Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Novedades Electrónicas: Herramienta informática de gestión de faltas de asistencia.

Grado en Ingeniería Mecánica

ALUMNO: Luis Astorga Boccherini

DIRECTORES: Norberto Fernández García

CURSO ACADÉMICO: 2015-2016

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Novedades Electrónicas: Herramienta informática de gestión de faltas de asistencia.

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

El control de asistencia se considera requisito indispensable para garantizar la obtención de las capacidades teóricas y prácticas en las que la Escuela Naval Militar forma a sus alumnos.
En el presente Trabajo de Fin de Grado se realiza un estudio del sistema de control de asistencia en la Escuela Naval Militar, exponiendo sus principales carencias. Se propone informatizar el sistema, programando una herramienta que opera sobre bases de datos relacionales mediante una interfaz web dinámica que permite automatizar el registro de asistencia a bordo de la Escuela Naval.
Se generan las tablas de una base de datos relacional con los datos aportados por la Escuela Naval mediante la herramienta MySQL, mientras que la interfaz web se programa con los lenguajes PHP y HTML. La herramienta programada genera una base de datos completa que cubre las necesidades de la ENM, además permite la exportación de los datos de asistencia de los alumnos a formato Excel para su análisis por parte del personal de Jefatura de Estudios de la Escuela Naval.
Se prueba la aplicación en un entorno realista comprobando su eficacia y se analizan los resultados obtenidos. Además se proponen líneas de mejora de la aplicación realizables a corto plazo.

PALABRAS CLAVE

Control de asistencia, Escuela Naval Militar, PHP, MySQL, HTML, programación web

AGRADECIMIENTOS

Me gustaría agradecer a la Jefatura de Estudios de la Escuela Naval Militar la disposición permanente para resolver mis inquietudes y el interés mostrado por el trabajo. Han invertido su valioso tiempo en reunirse conmigo y han aclarado muchas dudas acerca del control de asistencia de la Escuela Naval.

CONTENIDO

Contenido 1

Índice de Figuras 4

1 Introducción y objetivos 6

1.1 Contextualización 6

1.2 Importancia del control de asistencia, centros militares 7

1.3 Control de asistencia en la ENM actualmente. Ventajas e inconvenientes 7

1.4 Apuesta por la tecnología 9

1.5 Objeto y motivación del trabajo 9

1.6 Estructura de la memoria 9

2 Estado del arte 11

2.1 Alternativas 11

2.1.1 LEXTREND y Tracesign 11

2.1.2 Atenea 12

2.1.3 MIDS 12

2.2 Herramientas a utilizar en la realización del trabajo 13

2.2.1 HTML 13

2.2.2 Apache 14

2.2.3 MySQL 15

2.2.4 PHP 16

3 Desarrollo del TFG 18

3.1 Definición del problema a resolver y decisiones iniciales 18

3.2 Configuración del entorno de trabajo y preparación de las herramientas 19

3.3 Programación MVC 23

3.4 Estructura de la base de datos y de las vistas de la aplicación 24

3.5 Programación de la base de datos con MySQL 29

3.6 Presentación web con HTML 31

3.6.1 Utilidad del lenguaje HTML 31

3.6.2 Etiquetas HTML 31

3.6.3 CSS 34

3.6.4 Interacción del usuario con PHP a través de HTML 34

3.7 Programación de acciones con PHP 34

3.7.1 Características principales del lenguaje PHP 34

3.7.2 Variables 35

3.7.3 Estructuras de control 35
3.7.4 Funciones 35
3.7.5 Sesiones 36
3.8 Definiendo la aplicación web 37
3.8.1 Estructura del programa 37
3.8.2 Vistas de la aplicación 38
3.8.3 Estructura de control 44
3.8.4 Estructura de modelo 45
3.8.5 Lectura, introducción y borrado de datos en la BDR 46
3.9 Uso y gestión de la aplicación 47
4 Validación 49
4.1 Generación de una base de datos para pruebas en Excel 49
4.2 Lectura y registro de datos de Excel en la base de datos. 49
4.3 Prueba de funcionamiento de la aplicación 50
4.3.1 Uso normal sin errores 50
4.3.2 Uso forzando errores para la detección de fallos 54
4.4 Almacenamiento de novedades en la base de datos y exportación a formato Excel 57
5 Conclusiones y líneas futuras 58
5.1 Cumplimiento de objetivos 58
5.2 Generación automática de códigos de barras 59
5.2.1 Generación automática de códigos de barras 59
5.2.2 Procesado y almacenamiento de estadísticas 59
5.2.3 Alertas a profesores por correo electrónico 59
5.2.4 Lectura desde dispositivos móviles 60
5.3 Conclusión 60
6 Bibliografía 61
Anexo I: Comandos a introducir para crear la base de datos 64
Anexo II: Hoja de estilos CSS 66
Anexo III: index.php 68
Anexo IV: vista1.php 70
Anexo V: vista2.php 72
Anexo VI: vista3.php 74
Anexo VII: vista4.php 75
Anexo VIII: vista5.php 77

Anexo IX: control.php....78
Anexo X: modelo.php....81
Anexo XI: convertir.php....94
Anexo XII: convertirids.php....95
Anexo XIII: Tablas excel de la base de datos para la realización de pruebas de validación....96
Anexo XIV: Tarjetas de profesor y ficha de clase....99

ÍNDICE DE FIGURAS

Figura 1-1 Presentación de la Escuela Naval Militar 6
Figura 1-2 Parte de clase de la ENM 8
Figura 2-1 Logo de la empresa LEXTREND 11
Figura 2-2 Logo del Software Atenea 12
Figura 2-3 Tim Berners-Lee 14
Figura 2-4 Logo de la Apache Software Foundation, promotora del servidor Apache 14
Figura 2-5 Ejemplo de atributos y registros de una tabla en una base de datos 15
Figura 2-6 Logo de MySQL 16
Figura 2-7 Logo de PHP 17
Figura 3-1 Logo de Ubuntu 19
Figura 3-2 Pantalla de inicio de la herramienta VirtualBox 20
Figura 3-3 Página automática de confirmación del funcionamiento de Apache 22
Figura 3-4 Página automática de confirmación del funcionamiento de PHP 23
Figura 3-5 Esquema del modelo MVC 24
Figura 3-6 Tablas ideadas para la creación de la BDR 26
Figura 3-7 Idea de la presentación de inicio 27
Figura 3-8 Idea de la vista principal de la aplicación 28
Figura 3-9 Idea de la presentación para introducir novedades 28
Figura 3-10 Presentación de MySQL de las tablas utilizadas 30
Figura 3-11 Descripción de la tabla Novedades 30
Figura 3-12 Ejemplo de etiquetas de texto HTML y resultado 32
Figura 3-13 Ejemplo de etiquetas de interacción con HTML y resultado 33
Figura 3-14 Ejemplo de etiquetas de creación de tablas y resultado 34
Figura 3-15 Ejemplo de código y resultado de la función *echo* 36
Figura 3-16 Ejemplo de código y resultado de una función 36
Figura 3-17 Presentación index.php 38
Figura 3-18 Presentación de vista1.php 39
Figura 3-19 Ventana "*pop-up*" al introducir el grupo 39
Figura 3-20 Presentación de vista1.php con grupos introducidos 40
Figura 3-21 Presentación de la vista2.php 41
Figura 3-22 Presentación de vista3.php 42
Figura 3-23 Presentación de vista4.php 43
Figura 3-24 Presentación de vista5.php 44
Figura 3-25 Segmento de código de control.php 45

Figura 4-1 Presentación en MySQL de los datos de la tabla Profesor 50
Figura 4-2 Presentación de index.php al introducir un código.....50
Figura 4-3 Presentación de vista1.php con el nombre del profesor 51
Figura 4-4 Presentación de vista2.php introduciendo la novedad de los alumnos.....52
Figura 4-5 Presentación de vista1.php con los grupos introducidos 53
Figura 4-6 Resultado de la exportación a Excel de las novedades guardadas 53
Figura 4-7 Presentación de index.php al introducir un código de profesor incorrecto 54
Figura 4-8 Presentación de vista1.php al introducir un código de grupo incorrecto.....54
Figura 4-9 Sección de vista4.php en la que se aprecia el error al introducir la novedad 55
Figura 4-10 Presentación de vista3.php en la que se introduce el grupo a borrar.....55
Figura 4-11 Sección de vista1.php en la que se aprecia el borrado del grupo InformáticaG1.....55
Figura 4-12 Presentación de index.php al introducir un código de administrador erróneo 56
Figura 4-13 Resultado de la exportación a Excel de la novedad guardada.....56
Figura 4-14 Datos de la tabla Guardar en MySQL.....57

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contextualización

La Escuela Naval Militar, o ENM, es el único centro de formación de oficiales de la Armada, se encuentra en la ciudad de Marín, en la provincia de Pontevedra. La ENM lleva formando oficiales en Marín desde el año 1943, fecha en que fue trasladada desde su anterior ubicación en la Población naval de San Carlos, San Fernando, Cádiz [1] .

