UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FINAL DE CARRERA
Ingeniero en Telecomunicación.
Departamento de Telemática.

Diseño y desarrollo de una herramienta para la publicación electrónica y visualización de calificaciones de evaluación continua.

Autor: Sandra Pinero Sánchez

Tutora: Iria Estévez Ayres

Leganés, 2013

"La paciencia es la fortaleza del débil y la impaciencia, la debilidad del fuerte."

Inmanuel Kant

Agradecimientos

A pesar de todo lo pasado y sufrido para llegar aquí es irónico que lo que más me cueste sea dar las gracias a todas aquellas persona que me han ayudado a lograrlo.

En primer lugar quiero dar las gracias a mis abuelos. A mi abuela Amparo por todo lo que se ha gastado en velas para cada uno de mis exámenes y por todas esas llamadas de preocupación cada día, a mi abuela Eutimia por pensar que había acabado la carrera cada vez que terminaba un curso, gracias por creer que era tan "máquina", a mi abuelo German, por no creer nunca que "aún no había terminado" y por su gran apoyo para seguir, y en especial a mi abuelo Dionisio porque me hubiera encantado poder decirle "abuelo ya he terminado".

En segundo lugar quiero agradecérselo a mis padres, por soportar lo duro que ha sido todo para mí, por mis cambios de humor y por mis lágrimas y a Guillermo, por hacer que cada examen fuera más difícil y por ello que la victoria supiera más a gloria, gracias pequeño.

A mis tíos y tías, por la preocupación y el interés mostrado todos estos años y por siempre tener alguna palabra de apoyo para mí.

A la persona que ha hecho posible este largo camino, Roberto. No tengo palabras para agradecerte tanto, no tengo hojas para escribir todo lo que has significado para mí estos años, tanto en lo personal como en lo profesional.

Gracias por levantarme cuando caía, por prevenir caídas, por enseñarme que valgo mucho, que puedo conseguir todo lo que me proponga en la vida y que a veces la clave es ir poco a poco. Te quiero.

Está claro que como toda buena carrera no se gana si no tienes un buen equipo y yo me lleve al mejor. Esto no habría tenido sentido sin mis grandes acompañantes en este viaje: Ainhoa, Alba, Carmen, Dani, Gonzalo y Tania. Gracias por tantas risas día a día, gracias por tanto apoyo en los peores momentos de mi vida y por vuestra compañía en los mejores. Nunca olvidaré ni uno de los días vividos dentro y fuera de las paredes de la universidad, para mí los mejores años, y vosotros lo mejor que me llevo.

Gracias Natalia y Rocío, por sacarme siempre de la rutina y hacerme olvidar un poco esto.

Gracias a mi inseparable Flor, por esas llamadas que arreglaban nuestros mundos y por siempre estar ahí aún en la distancia.

Para finalizar y no por ello menos importante, quiero dar las gracias a mi tutora Iria Estévez, por hacer del proyecto un camino menos amargo, por enseñarme que la locura y la inteligencia son buenas amigas.

A mis compañeros de INECO: Agus, Alejandro, David, Jesús, Lidia, Natán y Silvia, por hacerme un hueco en vuestras vidas, por ser amigos y compañeros de trabajo, por apoyarme en el final de esta historia, por alegrar los días de oficina y por tantas risas después. Gracias por todo.

Y en general, a todos los que un día os preocupasteis por mí, muchísimas gracias.

Sandra.

Resumen

El presente proyecto tiene como objetivo el desarrollo y la implementación de una herramienta que sea capaz de ayudar en la gestión y la consulta de las asignaturas impartidas en la Universidad Carlos III de Madrid tanto a profesores como a alumnos.

Se propone el diseño de una aplicación web en la que se mostraran de forma sencilla y práctica las calificaciones obtenidas por los alumnos en sus exámenes.

El proyecto consta de los grandes bloques. Por un lado el diseño de la aplicación web que se implementará en las tecnologías de HTML5, CSS y JavaScript, y por otro lado, la creación de un servidor que contiene una base de datos que nos ayudará a gestionar los diferentes datos.

ÍNDICE GENERAL

Agradecimientos 2

Resumen 4

Capítulo 1. Introducción 13

1.1. Web 2.0 18

1.1.1 Ventajas de la Web 2.0 19

1.1.2 Desventajas de la Web 2.0 20

1.2 Estructura del Documento 14

Capítulo 2. Estado del arte 16

2.1. Introducción 16

2.2 Lenguajes de programación para la Web 20

2.2 Servidores Web 25

Capítulo 3. Tecnologías empleadas 29

3.1. Arquitectura del proyecto 29

3.2. Cliente 30

3.2.1. HTML5 31

3.2.2. CSS3 34

3.2.3. JavaScript 35

3.3. Capa intermedia o middle-tier 35

3.3.1 Servidor de Servlets....36

3.3.2 JSON....37

3.4. Servidor de datos....39

3.4.1. Tomcat....40

3.4.2. MySQL....40

3.4.3 Otras herramientas....41

Capítulo 4. Diseño e implementación de la aplicación....44

4.1. Cliente....44

4.2. Servidor....56

4.2.2 Servlet....59

4.3. Base de datos....60

Capítulo 5. Validación....68

5.1. Instalación del servidor....68

5.2. Conclusiones....84

Capítulo 6. Conclusiones, Líneas futuras y mejoras....85

6.1. Conclusiones....85

6.2. Líneas futuras....86

Capítulo 7. Presupuesto....88

Capítulo 8. Referencias....91

ÍNDICE DE FIGURAS

Figura 1: Web 1.0 Vs Web 2.0 19
Figura 2: Modelo OSI 17
Figura 3: Comparación, pila OSI vs pila TCP/IP 17
Figura 4: Ejemplo de lenguaje HTML 21
Figura 5: Arquitectura cliente-servidor 30
Figura 6: Jerarquía de clases y principales métodos para crear un servlet 36
Figura 7: Ejemplo de XML 38
Figura 8: Ejemplo *JSON*. 39
Figura 9: Entorno Apache Tomcat. 40
Figura 10: Entorno eclipse. 42
Figura 11: Entorno Navicat Lite 43
Figura 12: Diseño de las pantallas Web 45
Figura 13: Diagrama de estados de la aplicación Web 47
Figura 14: Tabal perfiles base de datos MySql 48
Figura 15: Autenticación entre cliente servidor 49
Figura 16: Diagrama de flujo de la página principal 51
Figura 17: Diagrama de flujo de la visualización de notas 53
Figura 18: diagrama de flujo de la página de medias 55
Figura 19: Mensajes página de gestión 55
Figura 20: Ciclo de funcionamiento del servlet. 57
Figura 21: Ciclo de vida del servlet 57
Figura 22: Esquema base de datos 61

Figura 23: Relaciones usuarios base de datos 62
Figura 24: Relaciones usuario - examen.....63
Figura 25: Relaciones usuario – materia.....64
Figura 26: Relación de la tabla materia.....65
Figura 27: Relaciones exámenes apartados 66
Figura 28: Relación materia - *url*.....67
Figura 29: Entorno Virtual Box 69
Figura 30: Apache Tomcat 70
Figura 31: Imagen de la página de login.html 72
Figura 32: Página de inicio (1) 73
Figura 33: Menú principal 73
Figura 34: Página de inicio (2) 74
Figura 35: Página notas.html 74
Figura 36: Pop-up de las notas.....75
Figura 37: Menú media.html 75
Figura 38: Página media.html 76
Figura 39: Página media.html (2).....76
Figura 40: Página media.html (3).....77
Figura 41: Página gestion.html.....78
Figura 42: Añadiendo un examen a la base de datos.....79
Figura 43: Añadiendo apartados a un examen.....80
Figura 44: Añadiendo la nota del examen de un alumno.....81
Figura 45: Añadiendo datos del alumno 82
Figura 46: Añadiendo *url* a la base de datos 83
Figura 47: Modificando datos del alumno.....84

ÍNDICE DE TABLAS

Tabla 1: Ventajas e inconvenientes de javascript 22

Tabla 2: ventajas e inconvenientes de php 23

Tabla 3: ventajas e inconvenientes de aps.net 24

Tabla 4: ventajas e inconvenientes de jsp 25

Tabla 5: Directorios tomcat 26

Tabla 6: Conclusiones finales apache tomcat e iss 28

Tabla 7: Etiquetas HTML5 33

Acrónimos

[1]	XHTML - eXtensible HyperText Markup Language
[2]	CSS - Cascading Style Sheets
[3]	XML - eXtensible Markup Language
[4]	URL - Uniform Resource Locator
[5]	PDAs - Asistente Digital Personal
[6]	PC - Personal Computer
[7]	HTML5 - HyperText Markup Language, versión 5
[8]	CSS3 - Cascading Style Sheets, versión 3
[9]	XSL - Extensible Stylesheet Language
[10]	W3C - World Wide Web Consortium
[11]	JSON - JavaScript Object Notation
[12]	PHP - PHP Hypertext Pre-processor
[13]	WWW - World Wide Web
[14]	TCP/IP - Transmission Control Protocol / Internet Protocol
[15]	OSI - Open System Interconnection
[16]	HTTP - Hypertext Transfer Protocol
[17]	FTP - File Transfer Protocol
[18]	JDBC - Java DataBase Connectivity
[19]	JSP - JavaServer Pages
[20]	OBDC - Open DataBase Connectivity
[21]	IIS - Internet Information Services

Capítulo 1.

Introducción

1.1 Estructura del Documento

Actualmente las universidades se rigen mediante el plan Bolonia, que consiste en unificar los contenidos y la evaluación a todos los centros europeos.

Mediante este programa el alumno tendrá varios exámenes parciales que le ayudarán a la superación de la asignatura, así como prácticas y trabajos.

Es por esto que se hace necesaria una herramienta que ayude a los profesores y a los alumnos a llevar la continuidad de las asignaturas cursadas e impartidas.

El principal objetivo de este proyecto es proporcionar una herramienta a la comunidad universitaria que ayude de forma gráfica y sencilla a llevar al día las asignaturas impartidas en la universidad y proporcione a los alumnos materiales para prepararlas.

Lo que se pretende conseguir con esto, es que el alumno pueda llevar al día sus calificaciones y además ayudarles a conseguir un buen aprovechamiento académico.

El objetivo del diseño a implementar es conseguir una herramienta que sea fácil de usar y que ayude al alumno en el seguimiento de sus asignaturas,

pero también, que sea útil para el profesorado a la hora de llevar al día sus asignaturas impartidas.

Para ello nuestro diseño tiene que ser fácil y accesible por toda la comunidad universitaria y debe mostrar de una forma sencilla y rápida los datos necesarios para cada usuario.

1.2 Estructura del Documento

El presente documento ha sido organizado de la siguiente manera.

I. Capítulo 2: Estado del arte.

Se explica el marco del proyecto.

II. Capítulo 3: Tecnologías empleadas.

Se explican las herramientas y tecnologías empleadas para la elaboración del presente proyecto final de carrera.

III. Capítulo 4: Diseño e Implementación de la aplicación.

En este capítulo se detalla el diseño de las diferentes partes del proyecto y la implementación que se ha seguido para cada una de ellas, así como los programas utilizados para ello.

IV. Capítulo 5: Validación.

En este capítulo se mostraran capturas de pantalla que demuestren el funcionamiento del proyecto final.

V. Capítulo 6: Líneas futuras y mejoras.

En este capítulo hablaremos sobre los posibles cambios a realizar en la aplicación de cara al futuro así como mejoras que pudieran realizarse.

VI. Capítulo 7: Presupuesto

En este capítulo se muestra un presupuesto del coste del proyecto.

Capítulo 2.

Estado del arte

En este capítulo se van a analizar las diferentes tecnologías que pueden aplicarse para el desarrollo de la aplicación web.

2.1. Introducción

WWW (World Wide Web), es un sistema de información completo y actual que combina tanto hipertexto como elementos multimedia y que es la base de Internet.

Los protocolos por excelencia de Internet son TCP e IP, más comúnmente conocidos por TCP/IP (Transmission Control Protocol/Internet Protocol).[3][4]

Son un conjunto de protocolos que forman la base del funcionamiento de Internet, y los que permiten que se puedan comunicar diferentes ordenadores aun que utilicen diferentes sistemas operativos.

La familia de protocolos de Internet puede describirse con el modelo de protocolos OSI (Open Systems Interconnect)[4], modelo de arquitectura desarrollado por la ISO (International Standards Organization). Este modelo se denomina Modelo de Referencia OSI (Figura 2 Y 3).

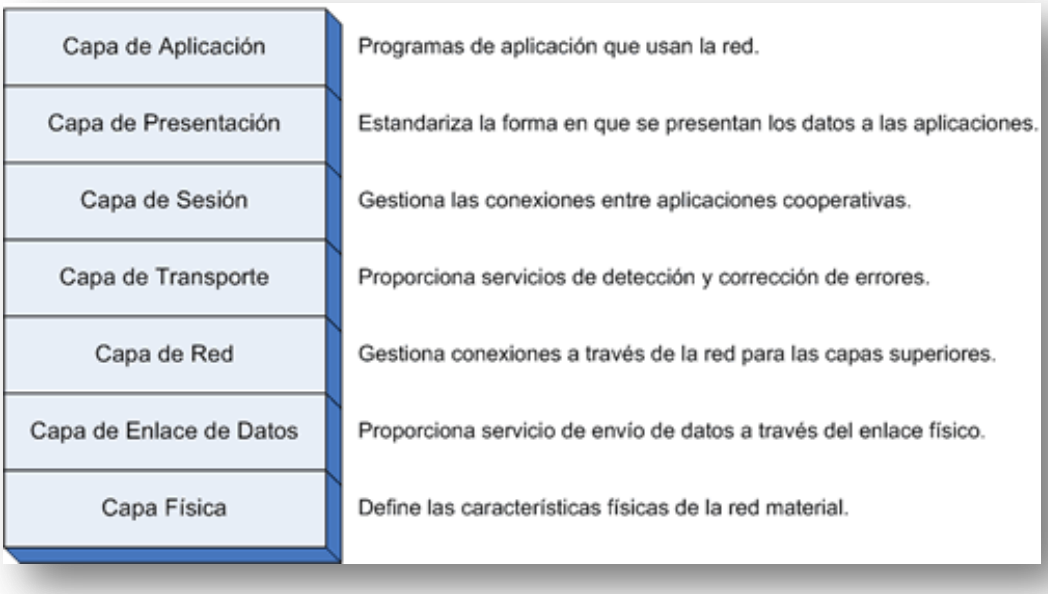


FIGURA 1: MODELO OSI

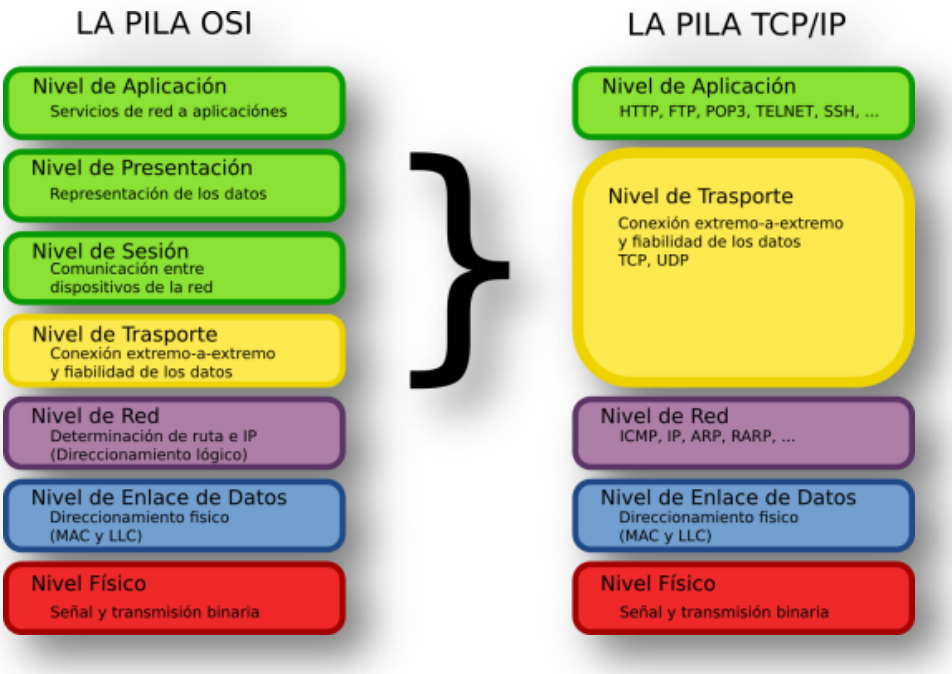


FIGURA 2: COMPARACIÓN, PILA OSI VS PILA TCP/IP

2.2. Web 2.0

La Web nació de la mano de Tim Berners-Lee el inventor del hipertexto. En su comienzo eran solo textos e imágenes "colgadas" o "subidas" en un ordenador, el cual, permanecía conectado a la red todo el día para que la gente pudiera entrar en ella en cualquier momento. A estos ordenadores los llamaron servidores.

