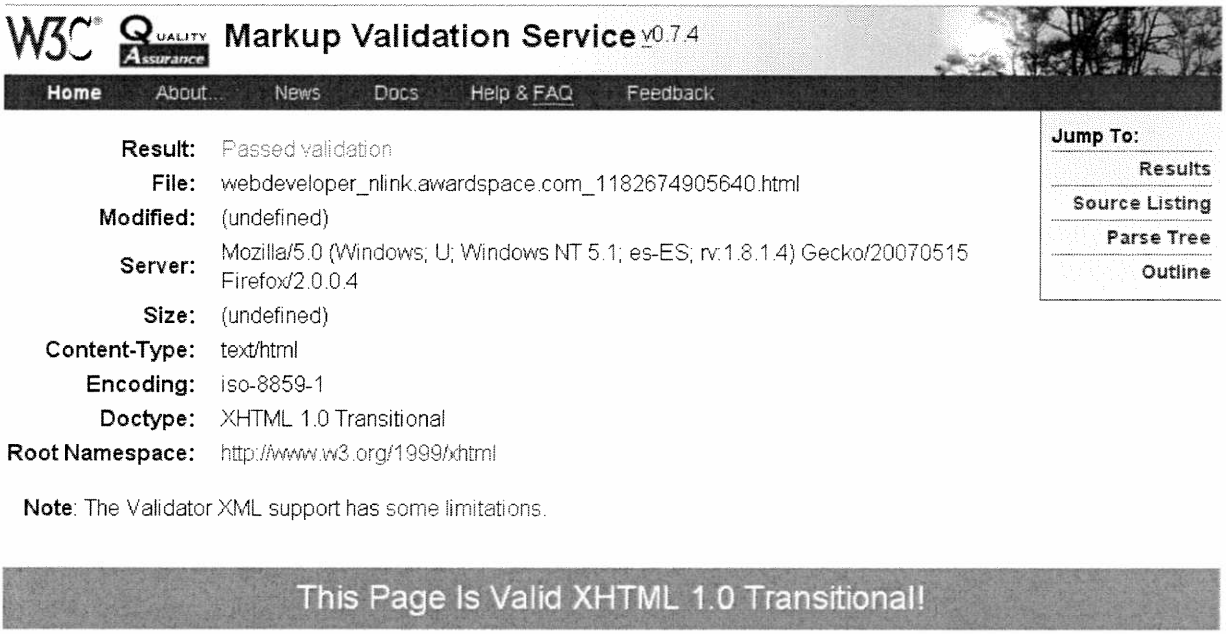


Ilustración 8: validación

Este validador nos ha sido de gran utilidad ya que teníamos la certeza de haber hecho el trabajo bien. También nos ha sido de gran ayuda la extensión Web Developer de Firefox, donde podíamos validar las páginas de la NWiki localmente.

Cuando la NWiki ya cumplía el estándar XHTML Transitional, es decir que todas las páginas de la Wiki validaban, hicimos un manual de XHTML Transitional donde explicábamos las principales reglas que debe cumplir una página Web para que su código esté bien formado. Esté manual estaba dirigido sobretodo a los miembros del DFWikiteam para que el nuevo código que se escribiese, cumpliese el estándar.

6.4. ETAPA 4: Creación de la capa de presentación

Tras conseguir que la NWiki para Moodle 1.7 cumpliese el XHTML Transitional, el grupo de desarrolladores DFWikiteam dio por cerrada la versión para 1.7, y empezamos con el desarrollo de la NWiki para Moodle 1.8. Esta versión utiliza el XHTML Strict y por lo tanto nuestro objetivo era conseguir que la NWiki también lo cumpliese.

Durante la fase de adaptación de la NWiki al estándar XHTML Transitional nos dimos cuenta que al haber en casi todos los ficheros código HTML, era muy costoso modificarlo y daba lugar a muchas ambigüedades. Por ese motivo se decidió que antes de conseguir que cumpliese con el estándar más estricto, se separase toda la presentación del contenido, para así solo modificar un único fichero.

Para ello se tenía que crear un fichero nuevo donde se situaría todo lo relacionado con la presentación (todos los tags de HTML), creando nuevas funciones que generaran todo el código HTML. De este modo se conseguiría separar por completo la presentación del dominio, creando una nueva capa.