La formación como oficial de la Armada se compagina con el Grado en Ingeniería Mecánica de la Universidad de Vigo, impartido por el Centro Universitario de la Defensa [2], desde el curso académico 2010/2011.

Actualmente el acceso en la Armada se puede realizar a uno de los cuerpos siguientes, Cuerpo General, Cuerpo de Infantería de Marina, Cuerpo de Intendencia o Cuerpo de Ingenieros de la Armada. A los cuatro Cuerpos se puede acceder con titulación previa, pero sólo a los dos primeros se puede acceder sin titulación universitaria.

Figura 1-1 Presentación de la Escuela Naval Militar [1]

En caso de acceder con titulación, en la ENM se realizarán dos años de formación militar, un único año en el caso del Cuerpo de Ingenieros, en los que recibirán una completa formación militar. Sin embargo, la gran mayoría de alumnos de la Escuela Naval acceden a los Cuerpos General y de Infantería de Marina sin titulación previa.

Esta preparación de oficiales consiste en cinco años de formación humana, civil, técnica y militar, con gran cantidad y variedad de clases, prácticas, seminarios, embarques, actividades culturales y profesionales y actividades de carácter físico.

1.2 Importancia del control de asistencia, centros militares

En numerosos centros educativos se lleva un control de la asistencia, que intenta ser lo más fiable y claro posible. El objetivo es ser capaz de evaluar si un alumno ha adquirido las competencias necesarias para pasar una asignatura y está preparado para el ejercicio profesional.

Estas competencias, especialmente las prácticas son siempre importantes, pero ganan especial relevancia en un centro como es la Escuela Naval Militar, destinada a formar oficiales listos para navegar, dirigir equipos y tomar decisiones. Estas capacidades se obtienen principalmente mediante una extensa formación teórica y mediante el ejercicio de simulaciones de situaciones reales y prácticas, que se llevan a cabo semana tras semana en la ENM.

La asistencia es por tanto de gran importancia para la formación completa del alumno, y debe por ello ser primordial para el centro docente ser capaz de llevar un control de esta.

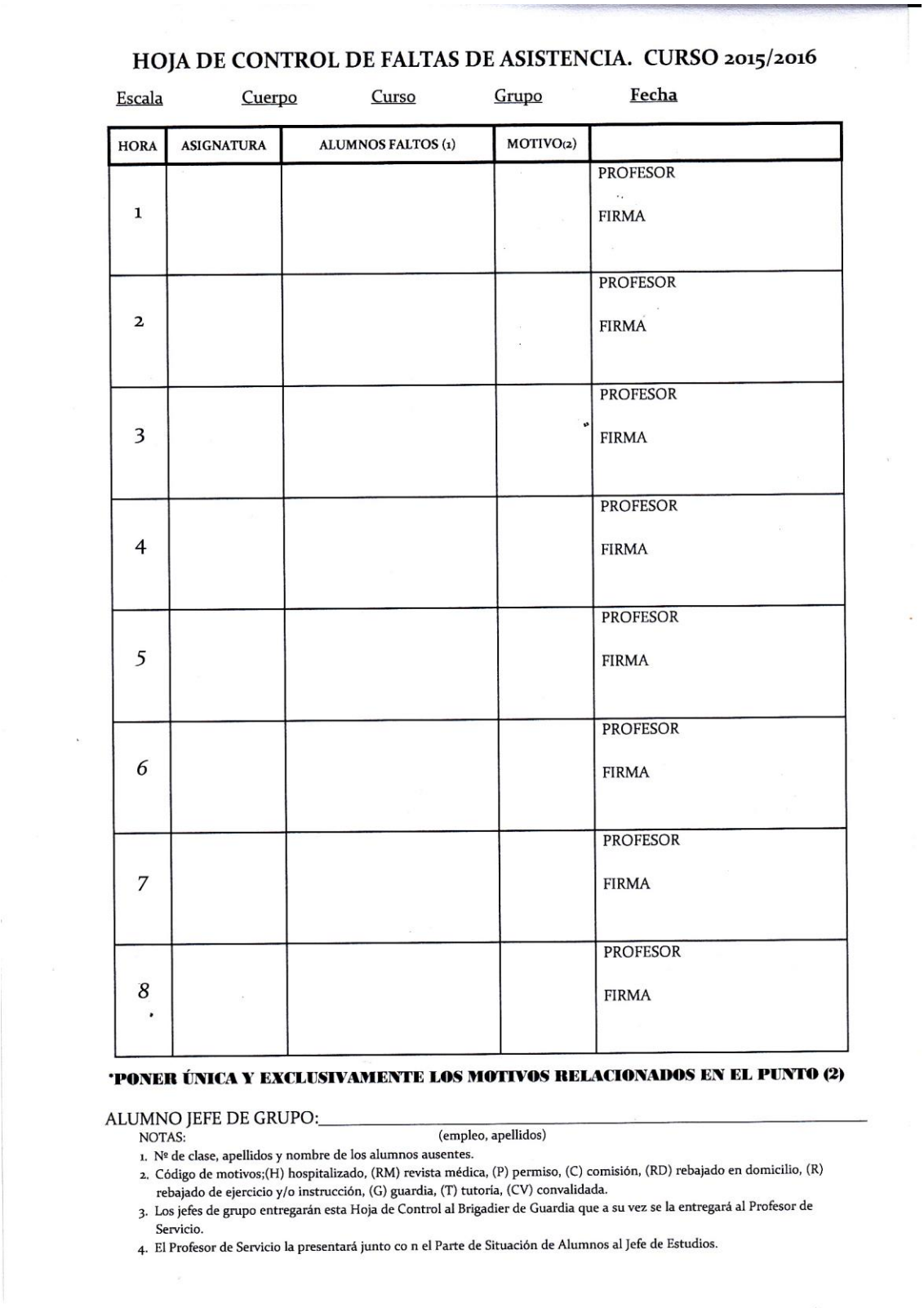
Por este motivo, la Escuela Naval es consciente de la importancia que tiene el ser capaz de controlar a los alumnos, sabiendo a qué actividades acuden y a cuáles faltan, además de conocer el motivo que impulsa esta falta. Esta es la única forma en la que la ENM puede certificar que sus alumnos están preparados para el desempeño profesional y cuentan con la experiencia y conocimientos que se espera de un oficial de la Armada formado cinco años en un centro de excelencia.

1.3 Control de asistencia en la ENM actualmente. Ventajas e inconvenientes

Los alumnos de la Escuela Naval Militar están organizados por cursos, cada curso supone una Brigada, y dentro de las Brigadas, se reparten en grupos de clase. El grupo de clase supone la unidad básica de trabajo de los alumnos, un grupo de clase normalmente irá completo a todas las clases, ya sean teóricas o prácticas, del día. La Escuela Naval cuenta actualmente con siete brigadas, cuatro de ellas divididas en 7 grupos, una en 8, otra en 2 y la última forma un único grupo. Además, los alumnos de intercambio procedentes del extranjero, los alumnos que repiten alguna asignatura y aquellos que estuvieron el año pasado en la Escuela Naval de la Armada estadounidense en Annapolis, forman grupos independientes.

Los grupos de clase son de entre ocho y diez alumnos, dependiendo del tamaño de la brigada en cuestión, y se mantienen fijos durante todo el año. Dentro del grupo de clase existe el papel del jefe de grupo, responsable de su grupo a la hora de controlar la asistencia a clase y dar la novedad al profesor antes de cada hora, indicándole los alumnos faltos y los motivos de la ausencia. El papel de jefe de clase no es fijo, sino que los alumnos tienen libertad para turnarse como les resulte más conveniente.

El jefe de cada grupo ha de plasmar la novedad de su grupo en el Parte de Clase, este es el documento que se utiliza para registrar las faltas y retrasos de los alumnos a clase. El documento, como se puede ver en la Figura 1-2, es una plantilla para introducir las distintas actividades que se realizan durante el día, las faltas y los motivos de dichas faltas. A continuación, el profesor responsable de la actividad firma la novedad en la última de las columnas.