Paralelamente a esto, comenzaron a aparecer tecnologías del tipo aplicaciones server-side, programas que funcionan en máquinas que alojan las páginas webs e interactúan con ellas.

También comenzaron a surgir los estándares XHTML, CSS o JavaScript, que hacían una web más rica en diseño, funcionalidad etc.

Por aquel entonces las páginas no se actualizaban muy a menudo y los usuarios no tenían la posibilidad de participar como creadores de los contenidos que se mostraban. A esta web es a la que llamamos Web 1.0.

La Web 2.0[1][2] representa una evolución de las aplicaciones que ya se conocían, a otras enfocadas al usuario final. En este modelo de trabajo el usuario pasa a ser parte activa de la web y no solo un mero lector como era anteriormente. (Figura 1)

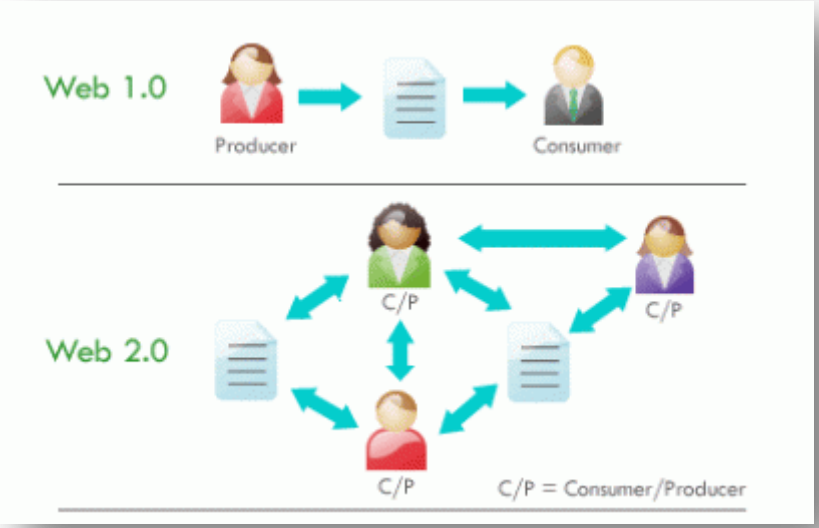


FIGURA 3: WEB 1.0 VS WEB 2.0

Algunas de las tecnologías que apoyan esta web son:

- Convertir software de escritorio en plataforma de web.
- Respetar los estándares como el XHTML.
- Separación de contenido del diseño con uso de hojas de estilo.
- Dar control total a los usuarios en el manejo de su información.
- Proveer APIs o XML para que las aplicaciones puedan ser manipuladas por otros.
- Facilitar el posicionamiento con URL sencillos.

1.1.1 Ventajas de la Web 2.0

- Permitir que el desarrollo de nuevas aplicaciones se centren en la aportación de valor añadido, centrando los recursos en lo nuevo, y sacando partido de lo hecho por otros.
- Aplicaciones como servicio y no como producto.
- Las aplicaciones basadas en Web pueden desarrollarse en cualquier terminal (y no necesariamente en los PC): ordenadores, móviles,

PDAs, TV digital, etc. Esto va a permitir tener la información en todo momento y desde cualquier terminal con conexión a Internet.

- Las necesidades de la sociedad y empresa red radican principalmente en la cooperación entre los diferentes actores.

1.1.2 Desventajas de la Web 2.0

- La seguridad de datos confidenciales, como la contabilidad, facturación, etc.
- La conexión a Internet. La dependencia del sistema a la conexión de Internet sigue siendo una barrera a su adopción.

2.3 Lenguajes de programación para la Web

La demanda de Internet por parte de los usuarios y de diferentes plataformas y tecnologías ha hecho que hoy en día contemos con un gran número de lenguajes que nos son útiles para la creación de entornos web.

- HTML

Para que nuestra Web pueda funcionar se utiliza el protocolo HTTP (HyperText Transfer Protocol) y está basado en HTML (HyperText Markup Language)[5], una herramienta fundamental para el uso de hipertexto en Internet. (Figura 4)

Gracias al hipertexto nuestra página web podrá acceder a cualquier otra página mediante el protocolo HTTP y todo esto se hace mediante programas

como el navegador que recoge el código en HTML lo procesa y lo representa en nuestras pantallas de la mejor manera posible.

Una de las características fundamentales del protocolo HTTP es que no es permanente, es decir, una vez que se ha hecho una petición al servidor y éste ha dado una respuesta se cierra la conexión y el servidor permanece a la espera hasta recibir una nueva petición, esto no ocurre por ejemplo, con protocolos como FTP (File Transfer Protocol), en este caso la conexión permanece abierta hasta que el usuario o el propio servidor mandan la orden de desconectar.

Que la conexión no sea permanente hace que sea necesaria la autenticación de usuarios, ya que el servidor es capaz de reconocer si la petición llega de un usuario aceptado o no, a esto se le llama mantener la sesión.

```
<html> (Inicio del documento HTML)
<head>
( Cabecera )
</head>
<body>
( Cuerpo )
</body>
</html>

<b>  </b> Negrita
<p>  </p> Definir parrafo
<etiqueta> Apertura de la etiqueta
</etiqueta> Cierre de la etiqueta
```

FIGURA 4: EJEMPLO DE LENGUAJE HTML

- JavaScript

Para añadirle inteligencia a nuestra página web hemos utilizado JavaScript[7][8] cuyo código fuente se introduce dentro del código HTML mediante tags <SCRIPT>…</SCRIPT>

El código JavaScript se puede incluir dentro del archivo HTML o fuera, en archivos con extensión .js que luego habrá que incluir en el código HTML.

```
<script type="text/javascript" src="./generic.js"></script>
```

Ventajas	Inconvenientes
Lenguaje de scripting seguro y fiable.	**El código debe descargarse completamente**
Código que se ejecuta en el cliente	**Puede tener problemas de seguridad.**

TABLA 1: VENTAJAS E INCONVENIENTES DE JAVASCRIPT

- PHP

Inicialmente se llamó "Personal Home Page" y surgió en 1995 desarrollado por PHP Group.

Lenguaje de programación utilizado del lado del servidor para la creación del sitio web dinámicas embebidas en páginas HTML y ejecutadas en el servidor.

Para su funcionamiento no necesita compilarse, solo tener instalado Apache Tomcat o IIS y las librerías de PHP.

La sintaxis utilizada para la incorporación del código PHP es la siguiente:

```
<?
$mensaje = “Hola”;
Echo $mensaje;
?>
```

Ventajas	Inconvenientes
Fácil y rápido	**Requiere servidor web**
Lenguaje multiplataforma: Linux, Windows, etc.	**Todo el trabajo está en la parte del servidor y no delega nada en el cliente.**
Capacidad de conexión a la mayoría de los manejadores de base de datos	**Dificultad a la hora de la organización de la aplicación por capas**
Se puede expandir utilizando módulos	**La programación orientada a objeto aun es deficiente.**
Es libre, por lo que es de fácil acceso para todos	

TABLA 2: VENTAJAS E INCONVENIENTES DE PHP

- ASP.NET

Lenguaje usado por programadores para desarrollar entre otras cosas, sitios web. Es el sucesor de la tecnología ASP y fue lanzado al mercado bajo una estrategia llamada .NET.

Para su funcionamiento se necesita tener instalado IIS con el Framework .NET.

Ventajas	Inconvenientes
Es orientado a objetos	**Consume muchos recursos**
Controles de usuarios	
División entre la capa de diseño y el código	
Fácil mantenimiento en grandes aplicaciones	
Alta velocidad y seguridad	

TABLA 3: VENTAJAS E INCONVENIENTES DE APS.NET

- JSP

Lenguaje para la creación de sitios web dinámicos.

Es acrónimo de 'Java Server Pages. Está orientado a desarrollar páginas web en Java.

Es un lenguaje multiplataforma que se ejecuta en el lado del servidor y posee un motor de páginas basado en servlets de Java.

Para su funcionamiento se necesita tener instalado un servidor Tomcat.

Ventajas	Inconvenientes
Ejecución rápida de servlets	**Complejo aprendizaje**
Crear páginas del lado del servidor	

Multiplataforma
Código bien estructurado
Ejecución rápida de servlets
Integridad con los módulos de Java
La parte dinámica está escrita en Java

TABLA 4: VENTAJAS E INCONVENIENTES DE JSP

2.2 Servidores Web

- Apache Tomcat

Apache 2.0 [9][10] fue lanzado al mercado a mediados de abril 2002.

Plataformas:

NetBSD, Digital UNIX, AIX, OS/2, Windows 3.x, Novell NetWare, Macintosh, Windows NT, Linux, Windows, Solaris etc.

Ventajas Principales:

- Cantidad de plataformas soportadas y disponibilidad de código fuente.
- Modularidad.
- Coste. Gratis.

A continuación vamos a ver los diferentes directorios de los que se compone Tomcat. (Tabla 5)

Para el correcto funcionamiento de nuestra página hemos desplegado la web en Tomcat.

Para ello debemos saber que Tomcat cuenta con una extensa variedad de directorios entre ellos:

Directorio Tomcat	Características
/bin	Contiene los scripts y códigos requeridos para que se ejecute el servidor. Es el principal utilizado para arrancar y parar Tomcat.
/conf	Contiene los ficheros necesarios para configurar y administrar Tomcat, es lo primero que lee al arrancar y se basa en su contenido para cargar el resto.
/lib	Contiene todos los JAR (Java ARchive) que usa el servidor.
/logs	El directorio usado para logging, es el que más vamos a consultar en sistemas.
/temp	Ficheros temporales.
/webapps	Donde desplegamos las aplicaciones, aquí ponemos los war y al reiniciar el Tomcat según la configuración que tengamos hará un despliegue.

TABLA 5: DIRECTORIOS TOMCAT

- IIS

Internet Information Services es el servidor web de la familia Microsoft

Plataformas:

- Windows NT
- Windows 2000

Ventajas Principales:

- ASP
- Soporte para los APIs de Microsoft
- ODBC

Como conclusión de ambos servidores podemos ver la tabla 6:

Apache Tomcat	IIS
Ventajas	
Proyecto Open Source bajo constante desarrollo	**Fácil de utilizar**
ODBC disponible sin costo	**Soporte ODBC integrado**
Gran cantidad de opciones	**Poco o nada de configuración necesaria**
Extensible	**Interfaz grafica**
Desventajas	
Configuración inicial necesaria	**Cualquier problema serio puede requerir una**

	reinstalación
La curva de aprendizaje para uso efectivo del servidor es muy empinada	Los módulos y utilidades adicionales deben ser compradas

TABLA 6: CONCLUSIONES FINALES APACHE TOMCAT E ISS

Capítulo 3.

Tecnologías empleadas

En este capítulo se explican las tecnologías que se han utilizado para la realización del proyecto.

3.1. Arquitectura del proyecto

La arquitectura del proyecto es una arquitectura a tres niveles. Con esta arquitectura tenemos la ventaja de tener todo centralizado, es decir, no se duplica por aplicación.

La capa intermedia que forma una nueva capa dentro del sistema cliente-servidor se llama “capa intermedia “o “middle-tier”. (Figura 5)

- La primera capa será la capa de presentación, es la capa que actúa con el usuario y a la que llamaremos cliente.

- La capa intermedia será la capa donde corran los programas. Recibirán peticiones de la capa de presentación y las conectará con la capa de datos.
- Por último tenemos la capa de datos, en la cual residen los datos que vamos a manejar.

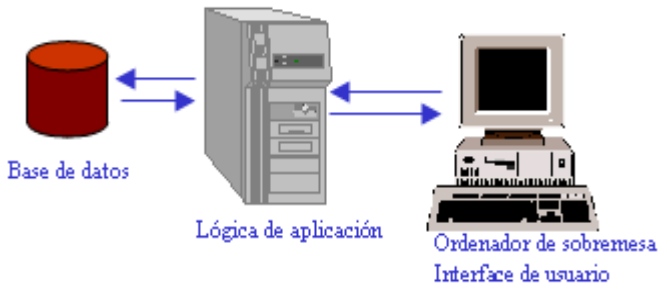


FIGURA 5: ARQUITECTURA CLIENTE-SERVIDOR

3.2. Cliente

En la parte del cliente tenemos una página web como interfaz de usuario que realizará las peticiones al servidor cuando sea necesario por medio de un proceso Java que estará corriendo en la capa intermedia.

Para la realización de la página se ha utilizado HTML5, CSS3 y JavaScript.

Se ha utilizado HTML5[11][12][13] y JavaScript porque se ha diseñado la aplicación Web desde el lado del cliente.

3.2.1. HTML5

HTML5 es la quinta versión de HTML (Hyper Text Markup Language), un lenguaje de programación que se utiliza para el desarrollo de páginas en Internet. Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos.

Su funcionamiento está basado en unas etiquetas. Las etiquetas, o "tags", son elementos dentro del código HTML que dan a la sección de código encerrada entre ellas unas características especiales; tipo de elemento, forma, color, tamaño etc.

Las etiquetas comienzan con <nombre_etiqueta> y finalizan con </nombre_etiqueta>.

A continuación podemos ver las etiquetas más usadas, y las añadidas a la nueva versión de HTML, HTML5.

Etiqueta	Descripción
<!--...-->	Define un comentario
<!DOCTYPE>	Define el tipo del documento
<a>	Define un hipervínculo
<article>Nueva	Define un articulo
<audio>Nueva	Define contenido con sonido
<b>	Define texto en negrita
<bdi>Nueva	Aísla una parte del texto que puede tener un formato diferente del texto externo.

<body>	Define el cuerpo del documento
** **	Define un salto de línea.
<!--...-->	Define un comentario
<!DOCTYPE>	Define el tipo del documento
<a>	Define un hipervínculo
<article>Nueva	Define un articulo
<button>	Define un botón clickleable
<canvas>Nueva	Usada para dibujar gráficos, por script, en el aire. (con JavaScript)
<command>Nueva	Define un botón command que un usuario puede invocar.
<datalist>Nueva	Especifica en un input una lista pre definida de opciones.
<details>Nueva	Define detalles adicionales que el usuario puede ver o esconder.
** **	Define una sección en un documento.
<figure>Nueva	Especifica contenido auto-contenido
<footer>Nueva	Define un pie página footer un documento o sección
<form>	Define un formulario HTML.
<html>	Define la raíz de un documento HTML.
<input>	Define un control de entrada de

	texto
<label>	Define un rótulo para un elemento <input>.
** **	Define un ítem de una lista
<link>	Define la relación entre un documento y un recurso externo (generalmente con hojas de estilo)
<nav>Nueva	Define un link de navegación.
<param>	Define un parámetro para un objeto
<script>	Define script del lado del cliente
<section>Nueva	Define una sección en un documento
<select>	Defines a drop-down list
<table>	Define una tabla
<td>	Define una celda en una tabla
<time>Nueva	Define una fecha/hora
<title>	Define un título para el documento.
<tr>	Define una fila en una tabla

TABLA 7: ETIQUETAS HTML5

3.2.2. CSS3

CSS (Cascading Style Sheets), es un mecanismo sencillo para añadir estilos a los documentos Web escritos en HTML o XHTML, tales como, color, fuentes, espaciado...

La información de estilo (aspecto y formato) puede ir fuera del propio documento html o xhtml, o dentro de los mismos. En nuestro caso esta información estará contenida en un documento independiente "default.css".

Las ventajas de utilizar un lenguaje de estilos, en este caso CSS son:

- La centralización del diseño hace que sea más rápido y fácil a la hora de modificar cualquier aspecto.
- Separa el contenido de la aplicación
- Optimiza el ancho de banda de la conexión ya que el mismo archivo CSS puede servir para más de un documento.
- Mejora la accesibilidad al documento evitando el uso de tablas como se hacía anteriormente.

Además de este mecanismo, también existe XSL. XSL es muy poco usado, aunque es más potente que CSS.

En este proyecto se eligió CSS, ya que como hemos dicho es sencillo de utilizar y fácil de aprender, pero además es el recomendado por W3C (The World Wide Web Consortium). La W3C es una comunidad internacional que desarrolla estándares abiertos para asegurar el crecimiento a largo plazo de la Web que dice: "Use CSS when you can, use XSL when you must."[1]

[1] http://www.w3.org/Style/CSS-vs-XSL

3.2.3. JAVASCRIPT

JavaScript es un potente lenguaje de programación que está integrado en los navegadores web.

Se usa para hacer las páginas web interactivas, por lo que suele ser usado al lado de HTML y CSS.

3.3. Capa intermedia o middle-tier

Para unir la interfaz del usuario (Web) con el servidor, y poder hacer peticiones a la base de datos, además de acceder a ella, se van a utilizar servlets en el lado del servidor.

Con los servlets conseguimos integrar en nuestra página lenguaje Java para poder operar con los datos de la base de datos.