Este fichero se tenía que llamar weblib.php dado que la NWiki tiene que seguir la misma estructura de Moodle y el fichero de Moodle que tiene la función de capa de presentación se llama así. Estudiamos ese fichero a fondo ya que teníamos que intentar utilizar sus funciones en todas las partes de código que se pudiese reemplazar y a partir de allí crear un fichero que complementase al weblib de Moodle y se adaptase a nuestras necesidades, sin que se replicase ninguna función.

Inicialmente se pensó en la idea de trasladar todo el código HTML a nuestro fichero sin tener en cuenta si hay información de dominio mezclada y después de haber aislado el código HTML en un único fichero, crear nuevas funciones para conseguir los datos del dominio necesarios que se hayan podido mezclar inicialmente, y así separar por completo la presentación.

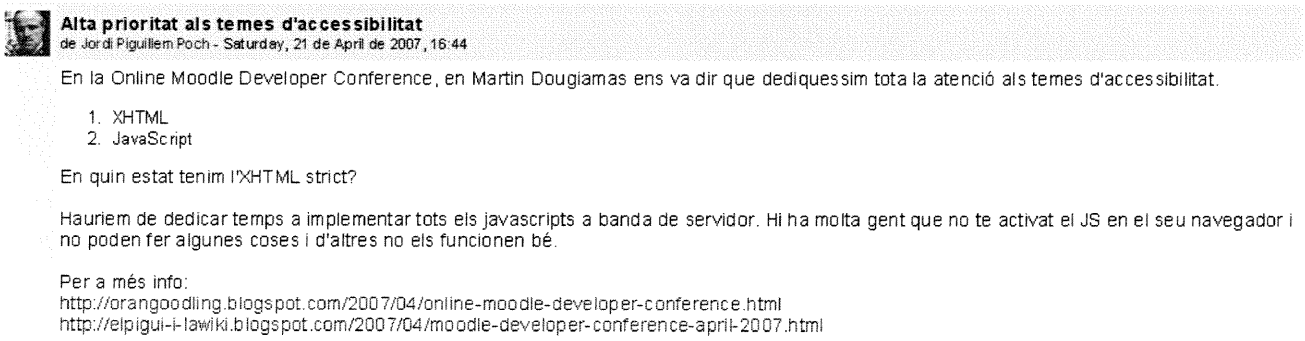
Tras varias pruebas decidimos que el enfoque que habíamos dado para crear este fichero no era el correcto ya que eran muchas funciones las que teníamos que copiar y al llamarse unas a otras nos era imposible separar por completo el contenido de la presentación.

Decidimos finalmente seguir otro proceso, creando un fichero que simplemente contendría una serie de funciones para crear las diferentes etiquetas de HTML que utilizamos en la NWiki, teniendo como parámetros únicamente propiedades de los tag que sigan el estándar XHTML Strict. De este modo siempre que se utilizase el weblib se estaría cumpliendo el estándar.

Una vez creadas todas las funciones teníamos que modificar todo el código HTML de la capa de dominio por las funciones del fichero weblib. Finalmente hicimos también un manual sobre el weblib para los compañeros del DFWiteam.

6.5. ETAPA 5: XHTML Strict

Cuando ya habíamos acabado la implementación de la capa de presentación y lo habíamos subido al CVS para que los compañeros del DFWikiteam lo probasen y ya estábamos empezando a pasar el código HTML a las funciones del weblib surgió un improvisto. Sucedió que se había hecho una reunión con los de Moodle y éstos habían dicho que como no cumpliese la NWiki el estándar XHTML strict en un plazo corto de tiempo no tendríamos posibilidades de ingresar en Moodle.



Alta prioritat als temes d'accessibilitat
de Jordi Piguillem Poch - Saturday, 21 de April de 2007, 16:44

En la Online Moodle Developer Conference, en Martin Dougiamas ens va dir que dediquessim tota la atenció als temes d'accessibilitat.

1. XHTML
2. JavaScript

En quin estat tenim l'XHTML strict?

Hauriem de dedicar temps a implementar tots els javascripts a banda de servidor. Hi ha molta gent que no te activat el JS en el seu navegador i no poden fer algunes coses i d'altres no els funcionen bé.