HOJA DE CONTROL DE FALTAS DE ASISTENCIA. CURSO 2015/2016

Escala Cuerpo Curso Grupo **Fecha**

HORA	ASIGNATURA	ALUMNOS FALTOS (1)	MOTIVO(2)	
1				PROFESOR FIRMA
2				PROFESOR FIRMA
3				PROFESOR FIRMA
4				PROFESOR FIRMA
5				PROFESOR FIRMA
6				PROFESOR FIRMA
7				PROFESOR FIRMA
8				PROFESOR FIRMA

*PONER ÚNICA Y EXCLUSIVAMENTE LOS MOTIVOS RELACIONADOS EN EL PUNTO (2)

ALUMNO JEFE DE GRUPO:_____
(empleo, apellidos)

NOTAS:
1. Nº de clase, apellidos y nombre de los alumnos ausentes.
2. Código de motivos;(H) hospitalizado, (RM) revista médica, (P) permiso, (C) comisión, (RD) rebajado en domicilio, (R) rebajado de ejercicio y/o instrucción, (G) guardia, (T) tutoría, (CV) convalidada.
3. Los jefes de grupo entregarán esta Hoja de Control al Brigadier de Guardia que a su vez se la entregará al Profesor de Servicio.
4. El Profesor de Servicio la presentará junto co n el Parte de Situación de Alumnos al Jefe de Estudios.

Figura 1-2 Parte de clase de la ENM

El parte de clase es responsabilidad del jefe de grupo, que tiene que saber dónde se encuentran los miembros del grupo en todo momento, conocer el horario y hacer firmar las distintas actividades. En la parte inferior del parte, el alumno se identifica como jefe de grupo, con empleo, nombre y apellidos.

El sistema actual tiene varios inconvenientes. En primer lugar supone una carga de trabajo innecesaria para el jefe de grupo, pues tiene que responsabilizarse de recoger el parte de clase a

primera hora de la mañana, rellenarlo con las actividades del grupo, cubrir las posibles faltas, hacerlo firmar cada hora y entregarlo a última hora de la tarde.

Por otro lado, supone una carga muy grande de trabajo para la Jefatura de Estudios de la Escuela, puesto que cada día se procesan de forma manual alrededor de 60 partes de clase, introduciendo a continuación los datos más relevantes en una hoja Excel.

Además supone un gasto de papel y tinta innecesario, teniendo en cuenta que con un desembolso inicial apostando por la tecnología, se puede lograr eliminar este gasto, pudiendo suponer un ahorro interesante a largo plazo.

Sin embargo, el sistema actual tiene una ventaja principal, a la que no se puede renunciar si queremos sustituir el parte de clase por nuestra aplicación informática. Esta ventaja es que el trabajo del profesor es mínimo, su función no es más que recibir la novedad del jefe de clase y firmar la casilla correspondiente al acabar la sesión. El sistema que pretendemos implementar no debe suponer un trabajo mayor al profesor si queremos que sea viable.

1.4 Apuesta por la tecnología

Las grandes universidades y los centros docentes de prestigio, tanto de España como de todo el mundo utilizan cada vez más las nuevas tecnologías para la gestión de alumnos, la difusión de información de interés, la publicación de las notas y otras acciones que antes se hacían de modo manual.

La Escuela Naval Militar, para ocupar el puesto que le corresponde como centro de formación de excelencia, debe sumarse a esta iniciativa de apostar por el futuro mediante las nuevas tecnologías, que han de suponer comodidades para alumnos, profesores, y un ahorro a largo plazo.

1.5 Objeto y motivación del trabajo

Teniendo en cuenta los apartados anteriores se llega a la conclusión de que es necesario actualizar y mejorar el sistema de gestión de asistencia en la Escuela Naval Militar.

El principal objetivo del presente trabajo consiste en la implementación de una herramienta informática que permita sentar las bases de un programa que gestione y automatice el control de la asistencia a clase de los alumnos de la Escuela Naval Militar, proveyendo estadísticas personalizadas de los alumnos y profesores.

Por otro lado, es imprescindible hacer que el nuevo sistema no suponga un incremento de esfuerzo para profesores o alumnos respecto al sistema actual. Además, se pretende garantizar un procesado de datos sencillo, automático e intuitivo para aliviar la carga de trabajo que se realiza diariamente en Jefatura de Estudios.

La consecución de estos objetivos principales viene motivada por lograr las ventajas siguientes respecto al sistema actual:

- Eliminar la dependencia del parte de clase en papel para el control de asistencia.
- Reducir el consumo de papel, logrando también un ahorro material a medio y largo plazo.
- Acercar la Escuela Naval Militar a las herramientas tecnológicas que utilizan todos los centros educativos de prestigio, categoría a la que debe pertenecer la ENM.

1.6 Estructura de la memoria

Una vez introducida la motivación y objetivos del trabajo, en el capítulo 2 se presentará el actual estado del arte, en el que se estudiarán alternativas similares a la propuesta realizada. También se presentarán las herramientas utilizadas para la realización del programa.

A continuación, en el capítulo 3 se presentarán las diferentes etapas de desarrollo del proyecto, que comienzan con la definición de los problemas a resolver y las decisiones que han sido tomadas.

Se explicarán los pasos a seguir para la instalación de un entorno de trabajo y la presentación de las capacidades de los lenguajes de programación empleados y la estructura del código asociado. Se expondrá también la base de datos utilizada para la realización del proyecto. El desarrollo continúa con la presentación de las páginas de código utilizadas y su función en la ejecución del programa. Por último se expondrá la forma de utilizar la aplicación y las posibilidades que ofrece a los usuarios.

En el capítulo 4 se procederá a la validación del programa creado, poniendo la aplicación a prueba en distintos escenarios y comprobando que los resultados obtenidos son correctos.

Por último, en el capítulo 5 se expondrán las conclusiones sacadas de la realización del presente trabajo y se propondrán las líneas a seguir en un futuro para mejorar el programa y añadirle nuevas funcionalidades.

2 ESTADO DEL ARTE

2.1 Alternativas

En el este apartado se comentarán trabajos previos existentes en el ámbito del presente trabajo, comentando tecnologías desarrolladas en el ámbito de la gestión de asistencia.

2.1.1 LEXTREND y Tracesign

Existen varias herramientas similares a la que se pretende desarrollar en este trabajo. En primer lugar encontramos la plataforma LEXTREND [3], de la empresa homónima, que junto con el programa Tracesign [4] forman una base de datos en línea, que se actualiza mediante los datos introducidos por los alumnos en cada clase. El sistema consta de una tableta contenida en una torre o kiosko en la cual firman los alumnos con un código propio antes de cada clase.

LEXTREND

Figura 2-1 Logo de la empresa LEXTREND [3]

El sistema a través de la tableta envía los datos de los alumnos a un panel web, donde la Escuela lleva el registro de alumnos por curso, asignatura, profesor, fechas, etc. Pueden obtenerse informes de asistencia automáticamente y enviar periódicamente a los alumnos por correo electrónico su porcentaje de asistencia, incluso su situación respecto a la media de asistencia.

Este sistema tiene varias ventajas, pues es similar al objetivo que se busca alcanzar con este trabajo, supone una gran reducción de esfuerzo de organización y dota de gran sencillez la recuperación de datos y estadísticas.

Sin embargo, tiene carencias en cuanto a las competencias que buscamos en este proyecto. No permite al alumno falto excusar su falta, por lo que los motivos de ausencia justificados no se tienen en cuenta. Un ejemplo de ausencia justificada en la Escuela Naval puede ser una guardia o una hospitalización.

Además, supone una pérdida de tiempo notable para los alumnos tener que pasar de uno en uno por el terminal de la tableta para registrarse en la clase. Teniendo en cuenta que hay clases de 40 alumnos, supone una gran pérdida de tiempo en el ya apretado horario.

Por último tiene otro inconveniente, y es que el sistema solo contempla clases teóricas en un aula, pero la Escuela Naval necesita llevar también registro de sus actividades prácticas de instrucción y adiestramiento. Instalar este sistema para las lanchas de instrucción, salidas al campo, o actividades de refuerzo físico puede ser complicado, pues está diseñado para un uso exclusivo en aulas.

2.1.2 Atenea

El software Atenea [5] de la empresa Ender [6] está organizado para academias de enseñanza, y permite controlar lo que sucede en las aulas, desde el punto de vista del profesor, de los alumnos y de la dirección del centro. El software está diseñado para la gestión completa de la academia, por lo que incluye funcionalidades que van más allá de los objetivos planteados en el presente proyecto, como puede ser la gestión de contratos, pagos a profesores, cobros de matrículas y gastos a alumnos, la gestión de sustituciones en caso de que haya profesores faltos, o el listado de matrículas de cada alumno.