Además para el intercambio de información de forma sencilla crearemos un archivo *JSON* mediante un proceso Java que estará corriendo en el servidor.

Para unificar todo y poder usar los Servlets que hemos creado se ha procedido a la instalación de Tomcat, el cual explicaremos posteriormente en el apartado del servidor.

3.3.1 SERVIDOR DE SERVLETS

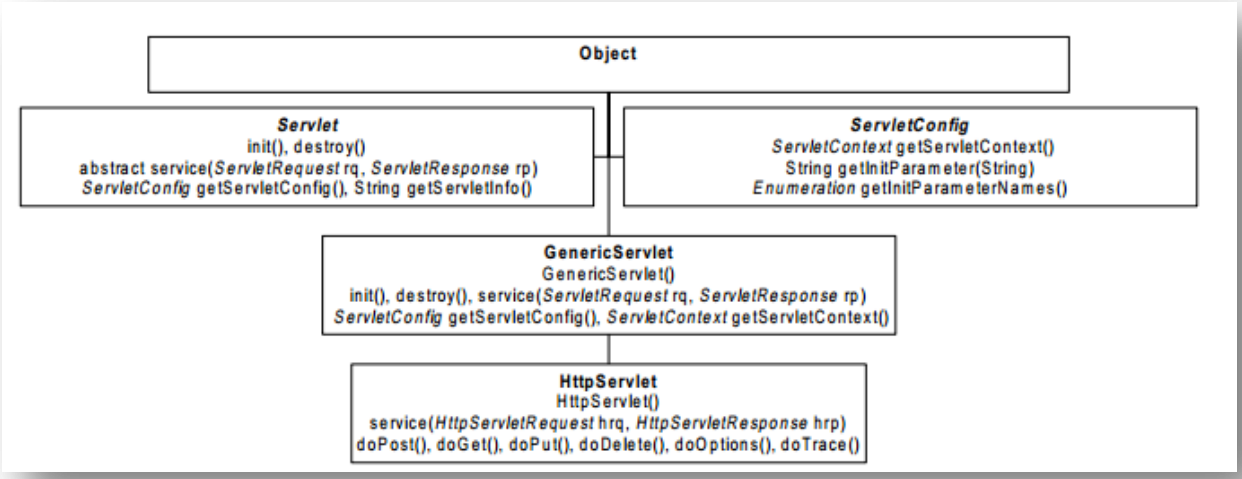


FIGURA 6: JERARQUÍA DE CLASES Y PRINCIPALES MÉTODOS PARA CREAR UN SERVLET

Los servlets son clases Java[14] que extienden normalmente de HttpServlet. (Figura 8).

Un servlet[15] es un módulo escrito en Java que se utiliza en el servidor y que sirve para extender sus capacidades de respuestas a clientes que utilizan el lenguaje Java.

Los servlets no tienen interfaz gráfica, a diferencia de los Applets, que van en el lado del navegador y que en nuestro proyecto no se consideraron por los problemas que daban con la apertura y cierre de conexión.

Además de lo mencionado en cuanto al lenguaje Java, los servlets también tienen las siguientes características:

- Soportan multihilo. Es decir, varios clientes conectados a la vez.
- Manejan sesiones, por lo que se puede hacer el seguimiento de usuarios por sus sesiones si fuera necesario.
- Utilizan cookies.

Una de las tareas más frecuentes realizadas por los servlets, y para las que los necesitamos en este proyecto es para el acceso a base de datos.

Para ejecutar un servlet basta con poner en el navegador

http://servidor:puerto/servlet/nombre_de_servlet

Si se trata de un formulario, como es nuestro caso, hay que indicar en la propiedad ACTION de la <tag> FORM en la página html la dirección *url* del servlet, y especificar el método GET / POST que se va a emplear en la propiedad METHOD.

```
<form name="FORM1" method="GET"
        action="http://localhost:8080/misServlets/LoginServlet">
```

Para ello se necesita un "conector" llamado JDBC (Java DataBase Connectivity).

JDBC es una parte de la API de Java que proporciona una serie de clases para realizar la conexión a la base de datos.

El API JDBC consiste en un conjunto de clases e interfaces que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea.

3.3.2 JSON

La información entre cliente servidor puede in encapsulada de varias formas.

Por ejemplo, podemos usar XML. (Figura 6)

Es un formato desarrollado por el W3C que permite definir la gramática de lenguajes determinados para organizar y gestionas grandes documentos.

Mediante otras tecnologías que complementan a XML, se pueden conseguir desarrollos mucho mayores que nos dan más posibilidades de comunicación, haciendo de ella que se más fiable segura y fácil.

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

FIGURA 7: EJEMPLO DE XML

Como ya hemos dicho, nosotros usaremos *JSON* (JavaScript Object Notation). (Figura 7)

El uso de *JSON* [16][17] es más sencillo que el de XML dado que es más fácil hacer un analizador sintáctico.

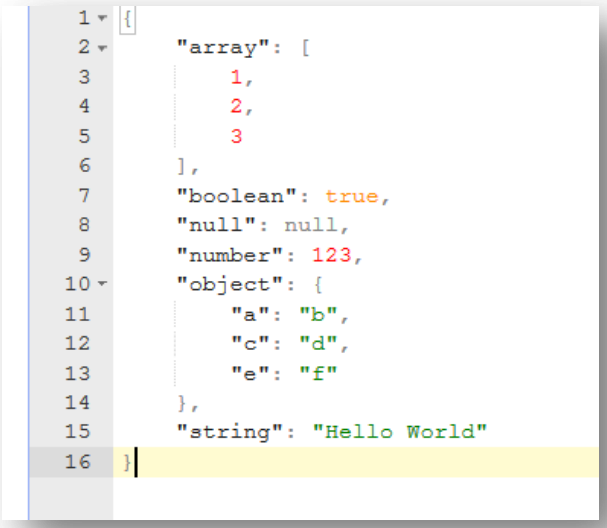
JSON se usa normalmente en aplicaciones donde se requiere de un gran tamaño de datos tanto de envió como de recepción, es un lenguaje ligero y fácil de utilizar para los humanos, y además fácil de analizar por los equipos.

Es independiente del lenguaje que se esté utilizando.

Se basa en dos estructuras:

- Colección de pares nombre/valor.

- Lista ordenadas de valores.



```
{
    "array": [
        1,
        2,
        3
    ],
    "boolean": true,
    "null": null,
    "number": 123,
    "object": {
        "a": "b",
        "c": "d",
        "e": "f"
    },
    "string": "Hello World"
}
```

FIGURA 8: EJEMPLO *JSON*.

Es habitual ver en aplicaciones la utilización de ambos formatos a pesar de estar ambos enfrentados en cuanto a su uso.

3.4. Servidor de datos

En este apartado vamos a hablar del servidor y su composición.

Para la realización del servidor hemos optado por el uso de una máquina virtual en Unix/Linux que nos proporcionará las respuestas a nuestras peticiones.

Hemos hecho uso de **Tomcat y MySQL,** como principales programas para el servicio, además hemos utilizado aplicaciones como **Virtual Box**, para operar nuestra máquina virtual, y **Navicat Lite** para MySQL, cabe destacar que Navicat Lite.

3.4.1. TOMCAT

Tomcat es un servidor de servlets que puede ejecutar servicios Web a través de Apache Axis Es OpenSource gestionado por la asociación Apache. (Figura 8)

Tomcat puede funcionar como servidor por sí solo, esto suele ocurrir en entornos con alto nivel de tráfico y alta disponibilidad.

Tomcat está escrito en Java, por lo tanto funciona en cualquier entorno que disponga de la máquina virtual de Java.

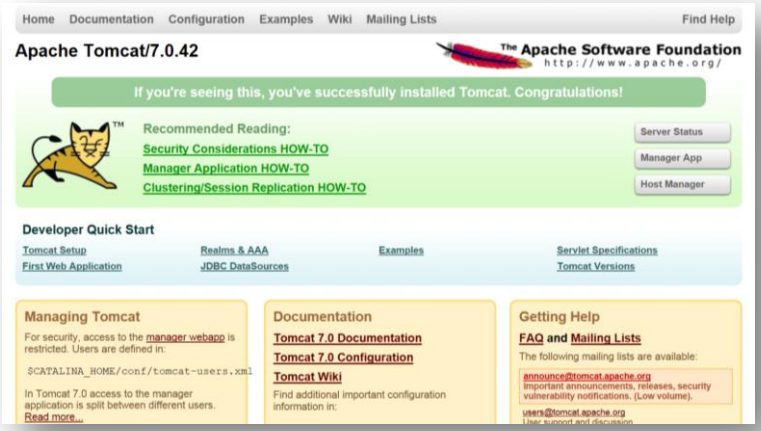


FIGURA 9: ENTORNO APACHE TOMCAT.

3.4.2. MYSQL

MySQL[18] es un sistema de gestión de base de datos relacional, multihilo y multiusuario.

A diferencia de Apache, que es gestionado por la comunidad, MySQL es patrocinado por una empresa privada.

MySQL es un sistema de administración de base de datos OpenSource que permite aplicaciones fiables, de alto rendimiento y cómodamente ampliables basadas en la Web e integradas. Además, proporciona herramientas de gestión, de soporte técnico para desplegar y gestionar las aplicaciones.

Una base de datos es una colección estructurada de tablas que contienen datos.

MySQL es un sistema relacional. Una base de datos relacional archiva todos los datos en tablas separadas, y no todos en una misma tabla. Esto permite velocidad y flexibilidad. Las tablas están conectadas mediante relaciones definidas que hacen posible combinar los datos de diferentes tablas.

Al igual que con CSS, también encontramos muchos gestores de bases de datos, pero además de ser OpenSource, MySQL es una base de datos de reconocido prestigio fácil de usar y de entender, fiable, con alto rendimiento y de la que podemos obtener mucha documentación ya que tiene una buena evolución y soporte.

3.4.3 Otras herramientas

Además de las mencionadas con anterioridad, para la realización del proyecto en general, hemos hecho uso de **Eclipse, Virtual Box y Navicat** Lite para MySQL.

Vamos a tratar de explicar brevemente cada uno de estas herramientas, así como el uso que le hemos dado para la realización de este proyecto.

2.4.3.1. ECLIPSE

Siguiendo con la filosofía de nuestro proyecto, Eclipse[19] es una herramienta OpenSource. (Figura 10)

Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados. Hay plugins para el desarrollo de Java, C/C++, PHP, etc.

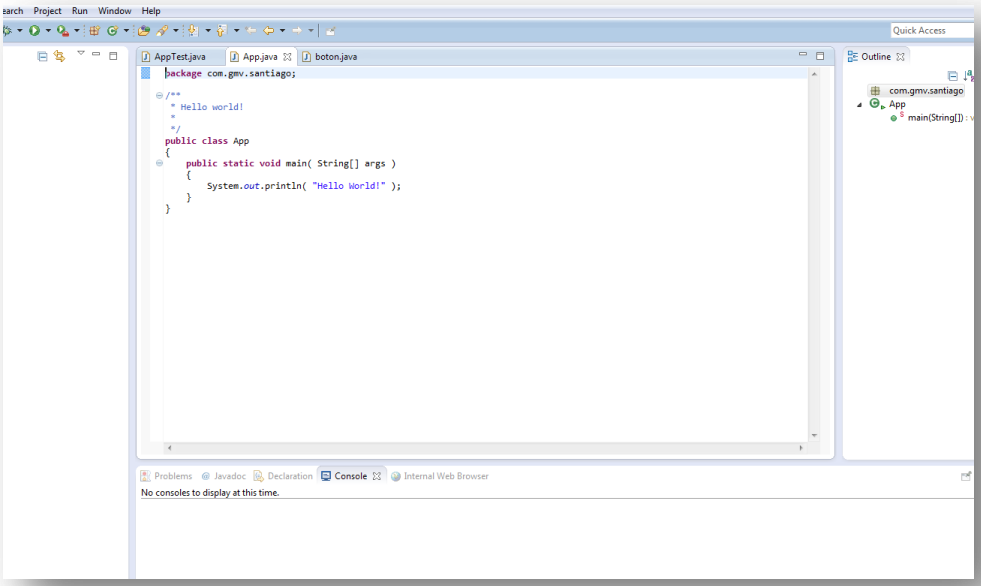


FIGURA 10: ENTORNO ECLIPSE.

2.4.3.2. VIRTUAL BOX

Oracle VM Virtual Box es un software de virtualización para arquitecturas creado inicialmente por la empresa alemana "innotek GmbH".

Actualmente es desarrollado por Oracle Corporation como parte de sus productos de virtualización.

Por medio de esta herramienta somos capaces de instalar sistemas operativos llamado "sistemas invitados", dentro de otro sistema operativo llamado "anfitrión" y que cada uno tenga su propio ambiente virtual.

En este proyecto final se ha utilizado Virtual Box para instalar una máquina virtual con un sistema operativo huésped Ubuntu donde se ha instalado el servidor de la aplicación.

2.4.3.3. NAVICAT LITE

Navicat es un gestor de base de datos. En este proyecto se ha utilizado la versión Lite. (Figura 11)

Navicat nos proporciona un entorno muy sencillo de utilizar y bastante intuitivo a la hora de crear una nueva conexión, nueva base de datos, etc.

Admite varias conexiones a la vez tanto locales como remotas.

Otro conocido gestor de base de datos es Workbench.

A pesar de que Workbench nos da un abanico de posibilidades mayor que Navicat Lite, ya que en esta versión no tenemos todas las funcionalidades propias de Navicat, solo la más básica, este fue el elegido ya que sus funciones eran suficientes para la realización de nuestra base de datos y nuestro servidor.

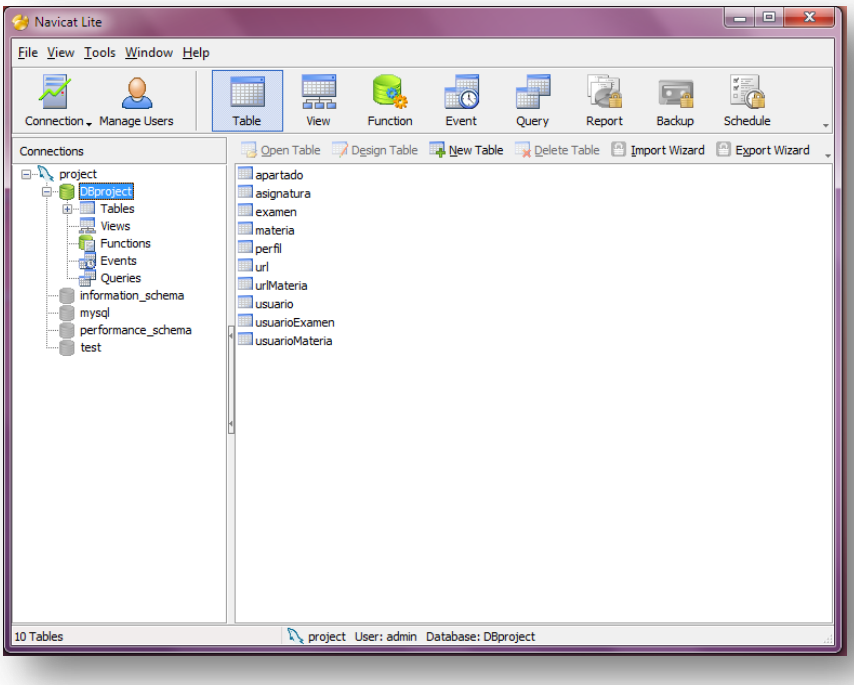


FIGURA 11: ENTORNO NAVICAT LITE

Capítulo 4.

Diseño e implementación de la aplicación

En este capítulo se presenta el diseño de la aplicación así como de la implementación llevada a cabo para lograrlo.

4.1. Cliente

El objetivo de este proyecto final de carrera es el diseño de una aplicación que sirva de apoyo a la comunidad universitaria para el seguimiento de asignaturas de forma fácil y sencilla.

Para ello ha sido necesario el diseño de una aplicación Web que muestre de forma gráfica con la mayor sencillez posible los datos necesarios para los alumnos y profesores para el fin propuesto.