Per a més info:
http://orangoodling.blogspot.com/2007/04/online-moodle-developer-conference.html
http://elpigui-i-lawiki.blogspot.com/2007/04/moodle-developer-conference-april-2007.html

Ilustración 9: conferencia Moodle

Por este motivo dejamos de lado la nueva capa de presentación y nos pusimos de lleno a hacer que la NWiki cumpliese el estándar sin pasarlo previamente a las funciones de la nueva capa de presentación.

Seguimos el mismo procedimiento que con el estándar XHTML Transitional. Nos repartimos el trabajo en dos partes iguales. En este caso solo se nos pidió que hiciésemos la parte principal de la NWiki, es decir las pestañitas y el parser del editor. El resto lo podíamos dejar para más tarde.

Utilizamos el validador W3C como en el caso anterior y con la ayuda de esta herramienta conseguimos que las partes principales de la NWiki cumpliesen el estándar.

Esta vez nos fue mucho más fácil ya que del transicional al strict la mayor y casi única diferencia es que se tiene que separar completamente las etiquetas de formato del contenido (el formato se apoya exclusivamente en la CSS). A parte de las etiquetas de formato, también habían algunas propiedades de las etiquetas que ya no se podían utilizar como por ejemplo el atributo align para las tablas. Todas esas etiquetas y propiedades de las etiquetas que no cumpliesen el strict se tenían se tenían que modificar cambiándolas por la propiedad llamada style o la clase en una hoja de estilo. Nosotros al no tener tiempo utilizamos inicialmente los styles y no hicimos hojas de estilo ya que cumplía con el estándar XHTML Strict igualmente.

- Ejemplo:

no cumple strict: <table align = “center”>

cumple strict modificado con la propiedad style:
<table style = “margin-left:auto; margin-right:auto” >

De este modo, para cada atributo no válido, se tenía que buscar su correspondiente style.
Por otro lado hicimos un gran trabajo de simplificación de código HTML ya que nos dimos cuenta que en la versión 1.8 de Moodle se utilizaban sobre todo el tag DIV, que define una división en un documento, en vez tablas. En la NWiki utilizábamos muchas tablas dentro de otras y eso complicaba mucho el código, así que lo modificamos para que como en Moodle se utilizasen DIV siempre que fuera posible.

6.6. ETAPA 6: Última versión de Weblib y css

Después de que la parte principal de la Wiki cumpliese el strict nos pusimos otra vez con la capa de presentación. Gracias a que ya la habíamos subido al CVS y los compañeros la habían probado, nos dimos cuenta de que el fichero resultaba muy ineficiente. A cada función se le pasaba como parámetros los atributos del tag que se quería escribir, esto provocaba que cada vez que se quería hacer una modificación añadiendo un atributo en la función de un tag, se tenían que cambiar todas las llamadas a esa función ya realizadas en la capa de dominio.
Como era imposible poner como parámetros todos los atributos de un tag en su correspondiente función decidimos modificar el weblib para que todas las propiedades de los tags estuviesen en un mismo objeto pasado como parámetro. De esta manera cada vez que se hiciese un cambio en alguna función no se verían afectadas las llamadas a esa función. Como se puede ver en el siguiente ejemplo, la llamada a la función es mucho más sencilla.

- Ejemplo:

Antes la cabecera de la función inicio tabla era así:
wiki_table_start($border=0, $width='', $padding='', $spacing='', $class='', $style='', $aligntd='', $valigntd='', $colspantd='', $rowspantd='', $classtd='', $styletd='', $return=false)

Ahora:
wiki_table_start($propert, $return=false)
Donde $propert contiene todas las propiedades del tag.

Finalmente nos encargaron la tarea de modificar las hojas de estilo que utiliza la NWiki (nwiki.css y wiki_tree.css) unificándolas en un único fichero y ampliándolas para que no quedase ningún style en el código ya que también era un requisito de Moodle, todo los estilos tenían que estar en un fichero aparte definidos como clases. Decidimos que esta tarea la haría mi compañero mientras yo acaba de pasar todo el código que quedaba de la NWiki a las funciones del weblib, haciendo que cumpliese el Strict en las partes que no habíamos tocado al principio y modificando el weblib si era necesario para adaptarlo completamente del todo a la NWiki.

7. PLANIFICACIÓN Y ANÁLISIS ECONÓMICO GLOBAL

En este apartado se expondrá un diagrama que muestre el tiempo que se ha tardado en realizar el proyecto y las principales etapas que ha tenido.