Figura 2-2 Logo del Software Atenea [5]

La función más interesante de este programa para nuestros propósitos es la gestión de asistencia. Atenea permite en tiempo real introducir los datos de asistencia a cada clase, y es multiplataforma, por lo que el profesor puede utilizar su ordenador, móvil o tablet. Sin embargo, Atenea no permite a sus alumnos conocer sus estadísticas de faltas a clase.

La principal ventaja de este programa, es la automatización y el diseño del software multiplataforma que se adapta a las necesidades de la Escuela Naval. Es muy cómodo para los profesores pues aporta mucha información acerca de sus clases.

Por otro lado, presenta varios inconvenientes. En primer lugar, el programa ofrece demasiadas funcionalidades, muchas de escaso interés para la Escuela Naval. Supone una carga de trabajo la actualización del software con los datos de la Escuela, y el control de asistencia no es más que una funcionalidad secundaria del sistema. Además las estadísticas las recoge para cada profesor y no es capaz de juntarlas alumno por alumno, para saber a cuántas clases ha faltado en total. Por otro lado, de forma análoga al programa de la empresa LEXTREND [3], no tiene en cuenta la justificación de las faltas, por lo que no permite introducir el motivo de una posible falta en el sistema.

2.1.3 MIDS

Los sistemas vistos anteriormente en este capítulo han sido desarrollados por y para el mundo civil. Sin embargo, la que es posiblemente la Academia de formación de Oficiales más prestigiosa del mundo, la Escuela Naval de Annapolis de la *United States Navy* (en adelante USNA), cuenta con un sistema conocido por sus siglas como MIDS.

MIDS, es el diminutivo de *Midshipmen* (Guardiamarinas), rango de los alumnos de la Academia, pero además proviene de *Midshipmen Information System* (Sistema de Información para Guardiamarinas). Es muy complicado encontrar información del MIDS, debido a que es un programa interno de la USNA. Sin embargo sabemos que es una herramienta informática que gestiona las notas de los alumnos, trabajos, correo, etc. La función del MIDS que más nos interesa es la de gestión de asistencia.

La interfaz del control de asistencia es muy sencilla, al profesor se le presenta por pantalla una tabla que contiene el listado de los alumnos que deben asistir a su clase, acompañada de un

desplegable de motivos de retrasos, o ausencias. Además incluye un enlace que abre una foto del alumno en cuestión, para facilitar al profesor la labor de identificación del alumno.

Este sistema es muy similar al que queremos implementar en el presente trabajo: permite la gestión de asistencia por parte del profesor. Además se trata de un software que lleva varios años en funcionamiento, prueba de su fiabilidad y adaptibilidad a las necesidades de una Escuela Naval. Sin embargo, no es viable implementar este sistema pues es cerrado y exclusivo para la USNA en su academia de Annapolis.

2.2 Herramientas a utilizar en la realización del trabajo

En la presente sección se presentarán brevemente las herramientas empleadas para la realización del trabajo, explicando su utilidad y resumiendo su historia y evolución.

2.2.1 HTML

2.2.1.1 Presentación

HTML [7], [8] (*HyperText Markup Language*) es el lenguaje en el que se escribe la gran mayoría de las páginas web. Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseño, es muy fácil de aprender y escribir.

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado *World Wide Web Consortium* (Consorcio de la Red de Alcance Global), más conocido como W3C [9]. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo.

HTML no es un lenguaje de programación, sino de etiquetado que permite etiquetar el texto para estructurar la presentación web en el navegador. Las etiquetas se definen con los símbolos “menor que” (<) y “mayor que” (>). El texto se introduce entre dos etiquetas, una de apertura y una de cierre. La de cierre se establece añadiendo una barra inclinada (/) tras el símbolo “menor que”.

Por ejemplo, para representar un párrafo (definido por la etiqueta “p”) el texto irá introducido en la página HTML de la siguiente forma:

- * Introducir párrafo *

La presentación del programa debe ser entendida en todas las plataformas y sistemas operativos. Por ello el lenguaje HTML es fundamental para crear la interfaz de usuario de la aplicación de forma que sea visible en todos los navegadores y sistemas operativos. Además se complementa perfectamente con el lenguaje PHP, descrito en el apartado 2.2.4, lo que resulta ideal para facilitar la programación y procesado.

2.2.1.2 Historia

El origen de HTML se remonta a 1980, cuando el físico Tim Berners-Lee propuso un nuevo sistema de hipertexto para compartir documentos [10]. Los sistemas de hipertexto habían sido desarrollados años antes. En el ámbito de la informática, el hipertexto permitía que los usuarios accedieran a la información relacionada con los documentos electrónicos que estaban visualizando.

Figura 2-3 Tim Berners-Lee [11]

Tras finalizar el desarrollo de su sistema de hipertexto, Tim Berners-Lee lo presentó a una convocatoria organizada para desarrollar un sistema de hipertexto para Internet. Después de unir sus fuerzas con el ingeniero de sistemas Robert Cailliau, presentaron la propuesta ganadora llamada WorldWideWeb (W3).

En el año 1995 se presenta la versión HTML 2, aunque hoy en día es considerada la primera versión de HTML, que aún no soportaba tablas.

HTML 3.2 se lanzó en 1997, impulsada por el organismo de estandarización W3C, e introducía los últimos avances en páginas web, como las aplicaciones de Java.

HTML 4 salió en 1998, aunque se revisó como HTML 4.1 en 1999. Permite incluir pequeños programas o *scripts* en las páginas web, mejora la accesibilidad de las páginas diseñadas, permite incluir tablas complejas y añade mejoras en los formularios.

Actualmente se trabaja en la versión HTML 5, que cuenta ya con varios borradores de versiones que pueden utilizarse. Añade nuevas funcionalidades, especialmente en cuanto a gestión de contenidos multimedia [8].

2.2.2 Apache

2.2.2.1 Presentación

El servidor Apache [12] es usado principalmente para publicar páginas web, tanto estáticas como dinámicas en la *World Wide Web*. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación en Apache, por lo que utilizarán características propias de este servidor web.

Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP [13] (Linux, Apache, MySQL, PHP), junto a MySQL y el lenguaje de programación PHP, que se explicarán a continuación. Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

Figura 2-4 Logo de la Apache Software Foundation, promotora del servidor Apache [12]

Apache es el servidor más utilizado en la red [14] por sus principales ventajas. Es modular, de código abierto, multi-plataforma y su popularidad hace que sea muy fácil de conseguir y de obtener ayuda o soporte. En la realización de este trabajo se utiliza la versión 2.14.18.

2.2.2.2 Historia

El servidor Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP y la noción de sitio virtual. El lanzamiento inicial de Apache fue en el año 1995 y su nombre se debe a una tribu indígena estadounidense. Además el Apache original consistía en un conjunto de parches aplicados a un servidor ya existente, el servidor NCSA (*National Center for Supercomputing Applications*); en inglés, un servidor "parcheado" (*a patchy server*) se pronuncia de forma muy similar a Apache Server [15].

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation [16] dentro del proyecto HTTP Server (httpd). Apache presenta una amplia variedad de características altamente configurables [17].

2.2.3 MySQL

2.2.3.1 Presentación

Una base de datos es un conjunto de datos estructurados que pertenecen a un ámbito funcional. Físicamente, una base de datos se corresponde con un cierto número de archivos almacenados en un dispositivo de almacenamiento [14] [18].

Estos datos son gestionados por un programa llamado SGBD, Sistema de Gestión de Base de Datos. El programa ofrece diferentes características, como acceso a los datos, gestión de las actualizaciones, comprobaciones de integridad, control de la seguridad de acceso, etc.

En una base de datos relacional, los datos se organizan en tablas enlazadas de manera lógica. Una tabla incluye atributos que describen registros. Los atributos se representan en las columnas, cada columna representa un atributo de la tabla. Debajo, en cada fila se introducen los registros asociados. Se muestra un ejemplo en la Figura 2-5.

Nombre	Apellido	Número
Pedro	García	7
María	Pérez	12

Figura 2-5 Ejemplo de atributos y registros de una tabla en una base de datos

En el ejemplo, los atributos son "Nombre", "Apellido" y "Número", y los registros son ("Pedro", "García", "7"); y ("María", "Pérez", "12").