Para el diseño del cliente ha sido necesario definir los diferentes archivos html de los que se iba a componer nuestra Web. (Figura 12)

Una vez realizado el diseño, se realiza la implementación de las mismas utilizando para ello las herramientas y tecnologías mencionadas con anterioridad. (Figura 12)

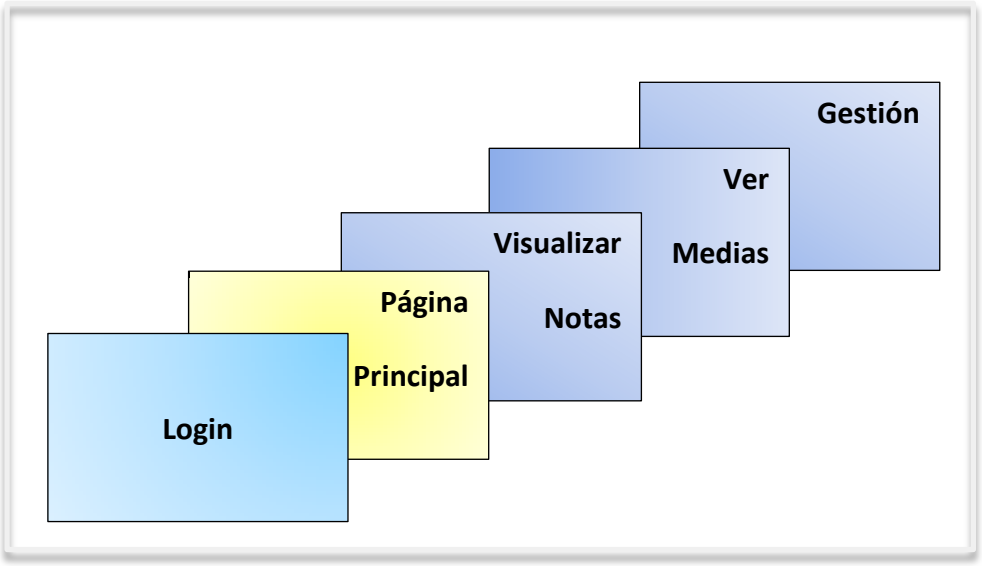


FIGURA 12: DISEÑO DE LAS PANTALLAS WEB

Para conseguir nuestro objetivo se han implementado un total de 5 archivos HTML.

- Login.html
- Home.html
- Media.html
- Notas.html
- Gestión.html

Cuando el usuario que accede a la página es un alumno, en el diseño de la página se ha tenido en cuenta que no pueda ver las calificaciones de sus compañeros, aunque si las estadísticas de la clase.

Por otro lado, si el usuario es profesor, además de poder ver las calificaciones de todos sus alumnos, podrá gestionar la base de datos, añadiendo exámenes, modificando notas de los alumnos etc.

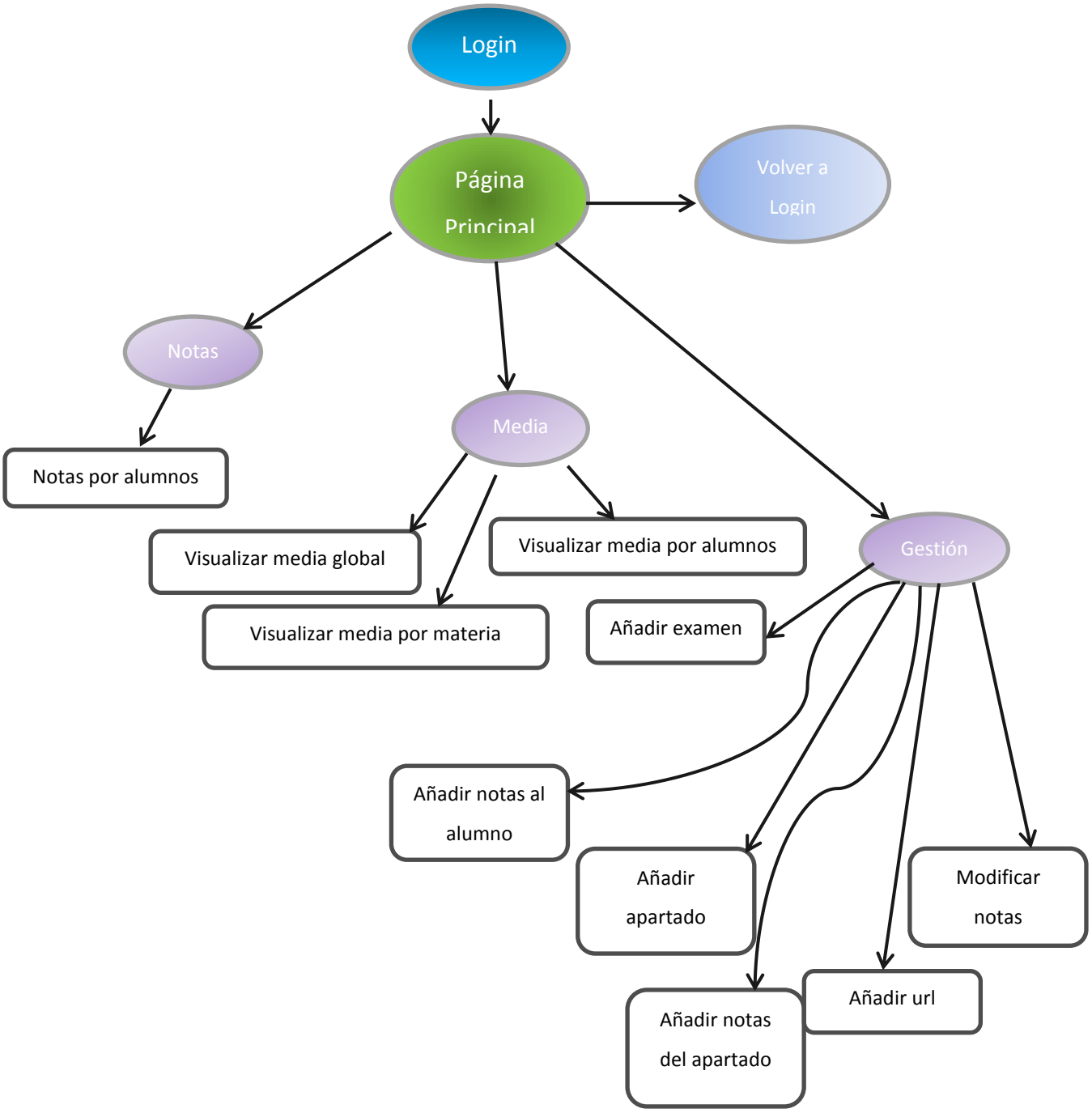
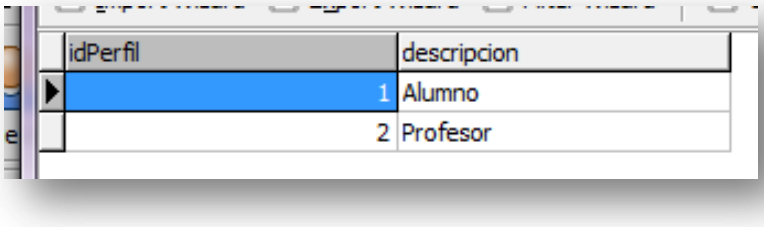


FIGURA 13: DIAGRAMA DE ESTADOS DE LA APLICACIÓN WEB

1. Autenticación del usuario

La autenticación se ha llevado a cabo mediante un registro de usuarios y sus perfiles en la base de datos de MySQL desde la página *"login.html"*. (Figura 14)



idPerfil	descripcion
1	Alumno
2	Profesor

FIGURA 14: TABAL PERFILES BASE DE DATOS MYSQL

Se ha realizado la implementación de la página mediante tres "imputs". Uno de ellos será de tipo texto, nombre; el otro será de tipo password, pwd. El último será un botón que llamara a un servlet.

Una vez introducidos los datos, que serán obligatorios, se hace una llamada al servlet LoginServlet.java que accede a la base de datos y comprueba que el usuario que se ha introducido está en la base de datos, así como que su contraseña es la correcta.(Figura 15)

```
        //Parámetros que hemos introducido en el formulario
        name = req.getParameter("nombre");
        pwd = req.getParameter("pwd");

        //Comprobamos si el usuario que hemos introducido está en la base de
datos
        //Método usuarioCorrecto()
        //Comprueba que el usuario es correcto
        //@param name: user introducido por el usuario
        //@param pwd: contraseña introducida por el usuario
        //@retun int: indica si la acción ha sido correcta, incorrecta, y si
        es correcta que usuario es alumno o profesor.
        int resp =usuarioCorrecto(name,pwd);
```

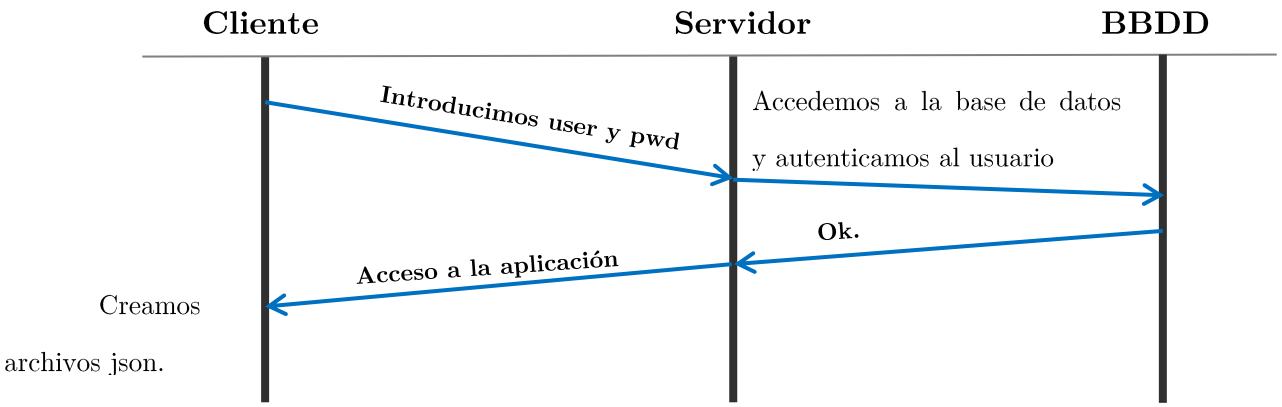


FIGURA 15: AUTENTICACIÓN ENTRE CLIENTE SERVIDOR

Si todos los campos son correctos se procede a la creación de dos archivos *JSON* (Figura 16):

- Datos_servidor.*json*
- Datos_gestion.*json*

El primero de los archivos, datos_servidor.*json*, es un archivo que nos da la información de toda la base de datos.

El primer valor que aparece es el "tipo". Será o el nombre del alumno o el string "profesor" e indicará cuando se ha conectado cada uno para que, si es un alumno, solo pueda ver su propia información.

En segundo lugar lo que aparece en el archivo es un menú, el menú se diseña de acuerdo al perfil del usuario, así podemos ver que si es un perfil alumno el menú tiene una opción menos, la de gestión, mientras que si es un profesor el menú tendrá todas las opciones posibles.

```
"menú": [
    {
        "pagina": "./home.html",
        "titulo": "Portada"
    },
    {
        "pagina": "./notas.html",
        "titulo": "Notas"
    },
    {
        "pagina": "./media.html",
        "titulo": "Media"
    }
],
```

En tercer y último lugar lo que aparece es un campo “list”. Aquí comienza lo que son los datos de la base de datos.

Para la realización del archivo *JSON* lo primero que deberíamos tener en cuenta es la asignatura para la cual se va a realizar. Este proyecto está diseñado para una sola asignatura por lo que el “idAsignatura” a buscar siempre será “1”. En el caso en el que hubiera más asignaturas habría que hacer una búsqueda a la base de datos y devolver un archivo por cada asignatura con la información correspondiente a cada una de ellas.

Una vez que sabemos la o las asignaturas para las que hay que crear los archivos *JSON,* en nuestro caso una, buscamos para cada alumno en la asignatura sus exámenes, formados por apartados, notas y materias.

Cada examen consta de una nota global, que será la que se use para hacer la media de alumnos en la asignatura, y para que el alumno pueda ver cómo va en ella.

Además, cada apartado tendrá una nota de forma independiente, es decir, cada apartado tendrá una nota correspondiente a su materia.

Puede darse el caso de que un mismo apartado trate de dos materias diferentes de la asignatura, en este caso, la nota se pondrá por materia, no por apartado global, tendremos registrada la materia que a tratar y además de unas *urls* correspondientes para tantas materias como tenga el apartado.

Como ya dijimos, la herramienta tiene como objetivo ayudar a los alumnos en la asignatura con ejercicios y apuntes, por lo que se ha sido por lo que las mencionadas *urls* llevaran al alumno a página de ayuda.

Para el diseño y el formato de la página de *login.html* nos hemos basado en CSS, explicado con anterioridad.

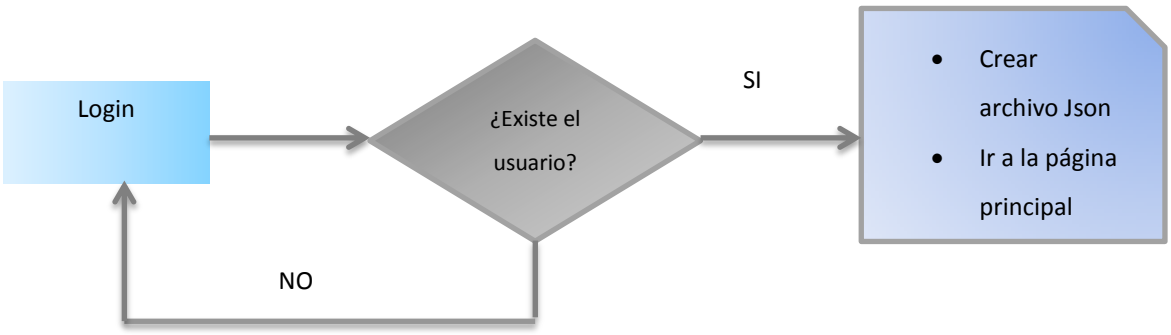


FIGURA 16: DIAGRAMA DE FLUJO DE LA PÁGINA PRINCIPAL

2. Página principal

Esta es la página "*home.html*" a la que accedemos una vez autenticados.

En ella tendremos un botón que nos devolverá a la página de autenticación y enlaces a página de interés, como es la página de la Universidad Carlos III de Madrid, o la página de Open Course Ware de la universidad Carlos III también.

Además se muestra un menú para navegar por las distintas opciones de la herramienta y un breve resumen del contenido de la misma.

3. Página de visualización de notas

Para dotar de inteligencia a esta página hemos creado un archivo JavaScript con las funciones necesarias, *tools.js*.

- La función *alumnos()* nos proporciona la lista de alumnos a seleccionar, esta lista de alumnos se realiza a partir de lecturas del archivo *JSON*.
- Cuando se selecciona al alumno que queremos este llama a la función notas_*alumno()*, a la que se le pasa por parámetro el nombre del alumno seleccionado.

```
notas_alumno(nombre_alumno)
```

- Con este nombre se buscan las notas pertenecientes al alumno que tiene ese nombre y con ellas se crea un "string" que será pasado como parámetro a la siguiente función *pop_up()*.
- Para cada alumno seleccionado aparecerá un gráfica en la que podremos ver una línea azul que nos indica en qué punto está el aprobado, normalmente suele ser en el cinco. En blanco aparecerá otra línea que indicará la nota obtenida por el alumno en cada uno de los parciales que se hayan realizado a lo largo de la asignatura.

Como ya hemos dicho las notas son por materia, y no por apartado, ya que así se ha decidido el diseño para llevar a cabo el cálculo de estadísticas y medias que explicaremos más adelante.

Los nombres de las materias serán enlaces. Estos enlaces están relacionados con las *urls* que se encuentran en la base de datos así, si seleccionamos cualquiera de las materias, iremos a una página que estará

relacionada con el temario y que su fin es ayudar al alumno con ejercicios, explicaciones, etc.

Para realizar la gráfica nos hemos basado en la librería RGraph[20][21], en la función *grafica_alumno()*. (Figura 17)

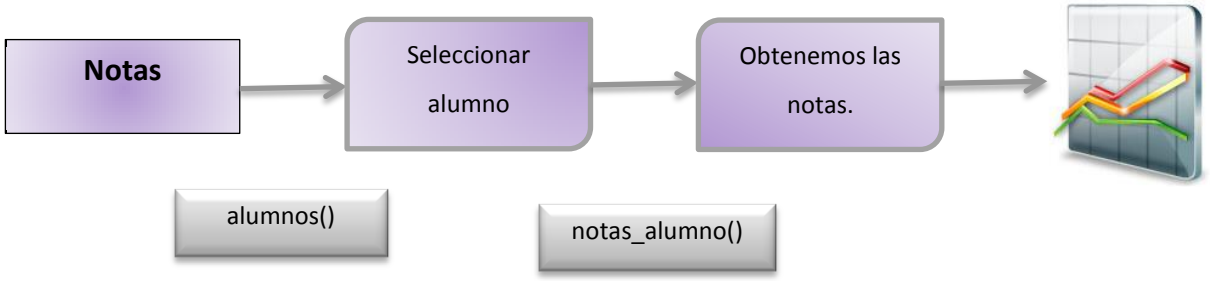


FIGURA 17: DIAGRAMA DE FLUJO DE LA VISUALIZACIÓN DE NOTAS

4. Página de visualización de medias.

En esta página veremos las estadísticas de la asignatura.

- La primera pestaña nos muestra la media total de la clase.
- En verde aparecerá la media de la clase total y en negro veremos la media de cinco.

Para ello utilizamos las funciones que nos ofrece el archivo de JavaScript *tools_media.js*.