Por otro lado se hará un pequeño presupuesto del coste económico según las horas dedicadas. El precio base del proyecto será 40 euros la hora.

7.1. DIAGRAMA DE GANTT

Aquí se muestra con un diagrama de Gantt el tiempo que se ha dedicado a las diferentes tareas del proyecto.

El proyecto se ha dividido en las siguientes tareas:

1. Aproximación a Moodle y NWiki
2. Formación sobre la Accesibilidad Web
3. XHTML Transitional
4. 1ª versión de la capa de presentación (weblib.php)
5. XHTML Strict del código principal
6. XHTML Strict de la NWiki utilizando weblib.php
7. Nueva versión de weblib.php
8. Utilización de la hoja de estilos de la NWiki

Los datos que ofrece el diagrama son la duración de las tareas, no teniéndose en cuenta para la suma los fines de semana y la fecha de comienzo y fin de la tarea de cada una de ellas.
También se puede visualizar las dependencias entre tareas. Estas dependencias están indicadas con una flechita. Que una tarea sea dependiente de la otra significa que hasta que no haya acabado una no puede comenzar la otra.

Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin
1	Aproximación a Moodle y Nwiki	35 días	jue 21/12/06	mié 07/02/07
2	Formación sobre accesibilidad Web	11 días	jue 08/02/07	jue 22/02/07
3	XHTML Transitional	30 días	vie 23/02/07	jue 05/04/07
4	1ª versión de la capa de presentación (weblib.php)	12 días	jue 05/04/07	vie 20/04/07
5	XHTML Strict del código principal	8 días	lun 23/04/07	mié 02/05/07
6	XHTML Strict de la Nwiki utilizando weblib.php	33 días	mié 02/05/07	vie 15/06/07
7	Nueva versión de weblib.php	16 días	vie 11/05/07	vie 01/06/07
8	Utilización de la hoja de estilos de la Nwiki	10 días	lun 04/06/07	vie 15/06/07

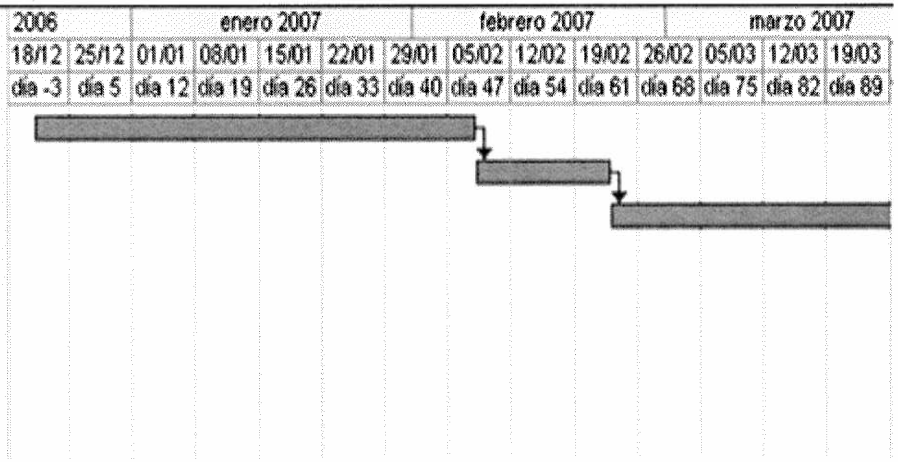
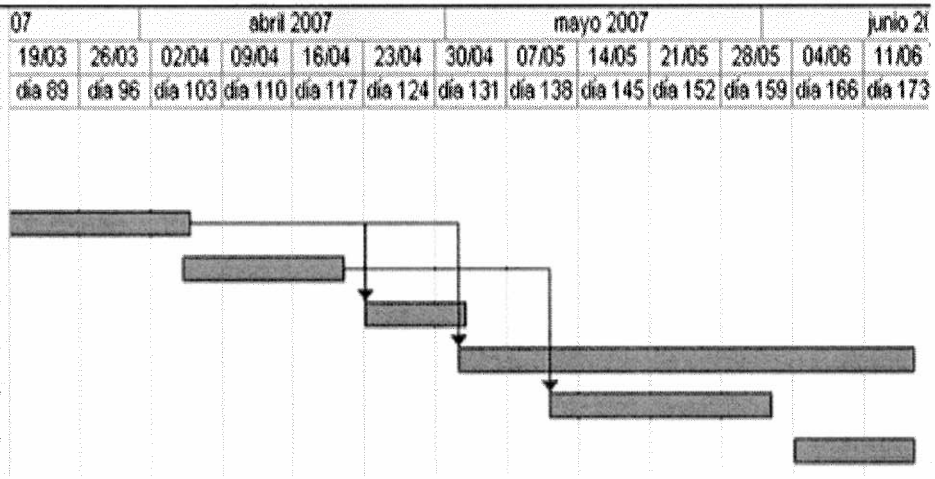


Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin
1	Aproximación a Moodle y Nwiki	35 días	jue 21/12/06	mié 07/02/07
2	Formación sobre accesibilidad Web	11 días	jue 08/02/07	jue 22/02/07
3	XHTML Transitional	30 días	vie 23/02/07	jue 05/04/07
4	1ª versión de la capa de presentación (weblib.php)	12 días	jue 05/04/07	vie 20/04/07
5	XHTML Strict del código principal	8 días	lun 23/04/07	mié 02/05/07
6	XHTML Strict de la Nwiki utilizando weblib.php	33 días	mié 02/05/07	vie 15/06/07
7	Nueva versión de weblib.php	16 días	vie 11/05/07	vie 01/06/07
8	Utilización de la hoja de estilos de la Nwiki	10 días	lun 04/06/07	vie 15/06/07



7.2. VALORACIÓN ECONÓMICA POR HORAS DE TRABAJO

Para valorar económicamente la totalidad del proyecto, he cogido cada tarea realizada que se puede ver en el diagrama de Gantt y he hecho un análisis aproximado de las horas dedicadas.

Tarea 1: En el diagrama de Gantt se puede visualizar que esta tarea de aproximación a Moodle y NWiki duró 30 días, pero es una cifra engañosa ya que era el mes de diciembre. En total se le dedicó a esta tarea unas **40 horas**.

Tarea 2: Las cifras del Diagrama de Gantt también son engañosas respecto a la tarea de formación sobre la Accesibilidad Web ya que todavía no habíamos comenzado la universidad. De los 11 días que duró la tarea se le dedicaron unas **20 horas** aproximadamente.

Tarea 3: A partir del comienzo de la realización de la tarea de XHTML Transitional, se dedicó 20 horas semanales al proyecto. Concretamente esta tarea se divide en tres fases:

- Hubo un periodo de formación sobre XHTML Transitional que duró 3 días
- Se estuvo modificando todo el código de la NWiki para que cumpliese XHTML Transitional durante 25 días.
- Por último se hizo un manual de XHTML para que los compañeros del DFWikiteam tuviesen unas pautas para que las NWiki esté bien formada. Tardamos 2 días en realizarlo.

Esta tarea duró 30 días, que suman un total **150 horas**, habiéndole dedicado 5 horas al día.

Tarea 4: Para la 1ª versión de la capa de presentación (weblib.php) se dedicó un total de 12 días dedicándole también 5 horas diarias, es decir **60 horas**.

- Estuvimos 3 días analizando el fichero weblib de Moodle.
- Estuvimos otros 3 días haciendo pruebas y pensando en como estructurar el weblib.
- En realizar la primera versión del weblib.php con todas sus funciones se tardó 4 días.
- Las pruebas de código duraron 2 días.

Tarea 5: Para pasar el código principal a XHTML Strict se tardó 8 días, sumando un total de **40 horas**.

Tarea 6: Para pasar el código a XHTML Strict de la NWiki utilizando las funciones del weblib.php se tardó 33 días. Esta tarea se hizo conjuntamente con la creación de la nueva versión del weblib.php, por ese motivo no se le dedicaron 5 horas diarias. Haciendo un cálculo, las horas aproximadas que se le dedicaron en total fueron **120 horas.**

Tarea 7: Para crear la nueva versión de weblib.php se tardó 16 días en los cuales se estuvo trabajando un total de **40 horas** para la realización de esta tarea.

Tarea 8: Por último estuve aproximadamente un total de **5 horas** repartidas en 10 días para la utilización de la hoja de estilos de la NWiki en todo el código.

Después de analizar el tiempo utilizado en la realización de cada tarea se estima que se ha trabajado un total de **475 horas**. Eso supone un coste total de **19000 euros** suponiendo que una hora de trabajo se remunera a 40 euros.