El objetivo de una base de datos relacional es almacenar en diferentes tablas los datos correspondientes a diferentes entidades y relacionarlas de forma eficiente, evitando redundancias y facilitando su procesado.

Para consultar grandes bases de datos, se ideó el lenguaje SQL (*Structured Query Language*). UN lenguaje que permite obtener la información de la base de datos.

MySQL [19] (*My Structuted Query Language*) no es más que un SGBD relacional basado en el lenguaje SQL. Este sistema es responsable de controlar las bases de datos, sus componentes, mantener

la integridad de los datos, recibir las consultas de petición, creación y actualización de la información, junto con otras muchas funcionalidades.

Figura 2-6 Logo de MySQL [19]

El presente trabajo necesita procesar gran cantidad de información de alumnos, profesores, asignaturas, y motivos de ausencias con el fin de obtener estadísticas. La gestión de la base de datos requiere de la herramienta MySQL. Además gracias a su carácter libre y gratuito, supone un coste cero para la Escuela Naval.

2.2.3.2 Historia

MySQL [20] es el sistema de administración de bases de datos de código abierto (*Open Source*) más extendido del mundo [14]. Fue desarrollado por MySQL AB, una empresa de origen sueco, aunque posteriormente fue adquirida por la empresa Oracle Corporation [21].

La primera versión de MySQL apareció en 1995, fue creada para un uso personal, y no fue hasta el año 2000 cuando la versión 3.23 pasó a tener licencia para el público general, licencia GPL (*General Public License*).

La versión 4 apareció en 2001, pero se declaró estable en 2003. Esta versión aportaba numerosas funcionalidades y mejoras con respecto a las anteriores, con especial énfasis en la gestión de varias tablas a la vez, asignación de privilegios e introducción de mejoras notables en el rendimiento.

Ese mismo año se publicó la versión 5, que de manera análoga no se declaró estable hasta el año 2005. Esta versión introdujo numerosas características que faltaban en MySQL, por lo que fue probablemente la versión mejor recibida y más destacada.

Desde 2013 se trabaja con la versión 5.6, la que se ha utilizado para la realización del presente trabajo.

2.2.4 PHP

2.2.4.1 Presentación

PHP [22] con el paso de los años se ha convertido en uno de los lenguajes más utilizados para la creación de páginas web dinámicas. Históricamente el nombre PHP viene de *Personal Home Page*, o Página de Inicio Personal, aunque ahora es conocido como *PHP Hypertext Preprocessor*, o Preprocesador de Hipertexto PHP. Varias razones han hecho de PHP un lenguaje tan popular:

- PHP es muy fácil de aprender con respecto a otros lenguajes utilizados para el mismo propósito como pueden ser Java o ASP (*Active Server Pages*). Debido a esto no es necesario hacer un estudio muy estricto de sus funciones para realizar programas sencillos que nos resuelvan gran cantidad de problemas diarios.

Figura 2-7 Logo de PHP [22]

- Las páginas escritas en PHP son simples páginas HTML, que contienen el programa que queremos ejecutar, a mayores de las etiquetas normales. Cuanto un cliente solicita la página, el servidor pre-procesa los datos y ejecuta las instrucciones de PHP. El resultado de la ejecución, usualmente una página web codificada en HTML, se le envía al cliente. El cliente no recibe nada de código PHP.
- Otra de las grandes ventajas de PHP es su capacidad de ejecutarse en multitud de plataformas, sistemas operativos y servidores existentes. Es compatible con los tres servidores líderes en el mercado (Apache, *Microsoft Internet Information Server* y Nginx) así como con los principales sistemas operativos (Windows, Mac y Linux).
- Desde el punto de vista económico tiene la ventaja de que PHP tiene una Licencia de Código Abierto (*Open Source*) que implica que el código fuente de PHP es libre de ser descargado e inspeccionado por cualquiera. Además, el coste del producto es cero.

Todas estas características han hecho que el uso de PHP se haya disparado desde el año 1999 y que sea uno de los lenguajes de desarrollo más utilizados en Internet según el índice TIOBE [23].

Por todas las razones anteriores, es por lo que se ha decidido utilizar este lenguaje a la hora de desarrollar nuestra aplicación web.

2.2.4.2 Historia

El lenguaje PHP [24], fue creado en el año 1994 por Rasmus Lerdorf. Su idea no era más que crear una herramienta para organizar y simplificar sus proyectos de desarrollo web personales, pero siguió trabajando en esta herramienta y la publicó en 1995.

Entrado ese mismo año, se publicó una versión totalmente reescrita con el nombre de PHP versión 2. Esta versión era capaz de gestionar formularios y acceder a bases de datos SQL.

En el año 1997, un equipo dirigido por Rasmus Lerdorf se hizo cargo del desarrollo del lenguaje, dando como resultado la salida de la versión 3. En el año 2000 cambiaron de motor de análisis para ofrecer un mayor rendimiento y soportar mayor número de extensiones: esta fue la versión 4.

La versión 5 nació en 2004, esta versión aportó grandes novedades, como la gestión de errores. La posterior versión 6 no tuvo éxito y no es utilizada en la actualidad. Recientemente, en noviembre de 2015, ha salido a la luz la versión 7, pero aún su uso no está extendido, por lo que hoy en día se utiliza la versión 5 en alguna de sus versiones. Normalmente 5.5, 5.6 ó 5.7. En este trabajo se ha trabajado con la versión 5.5.9.

3 DESARROLLO DEL TFG

3.1 Definición del problema a resolver y decisiones iniciales

Una vez establecidos los objetivos del trabajo, es necesario plantearse el proceso de desarrollo. Para la realización del trabajo se considera la necesidad de una base de datos que permita recoger toda la información necesaria para el control de asistencia, y a la vez permita una manera sencilla de obtener estadísticas. También se considera necesario lograr una interfaz sencilla para el usuario, con el fin de que introducir novedades sea lo más sencillo posible y no sea fuente de errores. Esta interfaz también debe garantizar que la carga de novedades no suponga un trabajo excesivo para profesores o jefes de grupo, y especialmente, que no implique una pérdida del tiempo de clase.

Uno de los principales problemas para el almacenamiento en la base de datos es que no se puede almacenar toda la información en una gran tabla. Esto se debe a que las relaciones que unen a los alumnos, profesores, grupos y asignaturas no son únicas. Es decir, un alumno está matriculado en varias asignaturas, y en una asignatura hay matriculados varios alumnos. De forma análoga, un profesor imparte varias asignaturas y una asignatura puede ser impartida por más de un profesor. Estas relaciones son referidas como relaciones N a M o "N:M".

Para la gestión de relaciones N:M es necesario contar con tablas que se relacionen entre sí, estos conjuntos de tablas se producen en las llamadas Bases de Datos Relacionales, o BDR. Para trabajar con una BDR se decide utilizar MySQL pues permite almacenar grandes cantidades de datos en distintas tablas e interconectarlas entre sí, para obtener estadísticas más complejas. También permite cargar la información de las tablas de forma sencilla para así introducir los datos de alumnos, profesores, etc; y trabajar con estos datos.

Por otro lado, cada día hay que introducir varias novedades cada hora de clase, y el error a la hora de introducir dicha información puede causar malinterpretaciones en la base de datos, y por tanto la generación de estadísticas erróneas. Esto puede incluso tener consecuencias negativas en los alumnos, por lo que es imprescindible idear algún sistema que evite estos errores.

Se valora que para que no se produzcan errores asociados a una mala introducción de datos, esta carga de datos debe hacerse en una interfaz sencilla y autoexplicativa que no requiera formación previa en la manera de introducir correctamente la novedad en la base de datos. Así mismo, una interfaz sencilla permitirá que el proceso de inserción de novedades sea lo más ágil posible, sin renunciar a la velocidad que supone el parte de clase actual, ni sacrificar minutos de clase.

Con el fin de lograr esta simplicidad, se opta por la programación web con HTML y PHP. El conjunto de ambas herramientas, PHP para programar acciones y HTML para la presentación en el navegador, permite generar una interfaz sencilla que cumple con todas las funciones propuestas. La principal ventaja de crear una aplicación web es la posibilidad de acceder desde cualquier dispositivo

conectado a internet, desde el navegador, sin necesidad de tener que descargar o instalar nada. Además, el uso del navegador es muy común hoy en día, por lo que reduce la necesidad de formación previa. PHP se complementa con HTML, como se explicó en el apartado 2.2.4, además de estar diseñado para compenetrarse con MySQL de la forma más sencilla posible. La carga y descarga de datos de MySQL se puede hacer con líneas de código PHP. Esto permite que la propia página web realice la introducción de datos en la BDR mediante el uso de formulacios, sin necesidad de que el usuario conozca la sintaxis de introducción de datos en MySQL.