- Para el cálculo de la media se han ponderado todos los exámenes por igual, si se decidiera otra forma de calcular la media por la asignatura debería integrarse en el código el nuevo algoritmo. En esta pantalla también podemos ver la media por materias. Puede

ser interesante saber en qué temario de una asignatura los alumnos necesitan más ayuda.

- A continuación se forma el desplegable con todas las materias que aparecen en los diferentes exámenes y apartados.
- Cuando seleccionemos una de las materias se llamará a la función *grafica_materias()*, a la cual se le pasa por parámetro la materia seleccionada.

```
grafica_materia(materia_seleccionada)
```

- Procedemos a buscar las notas para esa materia para posteriormente calcular la media y obtener la gráfica con los datos.

En la última pestaña se hace una media de alumnos, es decir, en esa pestaña se podrá ver una gráfica con la media de todos los alumnos para una materia.

En esta pestaña además, se ha añadido un gráfico de porcentajes que representa en total que porcentaje de alumnos están aprobados o suspensos en la materia en cuestión.

- En esta página se ha realizado un desplegable de materias que cuando seleccionamos se llama a la función *buscar_materia2()*, que recibe como parámetro la materia seleccionada.

```
buscar_materia2(materia)
```

- A continuación se busca esa materia para cada alumno y sus notas y se hace una media de ellas. Con estos datos utilizaremos la función *grafica_media_alumnos()* para representar los datos.(Figura 18)

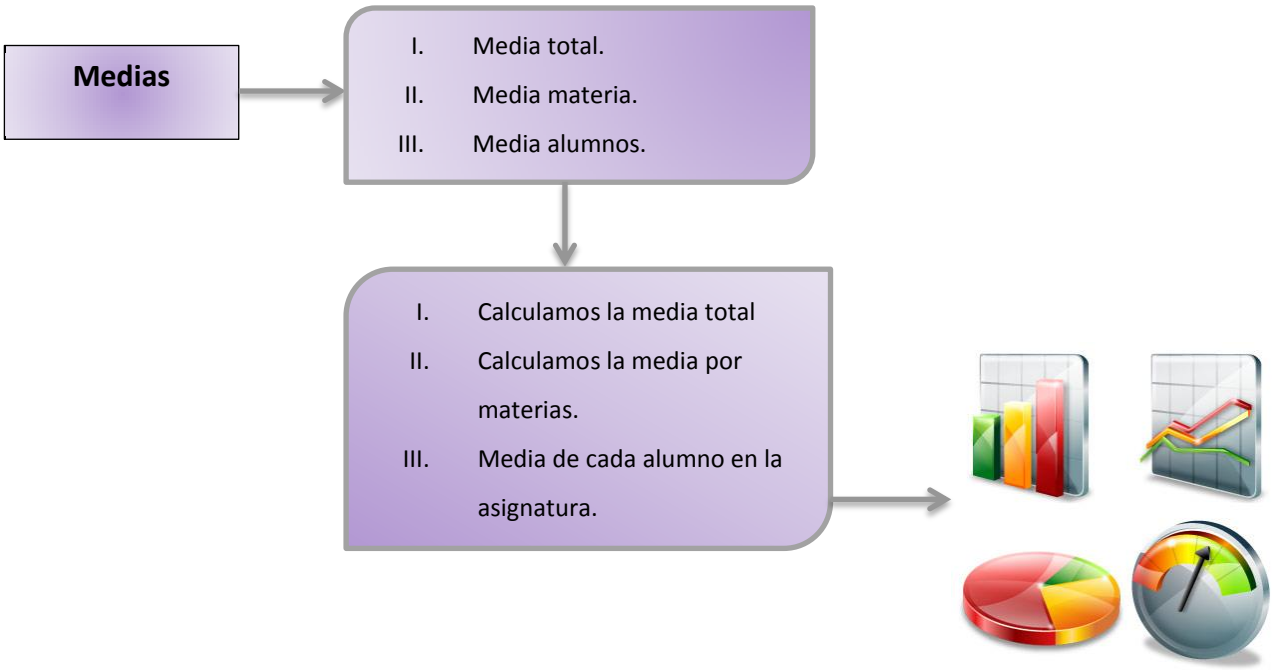


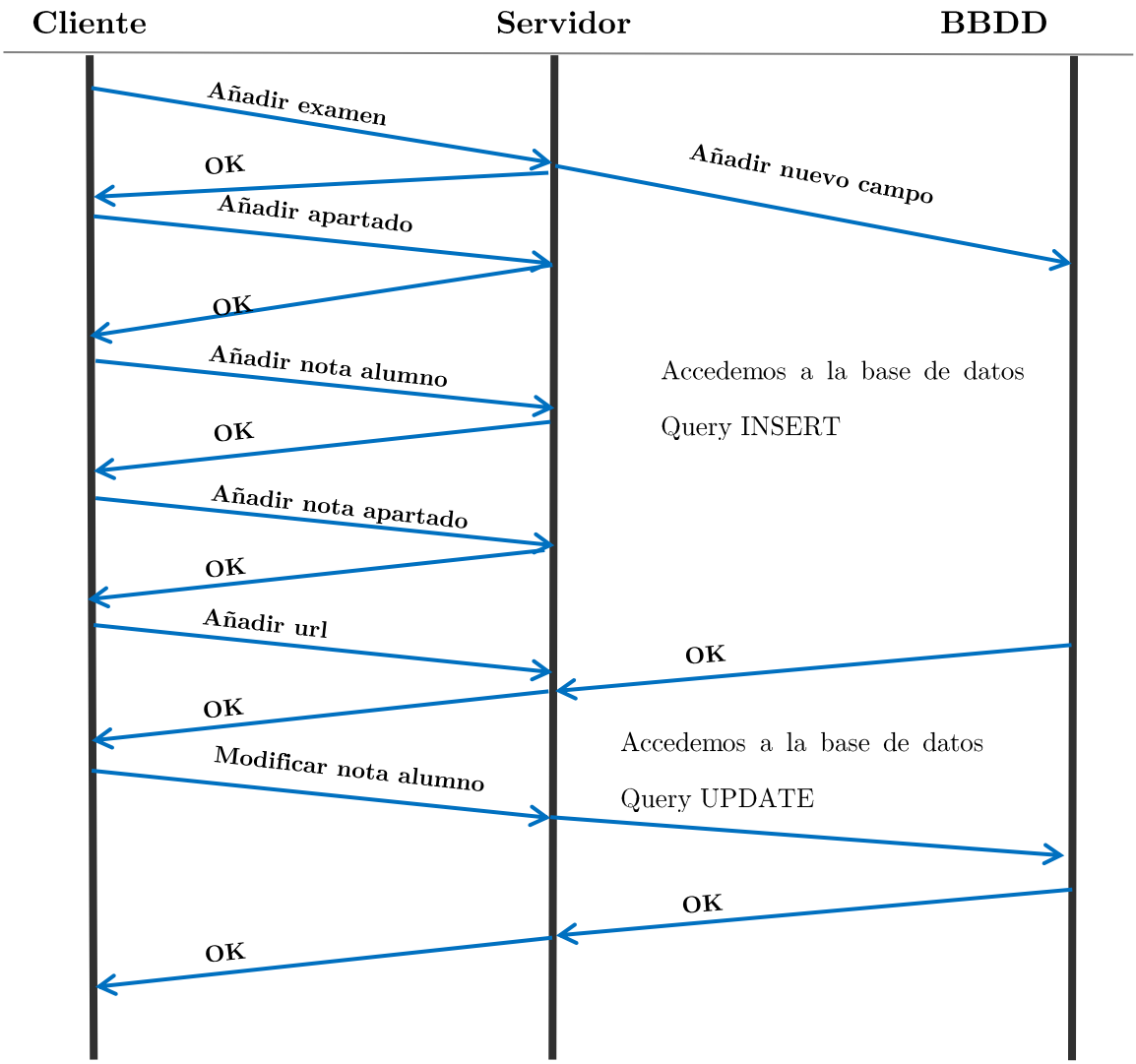
FIGURA 18: DIAGRAMA DE FLUJO DE LA PÁGINA DE MEDIAS

5. Página de gestión.

- Cuando accedemos a esta pestaña llamamos al servlet GestionServlet.java. Este servlet se encarga de hacer los accesos a la base de datos necesarios para añadir y/o modificar información de ella. (Figura 15)
- Cada vez que modifiquemos un valor de la base de datos llamaremos a la función *jsonGestion(),* que nos actualizará el *JSON* datos_gestion.*json* creado en la página *login.html*.

Tenemos varias pestañas, es recomendable, o más bien obligatorio seguir el orden de las pestañas a la hora de introducir los datos en la base de datos.

Para esta página se usa el archivo de JavaScript *tools_gestion.js*, en este archivo se encuentran las funciones necesarias para poder obtener las listas desplegables que aparecen en la web.



4.2. Servidor

El servidor consta de dos clases fundamentales, de dos clases Java – Servlets (Figura 20 y 21)

Ambas clases son llamadas desde el cliente mediante el protocolo HTTP, y responden con el mismo protocolo al cliente teniendo el siguiente ciclo:

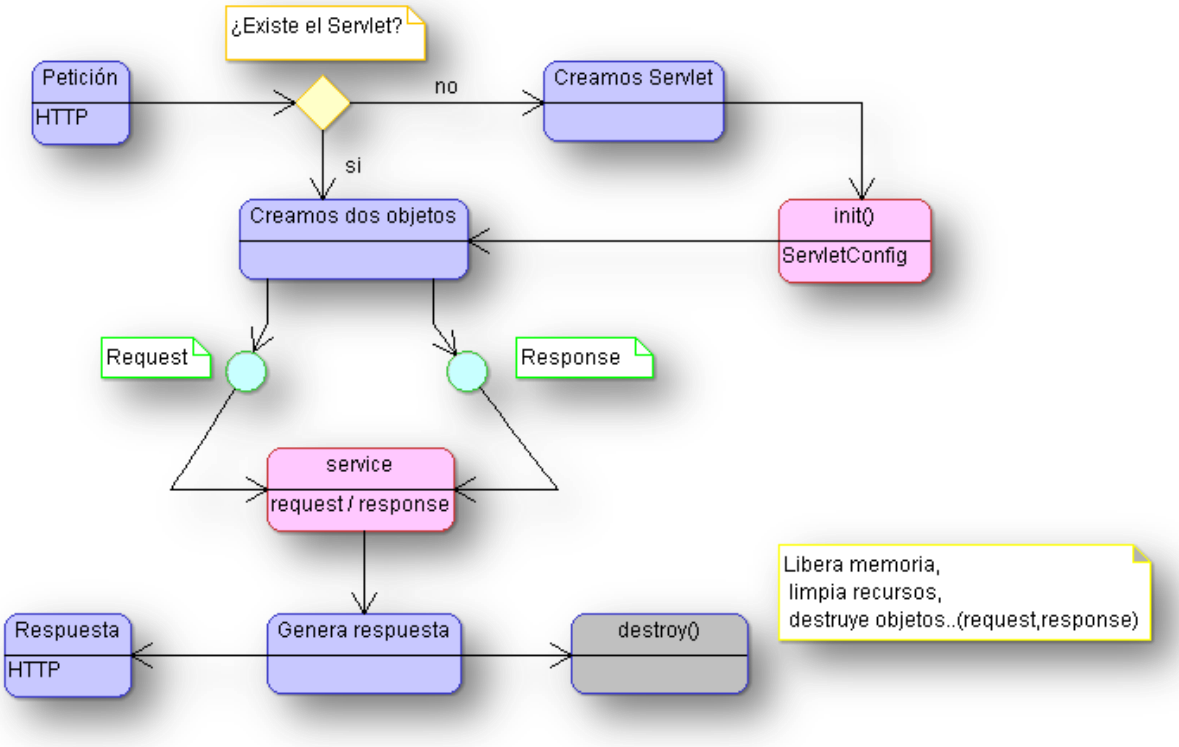


FIGURA 20: CICLO DE FUNCIONAMIENTO DEL SERVLET.

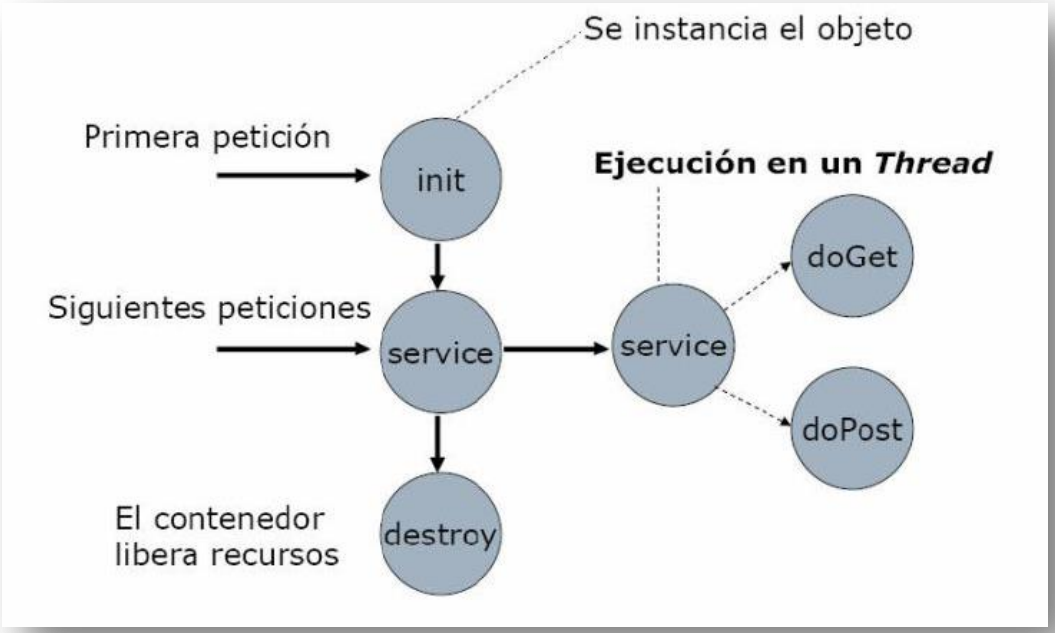


FIGURA 21: CICLO DE VIDA DEL SERVLET

Sus principales métodos:

- **init()**

  Se ejecuta una sola vez cuando arranca el servlet, y es el método donde se suelen inicializar variables o hacer operaciones costosas en tiempo de ejecución.

- **doGet() o doPost()**

  Recoge las peticiones del usuario, y ejecuta las operaciones pertinentes.

- **destroy()**

  Este método es llamado por el servidor al apagarse y nos permite dejar el sistema en buen estado ya que cierra todos los procesos en curso, librera memoria y cierra ficheros.

  ❖ LoginServlet

  ❖ GestionServlet

La clase *LoginServlet.java* es llamada desde la página inicial login.html y será la que nos de la autenticación del usuario. Se ha realizado la conexión con el método *doGet(),* mientras que en *GestionServlet.java*, clase que nos ayuda a la gestión de la base de datos se ha utilizado el método *doPost().*

Ambos métodos nos ayudan a obtener los valores de las variables que hay en nuestra página, por ejemplo un formulario, un acceso etc.

La principal diferencia entre ambos métodos es la forma de enviar los parámetros de entrada.

Mientras que en el método GET los parámetros van explícitos junto a la página, es decir, los manda junto con la *url* (uniform resource locator), en el

método POST van implícitos, nosotros no vemos en ningún momento los parámetros que se han introducido y solo son accesibles desde los formularios.

Por ello GET permite que se llame al servlet pasándole los parámetros de entrada desde la *url* mientras que POST solo llamará al servlet desde los parámetros introducidos en el formulario.

Una vez explicado las funciones del cliente, vamos a pasar a explicar cómo se ha implementado toda la parte del servidor.

También explicaremos el diseño que se ha hecho de la base de datos.

En primer lugar nuestro servidor corre en una máquina virtual de Ubuntu 12.04 en la cual está instalado el Apache Tomcat y MySQL.

4.2.2 Servlet

Una vez llegados a este punto, y teniendo el conector mysql-java en nuestro directorio accedemos a la base de datos desde el servlet.

En la siguientes líneas de código se muestran los parámetros de nuestra base de datos, la dirección IP fue elegida por nuestro rango, si se desea otra hay que modificar la IP como hemos visto anteriormente para ponerla fija y acordarse de modificarla también en ambos servlets. A continuación cargamos el driver que hemos descargado, y realizamos la conexión con la base de datos. Si todo ha ido bien ya tendremos acceso a la base de datos desde el servlet.

```
//Parámetros de la base de datos
static String bd="DBproject";
static String user="admin";
static String password="manager";
static String server="jdbc:mysql://192.168.1.22/"+bd;
```

```
        //Cargamos el driver
        try {
        Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e1) {}

try{
        conn = DriverManager.getConnection(server,user,password);
        System.out.println("Conexión con la BD realizada");
        }
```

4.3. Base de datos.

La base de datos se ha diseñado con MySQL.