Otro de los matices al tener en cuenta para preparar la aplicación fue el sistema de introducción de datos en la página web. El objetivo es idea un sistema que resulte lo más ágil posible, puesto la introducción de datos con el teclado no se considera suficientemente práctica y veloz.

En un primer lugar, se planteó la opción de adquirir tarjetas RFID (*Radio Frequence IDentifier*) codificadas [25] y distribuirlas entre alumnos y profesores para utilizarlas como identificación. Esto requeriría también un lector de tarjetas instalado en cada aula pero tiene dos inconvenientes. El primero es que las clases en la ENM pueden llegar a ser de 80 alumnos, incluso en horas de instrucción militar se junta todo el batallón para una misma actividad. Esto implicaría que todos los alumnos tienen que pasar su tarjeta por el lector antes de la actividad, suponiendo una considerable pérdida de tiempo. El segundo inconveniente es la dificultad de adaptar el sistema a los dispositivos móviles, requisito a medio plazo, para que el sistema pueda ser incorporado en las clases prácticas, como embarques o prácticas de campo.

Por estos motivos se toma la decisión de basar la introducción de datos en códigos de barras. Esto permite que, con la adquisición de sencillos lectores de códigos de barras, se pueda introducir de manera cómoda los datos en cualquier ordenador. Además, facilita mucho la progresión hacia los dispositivos móviles, ya existen hoy programas que interpretan códigos de barras con la cámara de teléfonos inteligentes. De esta manera, el profesor puede introducir las novedades de forma sencilla con tan solo llevar su teléfono móvil a la lancha de instrucción o a las prácticas de campo.

Este apartado describirá el trabajo realizado por el alumno, metodologías empleadas, problemas presentados, soluciones propuestas, etc.

3.2 Configuración del entorno de trabajo y preparación de las herramientas

El siguiente paso es configurar un entorno que permita trabajar con las herramientas explicadas en los apartados anteriores. En primer lugar se decide trabajar en una máquina virtual que opere sobre el sistema operativo Ubuntu [26].

Figura 3-1 Logo de Ubuntu [26]

Una máquina virtual es un software que simula un ordenador, accediendo a los recursos del ordenador anfitrión, pero siendo solo una aplicación dentro del sistema operativo original. Esto permite trabajar con Ubuntu en un ordenador que originalmente sua Windows o Macintosh como sistema operativo [27].

Ubuntu [26] es un sistema operativo basado en Linux, enfocado a usuarios principiantes e intermedios, con el objetivo de resultar sencillo para el usuario. La alta capacidad de configuración de Ubuntu, y la facilidad de uso de su terminal, hacen de este sistema operativo el ideal para este trabajo, especialmente para la gestión de MySQL.

Para la instalación de la máquina virtual se utiliza el software de la empresa Oracle [21] llamado VirtualBox [28].

Este programa es un gestor de máquinas virtuales desarrollado con una licencia GPL que funciona en, y soporta, la gran mayoría de sistemas operativos en el mercado. El programa se descarga de forma gratuita en su página web [28].

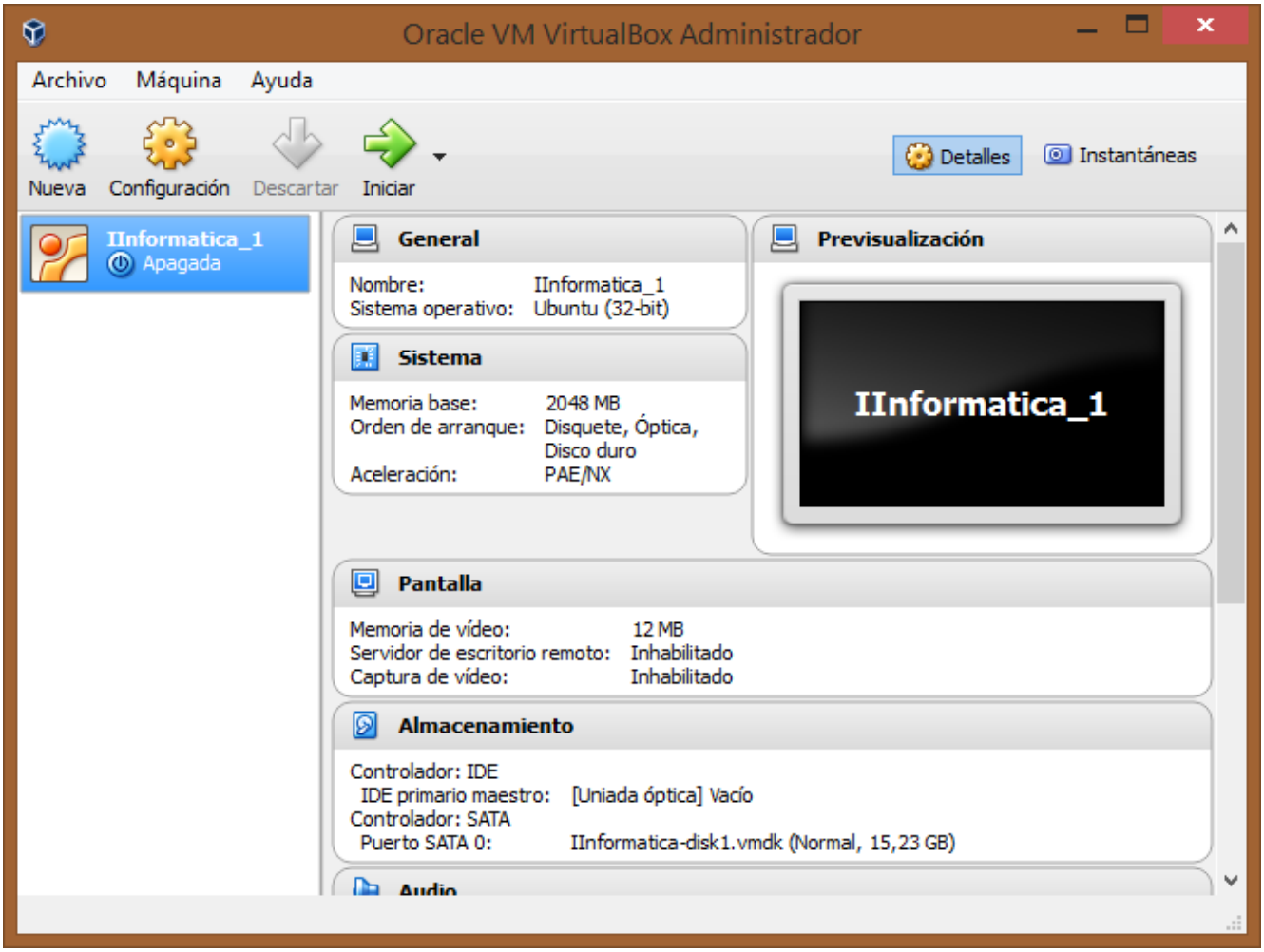


Figura 3-2 Pantalla de inicio de la herramienta VirtualBox

La máquina virtual utilizada con la versión 14.04.4 de Ubuntu es proporcionada por el Centro Universitario de la Defensa, CUD [2]. Se utiliza la máquina empleada para impartir la asignatura de Informática, de forma que el usuario tiene permisos de superadministrador, o superusuario (en Ubuntu, "*root*") a través del comando "*sudo*".

Una vez cargada la máquina virtual, es necesario comprobar que la versión de Ubuntu se encuentra correctamente actualizada antes de empezar a trabajar con ella. Para ello se introduce en el terminal el comando:

- *sudo apt-get update*

A continuación se procede con la instalación del servidor LAMP. Recordamos que LAMP proviene de las siglas de Linux, Apache, MySQL y PHP. Ya se cuenta con el sistema operativo Ubuntu Linux. El servidor LAMP incluye las últimas versiones de las tres herramientas restantes, y con un sencillo comando en el terminal se puede instalar y actualizar Apache, MySQL y PHP:

- *sudo apt-get install lamp-server^*

Una vez se llega a este punto se está en condiciones de empezar a trabajar. En primer lugar hay que crear la BDR en MySQL y dar permisos a un usuario para trabajar sobre la base de datos. Se accede a MySQL con el comando:

- *mysql -u root -p*

A continuación hay que crear la BDR. En este trabajo se ha decidido llamar a la BDR "trabajo". Es importante saber que MySQL es sensible a mayúsculas y minúsculas en los nombres, aunque no en los comandos. Para crear la base de datos se introduce:

- *CREATE DATABASE trabajo;*

El siguiente paso es crear un usuario que pueda trabajar sobre la base de datos "trabajo", pues trabajar con permisos de "*root"* no se considera buena práctica de seguridad. Se crea el usuario "Astorga", con una contraseña que contiene mayúsculas, minúsculas y números, por requisito del sistema, y se le conceden derechos de trabajar libremente en la BDR "trabajo" con los siguientes comandos;

- *create user "Astorga"@"localhost" identified by "xxxxxxxx";*
- *grant all on trabajo.* to "Astorga"@"localhost";*

Por último, antes de empezar con la programación comprobamos que tanto Apache como PHP están correctamente instalados y operativos. Para ello abrimos el navegador e introducimos la dirección:

- *localhost/index.html*

Si Apache funciona correctamente, nuestro navegador mostrará la siguiente imagen:

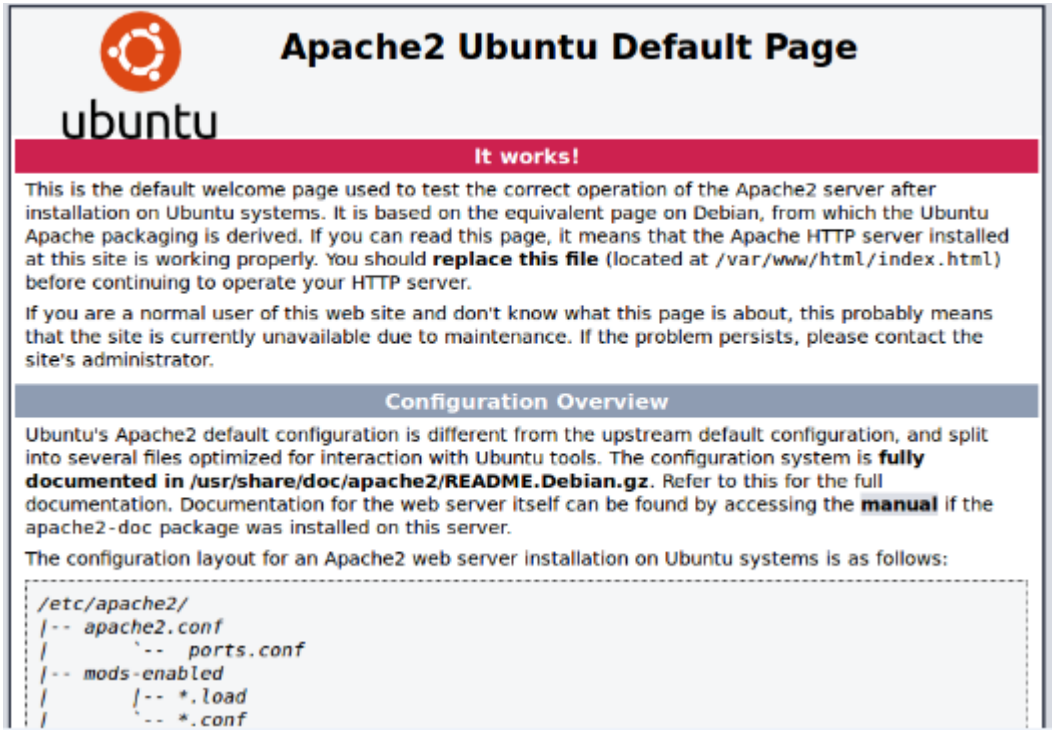
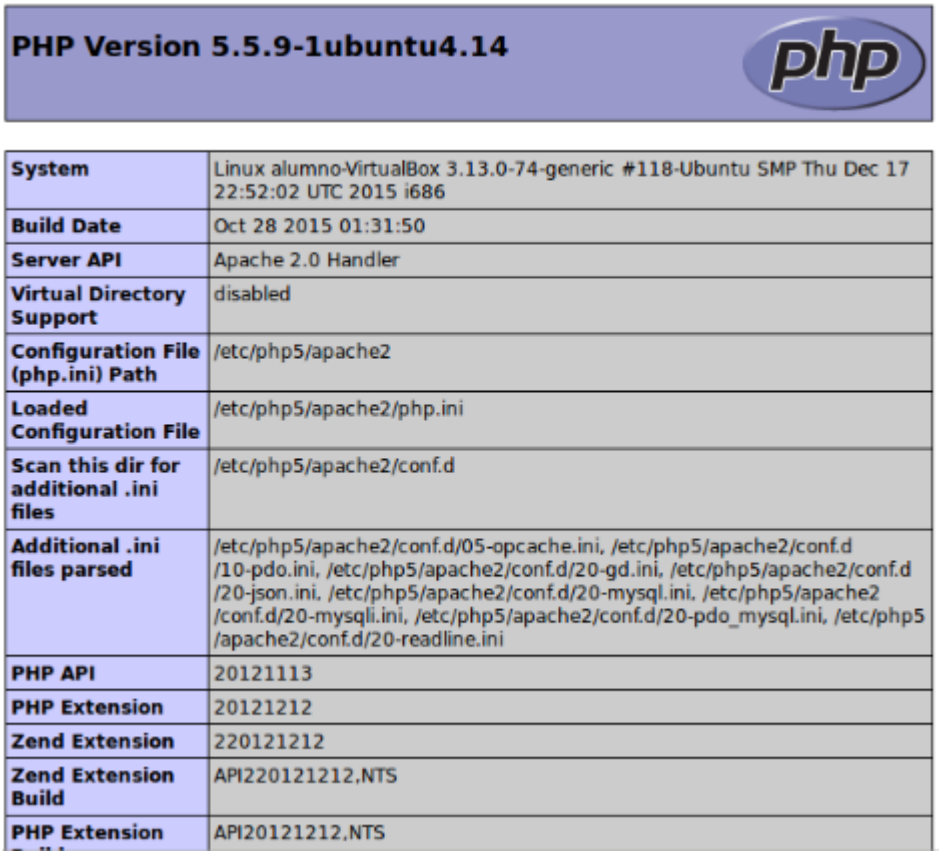


Figura 3-3 Página automática de confirmación del funcionamiento de Apache

A continuación hacemos el equivalente con PHP. Para ello se genera un archivo PHP con la línea "*phpinfo();*", y se guarda como *test.php* a continuación accedemos en nuestro navegador a:

- *localhost/test.php*

Con lo que la pantalla mostrará la tabla a continuación:



PHP Version 5.5.9-1ubuntu4.14

System	Linux alumno-VirtualBox 3.13.0-74-generic #118-Ubuntu SMP Thu Dec 17 22:52:02 UTC 2015 i686
Build Date	Oct 28 2015 01:31:50
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-readline.ini
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API220121212,NTS
PHP Extension	API20121212,NTS

Figura 3-4 Página automática de confirmación del funcionamiento de PHP

Nos encontramos ahora en condiciones de empezar la programación de la base de datos y de la página web.

3.3 Programación MVC

MVC [29] [30] hace referencia a Modelo-Vista-Controlador. MVC es un patrón de diseño software que propone separar el código de un programa en función de sus responsabilidades. El diseño MVC se utiliza para sistemas que requieren interacción con el usuario.

La razón de ser del diseño MVC es la generación de un código de calidad que permita un sencillo mantenimiento, separando conceptos y que facilite la optimización del código mediante la reutilización de líneas de código. Se fundamenta sobre tres pilares: modelo, vista y controlador. Cada pilar se acota en función de su responsabilidad en la ejecución del programa.

El modelo es la capa que trabaja con los datos, en ella se encuentran todas las funciones que ejecuta el programa. Estas funciones acceden a la información y la actualizan, borrando e introduciendo en las tablas. En el modelo se encontrarán todas las acciones que acceden a MySQL, leyendo datos, guardando datos e incluso eliminando las tablas.

Las vistas contienen el código del programa que se presenta en pantalla. Las vistas producen la visualización de las interfaces de usuario. En ellas se encuentra el código HTML que interpreta el navegador, y con el que interactúa el usuario. Las vistas requieren los datos al modelo, y el código PHP que contienen se utiliza para mostrar las salidas de las funciones del modelo.

Los controladores responden a las acciones que se solicitan en la aplicación, como puede ser visualizar un elemento, realizar una consulta de información, introducir datos, etc. La capa de control sirve de enlace entre vistas y modelo, respondiendo a los mecanismos que se requieran en la aplicación. Sin embargo no manipula directamente datos ni muestra ningún tipo de salida.