Se han diseñado un total de 10 tablas en MySQL con la información necesaria para la gestión y uso de la página web.

Las tablas recogen desde nombres de usuarios hasta direcciones urls para hacer la semántica de la página.

Hemos diseñado una base de datos con estructura EER como podemos ver en la siguiente imagen. (Figura 22)

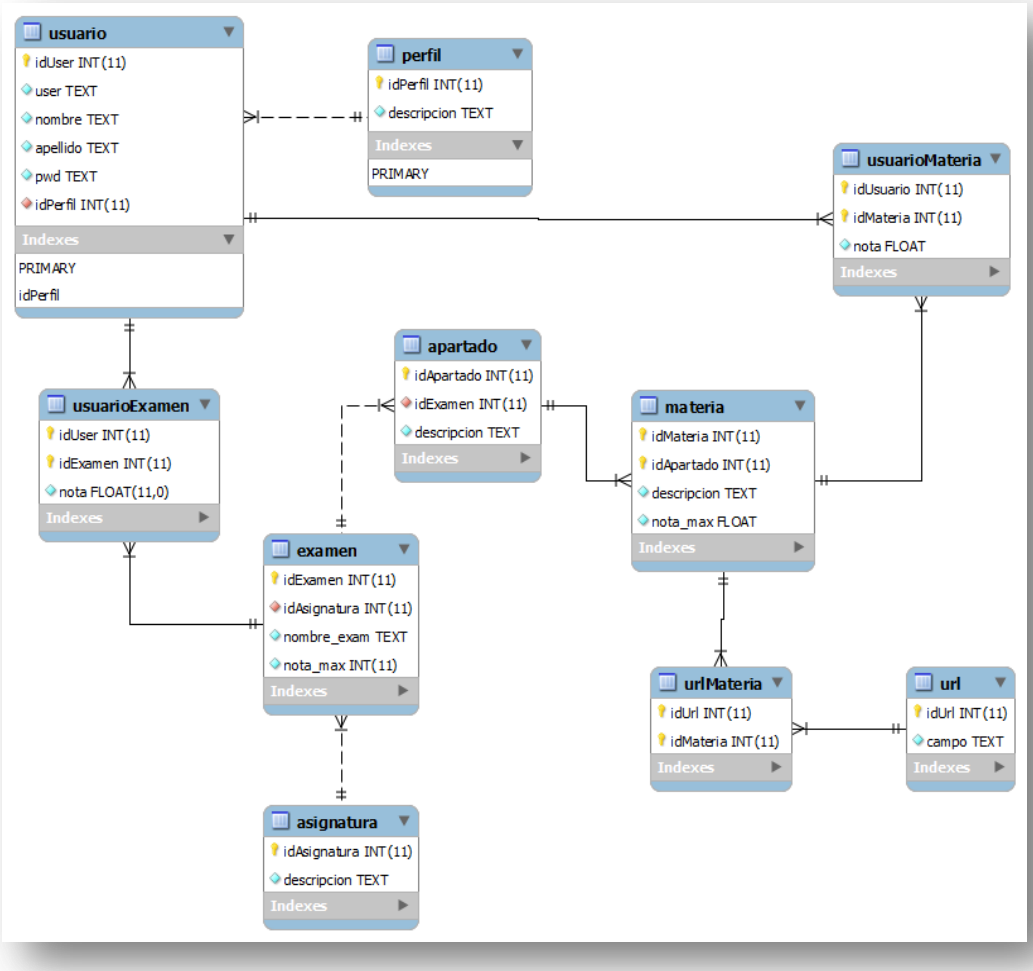


FIGURA 22: ESQUEMA BASE DE DATOS

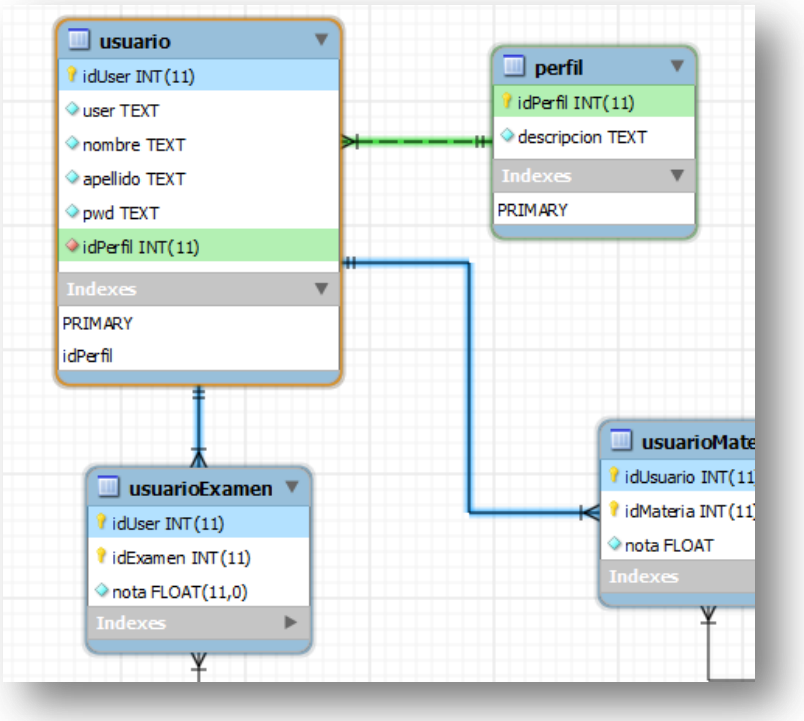


FIGURA 23: RELACIONES USUARIOS BASE DE DATOS

En la imagen anterior podemos ver la relación con de la tabla 'usuarios' (Figura 23).

En esta tabla, además del nombre, el 'pwd' y el 'username' del usuario en cuestión, también podemos ver el perfil del usuario, que estará relacionado con la tabla de perfiles, donde aparece el idPerfil y una descripción que será 'alumno' o '`profesor'.

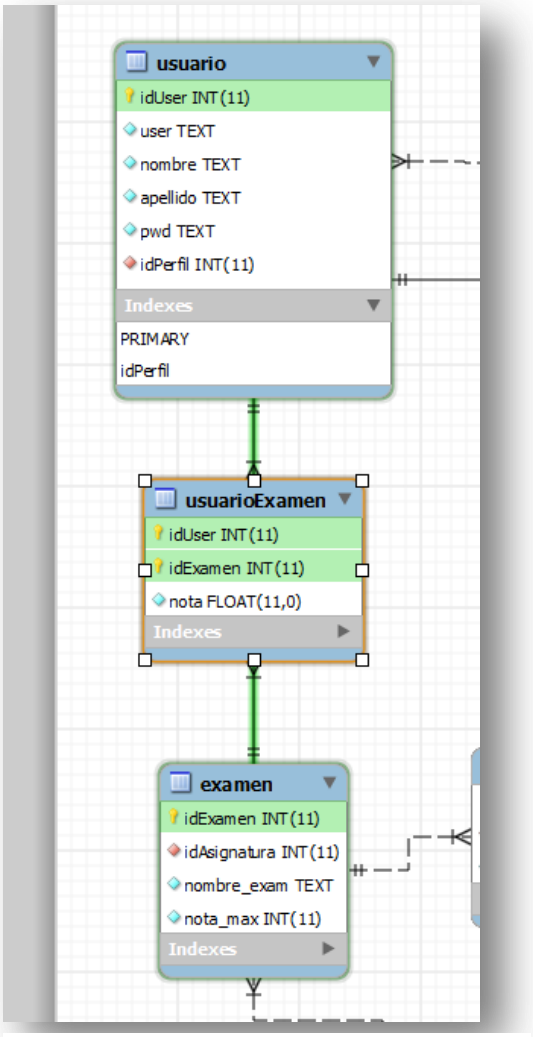


FIGURA 24: RELACIONES USUARIO - EXAMEN

A continuación vemos las relaciones de la tabla 'usuarioExamen'. (Figura 24)

Hemos seguido el diseño pensando en el examen propuesto por un profesor.

La tabla examen, estará relacionada a una asignatura, su nombre y una nota máxima, será como el diseño de un examen.

En nuestra tabla 'usuarioExamen 'tendremos la relación entre los alumnos y el examen propuesto por el profesor, así esta tabla nos relaciona los idExamen con los IdUser y añade la nota del alumno para ese examen.

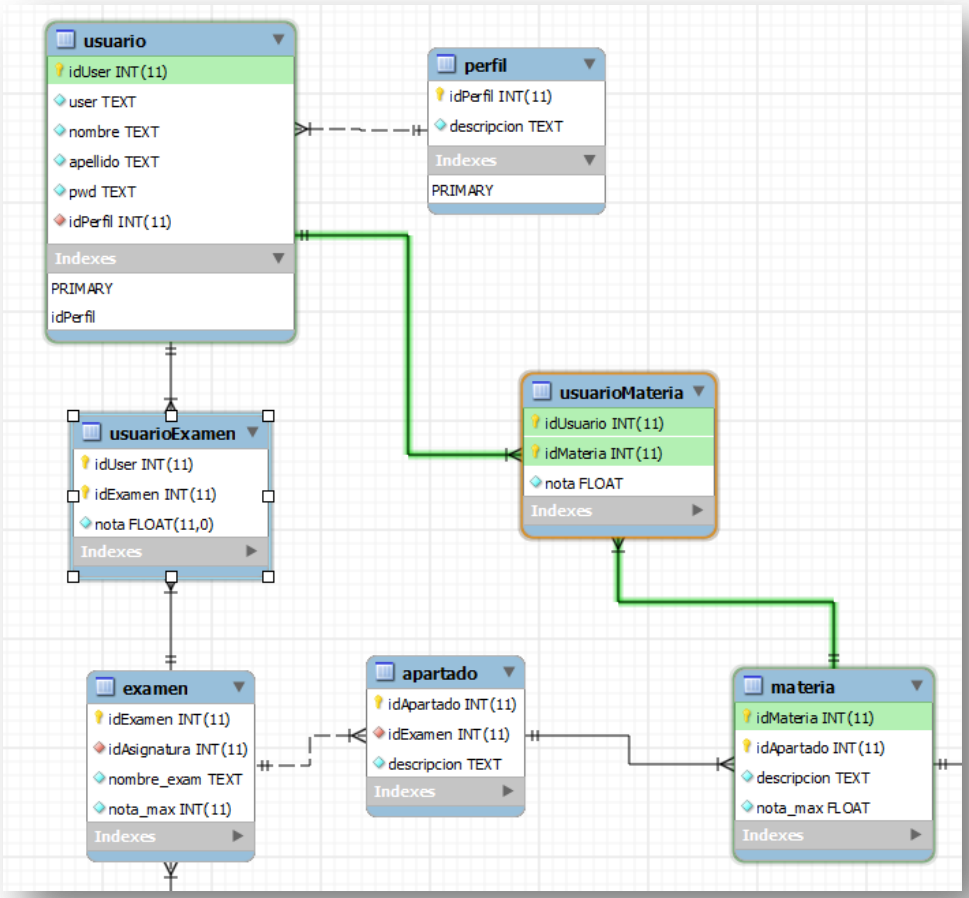


FIGURA 25: RELACIONES USUARIO – MATERIA

Al igual que en el caso anterior, tenemos una tabla 'usuarioMateria' (Figura 25) que nos relaciona al usuario con una materia concreta.

Esto se ha diseñado así, porque recordemos las notas de los apartados están asociadas a la materia de la que se trata ese apartado, y no al apartado en cuestión.

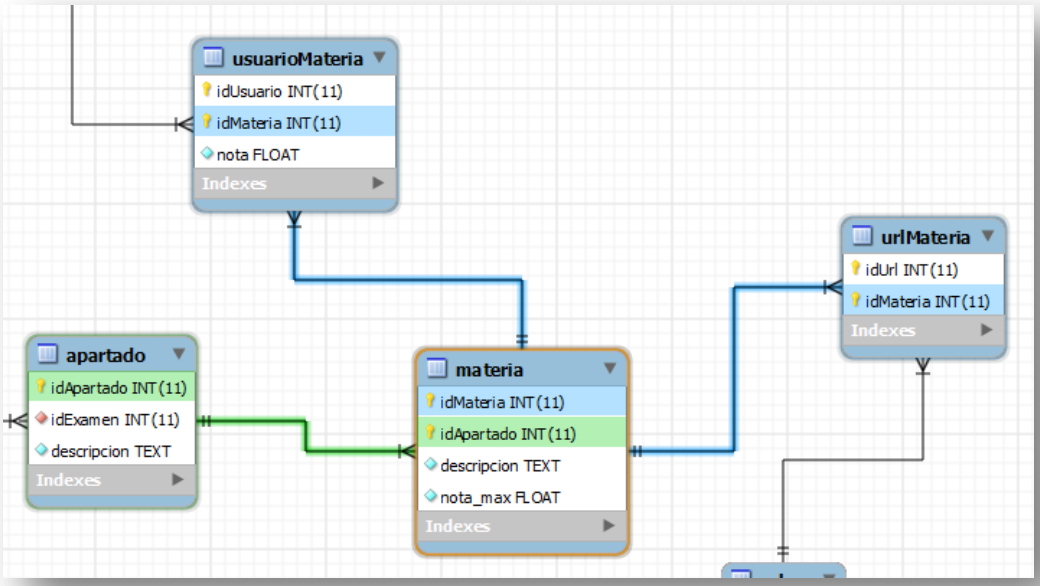


FIGURA 26: RELACIÓN DE LA TABLA MATERIA

En la imagen anterior vemos la relación de la tabla 'materia'.(Figura 26)

Se relaciona con la tabla 'usuarioMateria' explicada anteriormente, con la tabla 'urlMateria' y con los apartados.

De esta forma un materia corresponde a un apartado concreto, que a su vez corresponde a un examen, tiene un alumno asignado, es decir, un alumno tiene esa materia, y además está asociada a una *url* que se une por medio de tabla maestra 'urlMateria'.

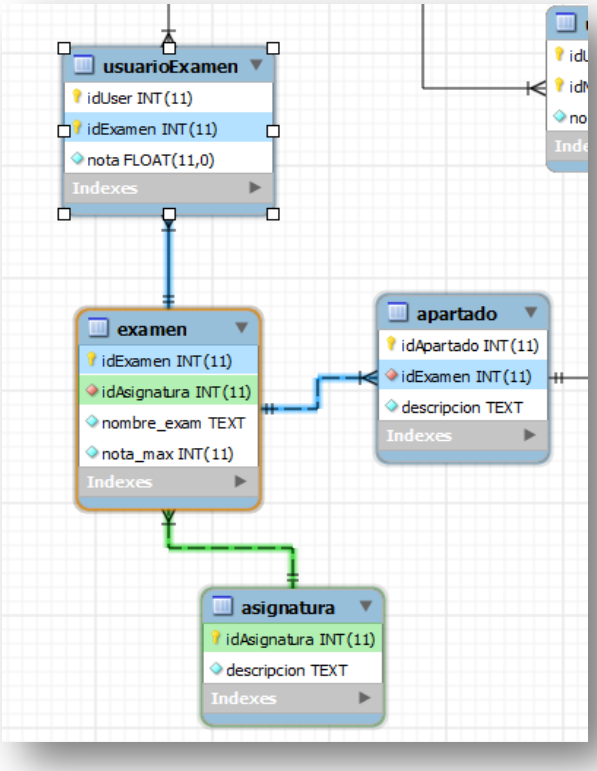


FIGURA 27: RELACIONES EXÁMENES APARTADOS

Aquí podemos ver la relación de los apartados en los exámenes. (Figura 27). Un apartado siempre irá asociado a un examen, al igual que un examen siempre irá asociado a una asignatura.

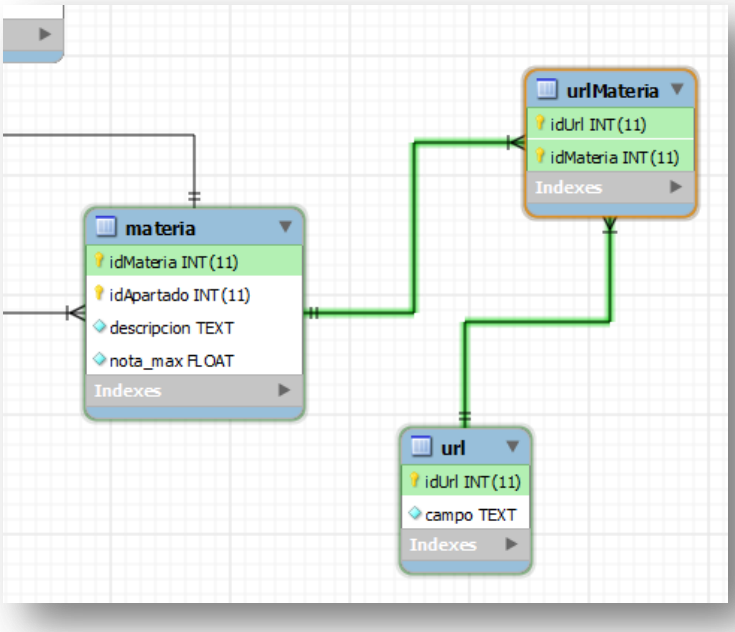


FIGURA 28: RELACIÓN MATERIA - *URL*

Por último tenemos las tablas que relacionan las materias con las *urls*. (Figura 28)

Simplemente tenemos una tabla maestra que nos relaciona el idUrl con el idMateria. El diseño está pensado así para poder asignar a una misma materia varias *urls*.

Capítulo 5.

Validación

En este capítulo vamos a mostrar unas capturas de la aplicación conseguida y los pasos a seguir para la instalación del servidor.

Además explicaremos brevemente los pasos a dar en cada una de ellas para su uso.

5.1. Instalación del servidor

Como ya hemos dicho el servidor se ha instalado en una máquina virtual mediante VM Virtual Box. (Figura 18).

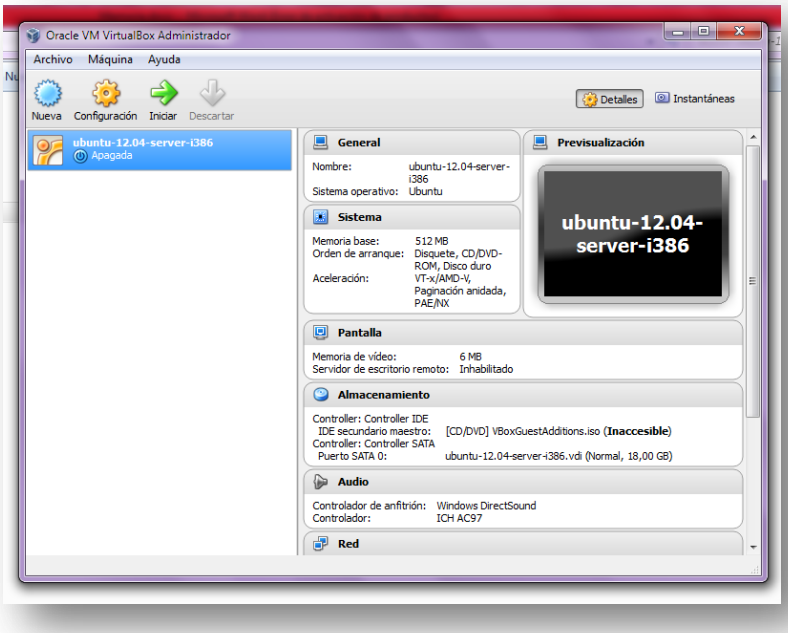


FIGURA 29: ENTORNO VIRTUAL BOX

1. Lo primero que hemos hecho ha sido descargar Tomcat en nuestra máquina virtual, para ellos hemos seguido los siguientes pasos:

   Instalamos Apache Tomcat 6

```
>> sudo apt-get install tomcat 6
```

   Puedes moverlo de ubicación si lo deseas:

```
>> sudo mv apache-tomcat-6.0.37/ /nuevadir/
```

2. Instalamos jdk Java

```
>> sudo apt-get install default-jdk
```

3. Añadimos la información de las variables de entorno.

```
>> export JAVA_HOME=/usr/lib/jvm/default-java

>> export CATALINA_HOME=~/path/to/tomcat
```

4. Por ultimo podemos activar Apache Tomcat

```
>> $CATALINA_HOME/bin/startup.sh
```

5. Desde el navegador que utilicemos pondremos

   Localhost:8080

   Si todo ha ido bien deberíamos ver algo como esto:

FIGURA 30: APACHE TOMCAT

6. A continuación realizamos la instalación de MySQL.

   Instalamos MySQL

```
>> sudo apt-get install mysql-server
```

7. Una vez terminada la instalación podemos ver si ha sido correcta.

```
>> sudo netstat -tap | grep mysql
```

8. Cuando lancemos este comando deberíamos a ver algo así.

```
tcp        0      0 localhost:mysql         *:*                 LISTEN      2556/mysqld
```

Y restauramos MySQL.

```
>> sudo service mysql restart
```

En nuestro desarrollo, se decidió tener un servidor con dirección IP fija, ya que es más fácil trabajar con las clases Java y el acceso a la base de datos.

Para tener la IP fija basta con cambiar las IP's que vamos a utilizar del siguiente modo.

1. Accedemos al archivo /etc/mysql/my.cnf. Podemos usar editores como gedit que se llaman desde la consola de Ubuntu.
2. Una vez en ese archivo encontraremos una línea con

```
bind-address = 127.0.0.1
```

   La comentamos, y añadimos una línea igual pero con la dirección IP que elegimos para nuestro servidor.

3. Ahora también tenemos que cambiar la DNS.

   Accedemos al archivo /etc/network/interfaces. Igual que antes podemos acceder con gedit desde la consola.

   En este archivo veremos La dirección ha cambiar, el Gateway y la máscara.

   Cambiamos las tres cosas, colocando en la dirección la misma elegida antes para colocar en el archivo.

Con esto ya podemos acceder a nuestro servidor.

A continuación accederemos a la base de datos, para ello, desde el servidor es necesario contar con los servlets, que son clases java como ya hemos explicado.

Al iniciar la página web la primera imagen que se nos aparecerá será del archivo de **login.html**

FIGURA 31: IMAGEN DE LA PÁGINA DE LOGIN.HTML

En esta primera página se nos pedirá una autenticación para saber principalmente si el usuario es alumno o profesor.

A continuación veremos la página de inicio, **home.html**

Esta será nuestra página principal. Aquí vemos un menú en el que se aprecian las páginas que hemos comentado anteriormente.

FIGURA 32: PÁGINA DE INICIO (1)

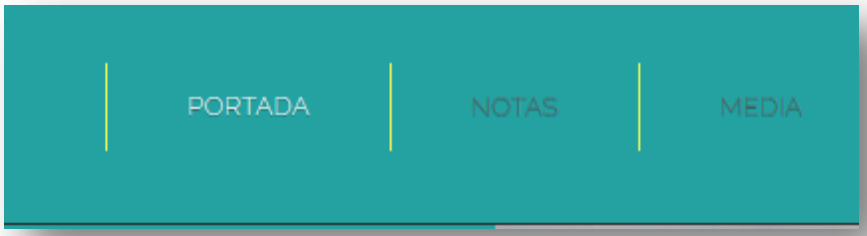


FIGURA 33: MENÚ PRINCIPAL

FIGURA 34: PÁGINA DE INICIO (2)

Al igual que en el caso anterior, el formato de la página es dado por un archivo css, en este caso "default.css".

La siguiente página que vemos es **notas.html.**

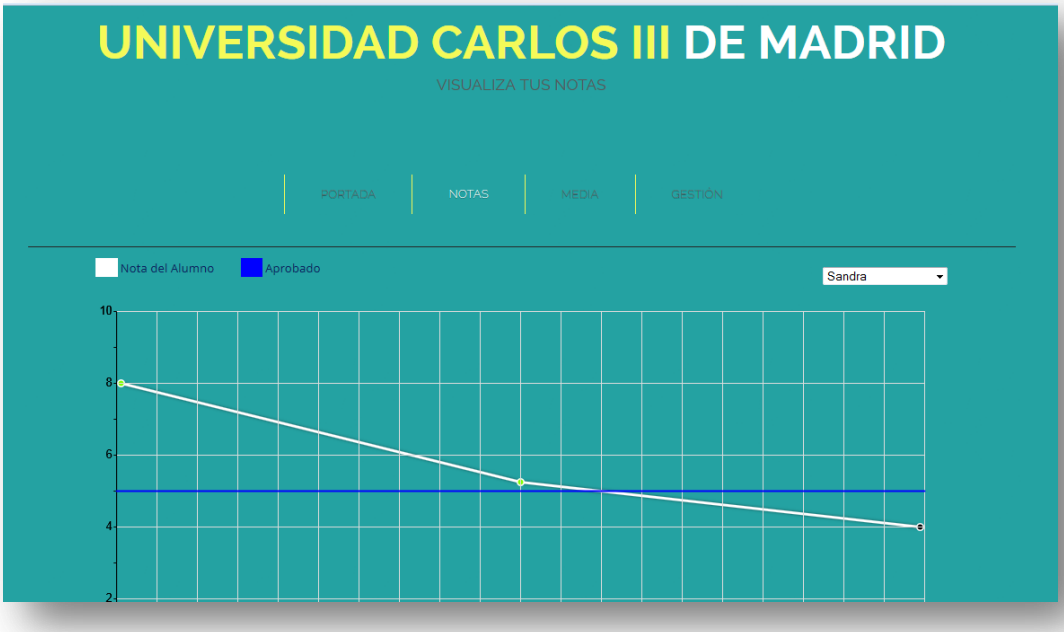


FIGURA 35: PÁGINA NOTAS.HTML

Si pasmos por encima con el cursor del ordenador podemos ver como aparece una ventana pop-up, que nos muestra la información para el examen en cuestión.

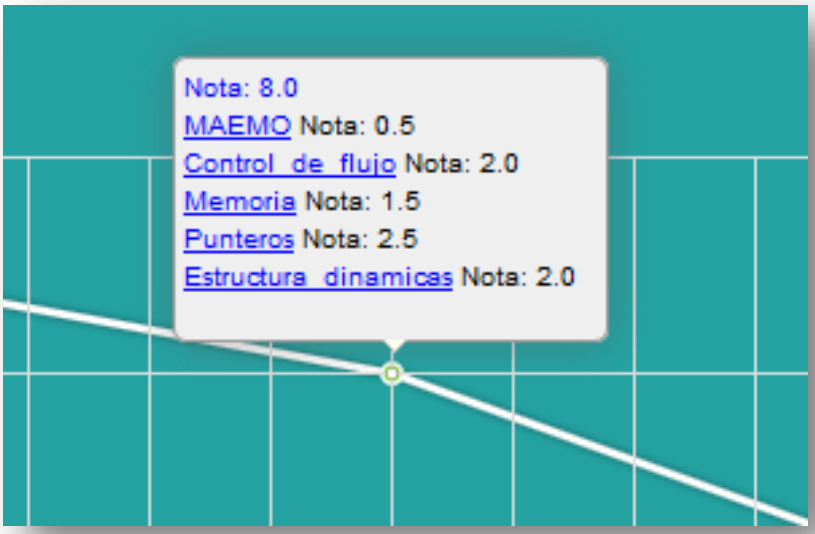


FIGURA 36: POP-UP DE LAS NOTAS

En la siguiente ventana tenemos la página **media.html**

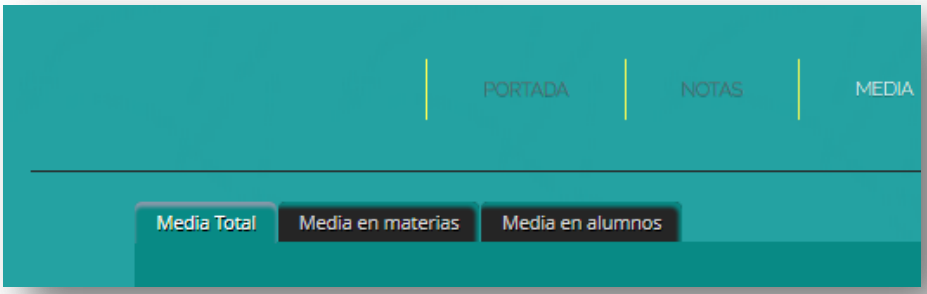


FIGURA 37: MENÚ MEDIA.HTML

FIGURA 38: PÁGINA MEDIA.HTML

En la segunda pestaña podemos ver la media por materias.

FIGURA 39: PÁGINA MEDIA.HTML (2)

Si el usuario es profesor se le permitirá ver en la gráfica los nombres de los alumnos para los cuales son las notas, mientras que si el usuario es el alumno podrá ver cómo van los demás, sin saber quiénes son.

En esta pestaña además, se ha añadido un gráfico de porcentajes que representa en total que porcentaje de alumnos están aprobados o suspensos en la materia en cuestión.

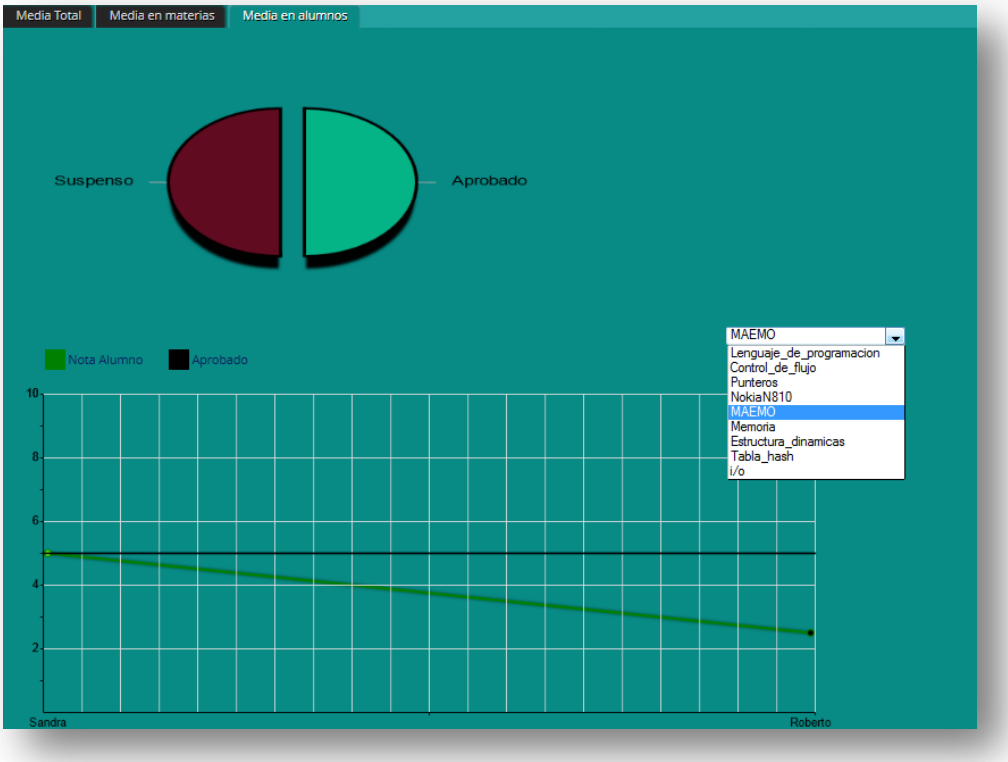


FIGURA 40: PÁGINA MEDIA.HTML (3)

En la última página veremos **gestión.html.**

FIGURA 41: PÁGINA GESTION.HTML

Esta opción solo está habilitada para profesores.

En la primera pestaña se nos permite agregar un nuevo examen. Para ello habrá que seleccionar la asignatura para la que será el examen, aunque en nuestro caso solo disponemos de una asignatura puede darse el caso de que tengamos más, por lo que se ha diseñado ya para esta última opción.

A continuación le asignaremos un nombre al examen, y después la nota máxima que va a valer este examen, típicamente 10.

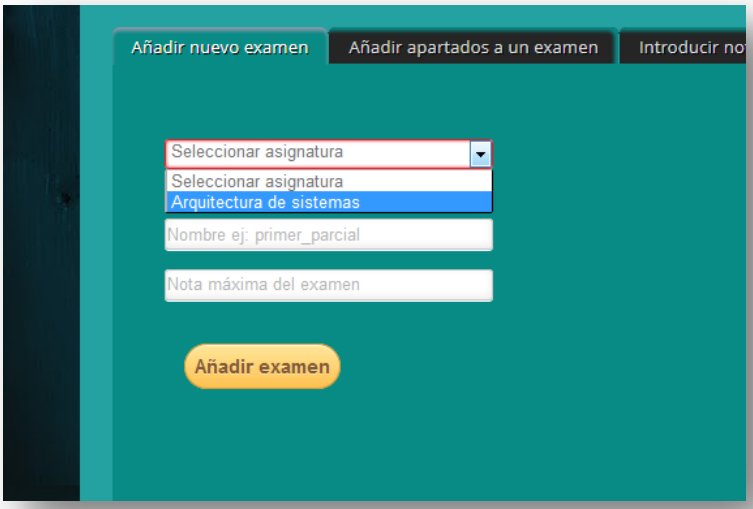


FIGURA 42: AÑADIENDO UN EXAMEN A LA BASE DE DATOS

En la segunda pestaña podemos añadir apartados a los exámenes.

Para ello como en el primer caso, seleccionamos la asignatura, y después el examen al que le queremos agregar el apartado.

Lo siguiente será el nombre del apartado y la nota máxima que tendrá el apartado.

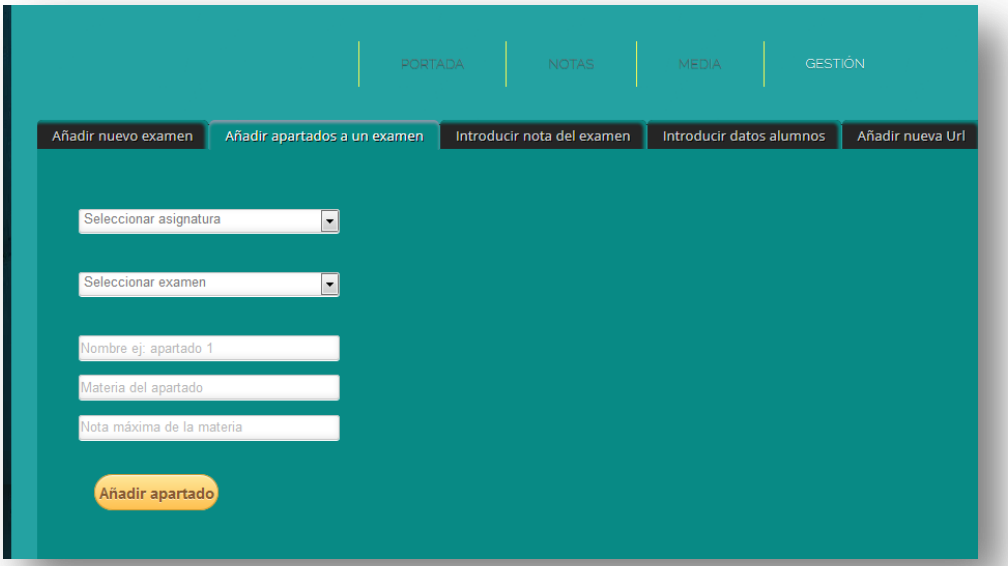


FIGURA 43: AÑADIENDO APARTADOS A UN EXAMEN

En la tercera pestaña introducimos la nota que ha sacado el alumno en el examen.

Es necesario hacerlo de forma independiente a los apartados ya que la nota del examen solo se apondrá una vez, mientras que apartados por examen podemos tener más de uno.

FIGURA 44: AÑADIENDO LA NOTA DEL EXAMEN DE UN ALUMNO

En la cuarta pestaña relacionamos alumnos con exámenes y así ponemos las notas de los alumnos.

En primer lugar de nuevo, elegimos la asignatura. A continuación elegimos el examen de todos los que hay en la base de datos para el que queremos añadir la nota.

Una vez seleccionado el examen en la pestaña de "selecciona apartado" nos aparecerán únicamente los apartados que corresponden al examen seleccionado, y después en el desplegable "selecciona materia" pasará exactamente lo mismo, tendremos solo las materias que corresponden al apartado seleccionado.

Por último añadimos la nota que el alumno ha obtenido en ese apartado y para esa materia en concreto.

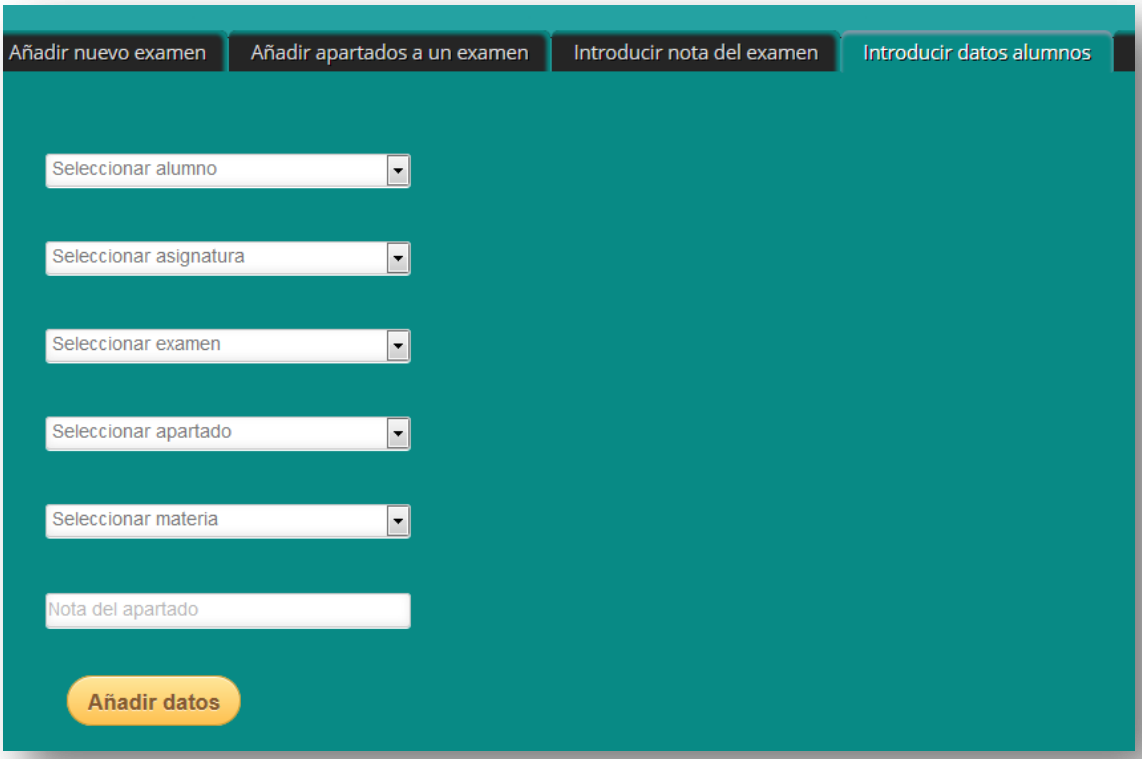


FIGURA 45: AÑADIENDO DATOS DEL ALUMNO

La siguiente pestaña nos sirve para añadir urls a las materias.

Ya dijimos que para añadir una parte semántica a la web tendríamos urls de las materias que nos llevarían a páginas de apuntes, ejercicios etc que ayudarían al alumno a preparar la asignatura.

En esta pestaña basta con decidir a qué materia queremos añadir la *url* y añadirla.

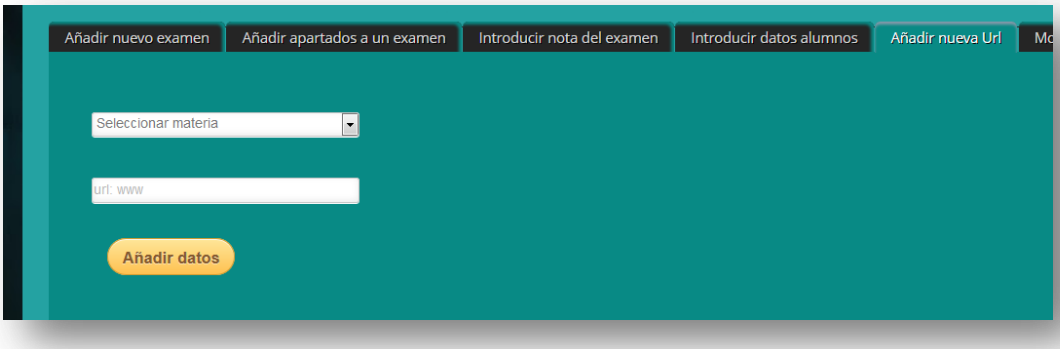


FIGURA 46: AÑADIENDO *URL* A LA BASE DE DATOS

Por último tenemos la pestaña que nos modifica la nota de los alumnos.

Como sabemos cuándo se suspende un examen hay una revisión en la que el profesor puede modificar la nota del alumno o puede ocurrir que nos confundamos al pasar la nota del examen a la página web.

El procedimiento será igual que en el caso de la pestaña tres, donde agregábamos la nota del alumno, pero en el servlet en vez que una QUERY haremos un UPDATE para modificar valores de la base de datos.

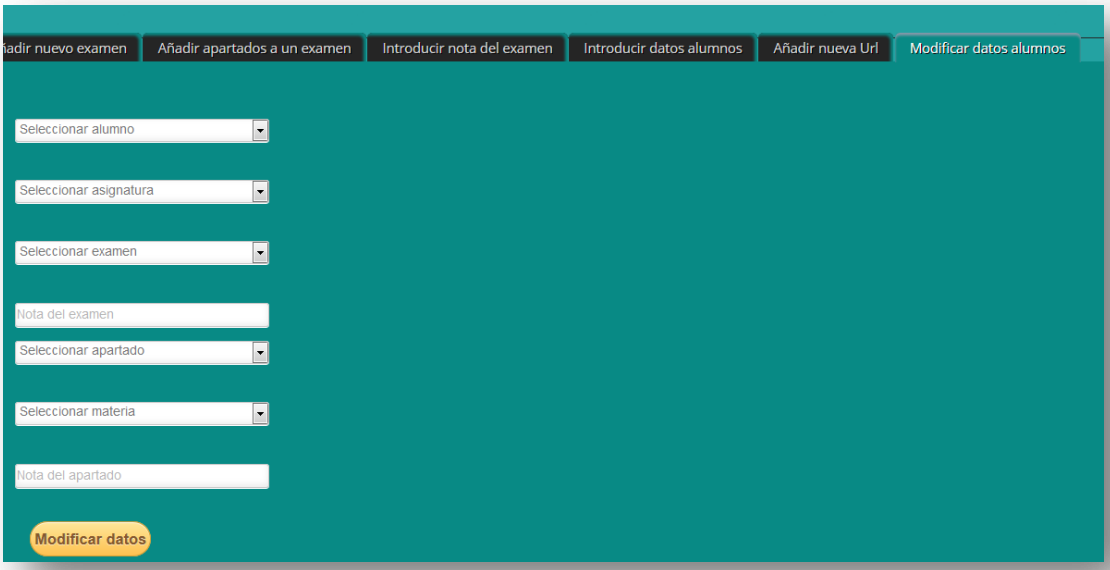


FIGURA 47: MODIFICANDO DATOS DEL ALUMNO

5.2. Conclusiones

Como hemos podido comprobar en este capítulo se han conseguido los objetivos propuestos para el presente proyecto final de carrera.

La herramienta diseñada muestra de una forma clara y simple las estadísticas de las notas necesarias para que los alumnos y profesores puedan hacer un mejor seguimiento de las asignaturas.

Además, el entorno es amigable, por lo que no da lugar a confusión ni es molesto de utilizar.

Por otro lado se permiten las modificaciones necesarias por los profesores para que se pueda tener un seguimiento favorable del curso en cuento a poner notas a los alumnos y/o modificarlas.

Capítulo 6.

Conclusiones, Líneas futuras y mejoras

En este capítulo vamos a concluir el presente proyecto final de carrera explicando nuestra visión de los resultados obtenidos.

Además, vamos a proponer unas mejoras y cambios que pueden ser interesantes de cara al futuro.

6.1. Conclusiones

Una vez terminada la aplicación y presentado en este documentos los resultados obtenidos, podemos decir que se han cumplido los objetivos de los que partíamos inicialmente.

Como hemos podido ver se ha creado una aplicación que cumple los requisitos que se esperaban.

- Consideramos que es una aplicación fácil de utilizar.
- Se considera que ayuda al alumno con el seguimiento de las asignaturas que imparte en la universidad Carlos III de Madrid.

- Consideramos que ayuda a los profesores para el seguimiento de la asignatura que imparte.
- Se visualiza de forma fácil y rápida las estadísticas de la asignatura en cuanto a medias, suspensa y aprobada.
- Proporciona una gestión fácil a los profesores para que puedan modificar asignaturas y añadir exámenes.

6.2. Líneas futuras

En este capítulo vamos a tratar las posibles líneas futuras que se pueden dar en este proyecto así como las mejoras.

1. Aplicación para varias asignaturas

Como hemos comentado a lo largo de este documento, el proyecto está desarrollado para una sola asignatura, pero podría desarrollarse para varias.

Para pode utilizar la Web con más de una asignatura serán necesario unos pasos:

- Añadir la asignatura a la base de datos
- Modificar el archivo LoginServlet.java para que se genere un archivo *JSON* por cada una de las asignaturas de la base de datos.

  Nombre_asignatura.*json*

  Cuando el usuario se autentique se crearan tantos archivos *JSON* como asignaturas tenga el alumno, o imparta el profesor.

- Modificar la visualización para poder seleccionar de que asignatura queremos ver las notas y su media.
- En este caso, podríamos mejorar la aplicación haciendo que el alumno pudiera ver su "media" global, es decir, ver cómo va en general en la carrera haciéndole una estadística de todas las asignaturas.

2. Cambios en la visualización

- Añadir más gráficos. Se podrían añadir más gráficos que representen la media de los alumnos en torno al aprobado. Gráficos de dispersión
- También sería interesante añadir gráficos que muestren el uso de la aplicación.

3. Añadir cuestionarios

- Elaborar un cuestionario para el profesor que nos muestre después en los nuevos gráficos la utilidad de la herramienta. Así podremos ver si:
    - Realmente la herramienta es utilizada
    - Si es fácil de manejar
    - Si los resultados obtenido son los que interesan
    - Si es necesario añadir otros gráficos para su uso o modificar algo.
- Elaborar un cuestionario para alumnos reales. Se pueden hacer pruebas con alumnos reales para ver si la información les es útil, si ellos usan la herramienta, si los enlaces les ayudan. Al igual que para los profesores, podemos añadir más ejercicios, o información que para el alumno fuera necesaria.

Capítulo 7.

Presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: Sandra Pinero Sánchez

2.- Departamento: Telemática

3.- Descripción del Proyecto:

- Titulo	**Herramienta para la publicación electrónica y visualización de calificaciones de evaluación cointinua con información semántica**
- Duración (meses)	6
Tasa de costes Indirectos:	20%

4.- Presupuesto total del Proyecto (valores en Euros):

Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
					0,00	
Estevez Ayres, Iria		Ingeniero Senior	0,9	4.289,54	3.860,59	
Pinero Sánchez, Sandra		Ingeniero	6	2.694,39	16.166,34	
					0,00	
					0,00	
		Hombres mes	**6,9**	**Total**	20.026,93	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
		100		60	0,00
Ordenador de desarrollo	800,00	100	6	60	80,00
		100		60	0,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	80,00

Página 1

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
	Total	0,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	20.027
Amortización	80
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	4.021
Total	24.128

Capítulo 8.

Referencias

[1] Web 2.0(Consultado el 25/09/2013)
http://cursoweb20.net/2012/02/22/que-es-web-20/

[2] Maestros del Web 2.0 (Consultado el 25/09/2013)
http://www.maestrosdelweb.com/editorial/web2/

[3] TCP/IP (Consultado el 25/09/2013)
http://www.w3schools.com/tcpip/tcpip_intro.asp

[4] TCP/IP Protocolos OSI (Consultado el 25/09/2013)
http://resplandorrc.blogspot.com.es/2010/10/pila-o-modelo-osi-y-tcp-ip.html

[5] HTML Y CSS3 (Consultado 3/04/2013)
http://blogs.msdn.com/b/esmsdn/archive/2011/11/07/c-243-mo-crear-mi-primera-p-225-gina-en-html5-y-css3.aspx

[6] CSS (Consultado el 5/04/2013)
http://www.cristalab.com/tags/css/

[7] JavaScript (Consultado 10/04/2013)

http://paperjs.org/

[8] Ejemplos JavaScript (Consultado 10/04/2013)

http://www.uv.es/jac/guia/jscript/javascr.htm

[9] Apache Tomcat (Consultado el 27/07/2013)

http://tomcat.apache.org/

[10] Apache Tomcat (Consultado el 20/08/2013)

http://www.lsi.us.es/docencia/get.php?id=1923

[11] Guía HTML5 (Consultado el 10/04/2013)

https://mejorando.la/guia-html5/

[12] HTML5 (Consultado 10/04/2013)

http://www.comocrearunsitioweb.com/conceptos-basicos-html5

[13] Etiquetas HTML5 (Consultado 10/04/2013)

http://www.dirinfo.unsl.edu.ar/dweb/listaetiquetasHTML5.pdf

[14] Java Util (Consultado el 27/07/2013)

http://javainutil.blogspot.com.es/2013/04/

[15] Aprenda Servlets de Java como si estuviera en segundo.
Javier García de Jalón, José Ignacio Rodríguez, Aitor Imaz
San Sebastián, Abril 1999 (Consultado el 20/08/2013)
http://mat21.etsii.upm.es/ayudainf/aprendainf/javaservlets/servlets.pdf

[16] *JSON* (Consultado el 20/06/2013)
http://www.*json*.org/

[17] *JSON* Ejemplos (Consultado el 20/06/2013)
http://adobe.github.io/Spry/samples/data_region/*Json*DataSetSample.html

[18] MySQL (Consultado 20/08/2013)
http://www.mysql.com/

[19] Tutorial Eclipse (Consultado el 20/08/2013)
http://www.uv.es/~jgutierr/MySQL_Java/TutorialEclipse.pdf

[20] Librería RGraph (Consultado el 15/04/2013)
Richard Heyes
http://www.rgraph.net/

[21] Canvas Demos (Consultado el 15/04/2013)
http://www.canvasdemos.com/type/tools/