En la Figura 3-5, podemos ver un esquema de cómo es la interacción modelo-vista-control-usuario.

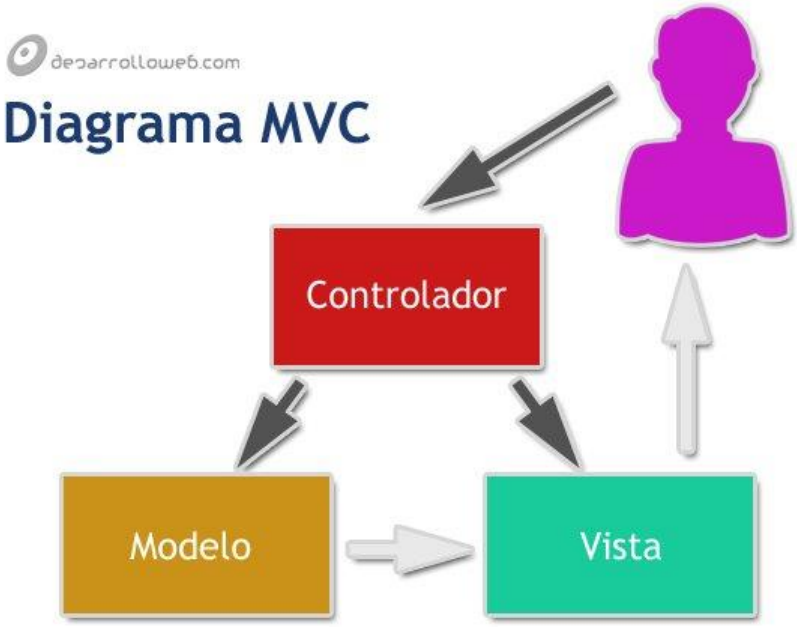


Figura 3-5 Esquema del modelo MVC [29]

Las necesidades de la Escuela Naval pueden evolucionar con el tiempo, por lo que puede llegar a ser necesario modificar el programa. La ventaja de una arquitectura MVC es la sencillez que supone para futuras modificaciones del código, esta arquitectura no solo existe para crear aplicaciones de mayor calidad, sino que además es mucho más fácil de entender el código para un programador que lo lea por primera vez.

Por otro lado, es probable que al manipular el programa se realice una misma acción varias veces, como puede ser nombrar a los alumnos: se nombran una vez para introducir la novedad del grupo de clase y otra vez para mostrar dicha novedad. Sin la arquitectura MVC, habría que escribir dos veces la secuencia de código que muestra los alumnos exactamente igual. MVC permite generar una función que nombre los alumnos, y simplemente llamarla cada vez que se requiera esa acción, reutilizando así el código.

3.4 Estructura de la base de datos y de las vistas de la aplicación

A la hora de programar una aplicación como la del presente trabajo, tiene especial relevancia tener claro qué es lo que se quiere conseguir, y qué elementos son necesarios para lograrlo.

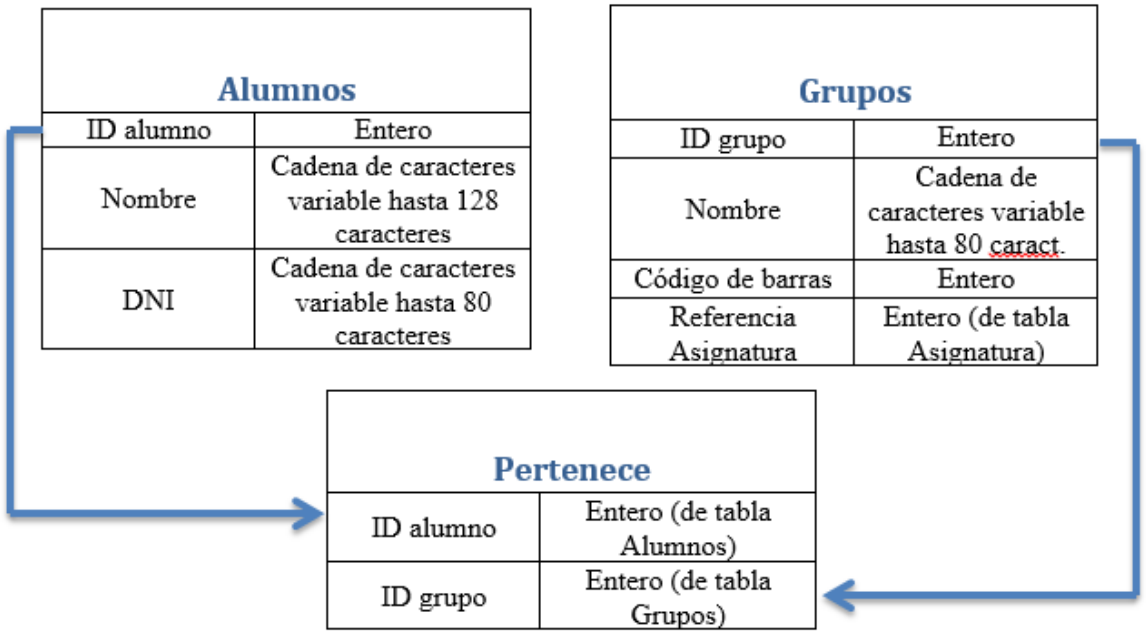
La BDR es la base de la aplicación, por lo que si su diseño tiene fallos o faltan datos, la aplicación nunca podrá tener todas las funcionalidades que se pretenden. Lo mismo sucede con la programación web, es necesario conocer las variables de las que se dispone y las que se quiere obtener antes de programar y darse cuenta que faltan datos cuando es demasiado tarde.

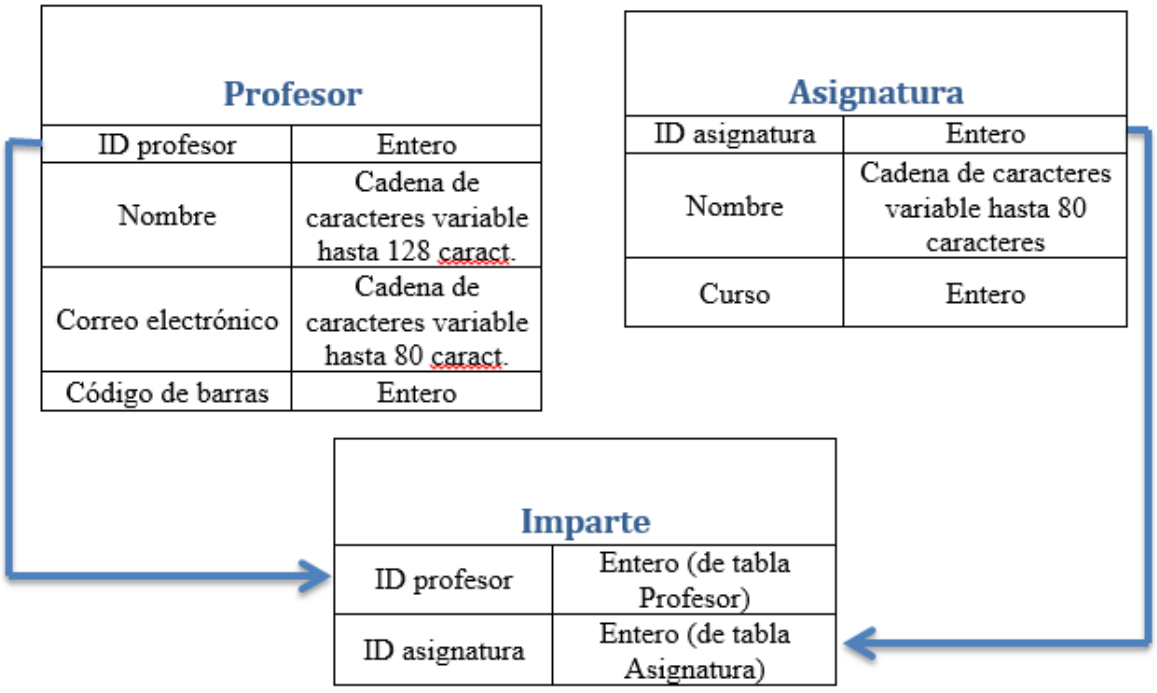
Por ello se realiza una tormenta de ideas en la que se considera todos los datos que se necesitarán y los resultados que habrá que obtener. A continuación se diseña un esquema de tablas que contenga todos los datos necesarios para recoger información, y tablas con los campos adecuados para volcar las novedades. Además la programación en MySQL requiere que para cada campo se defina el tipo de variable que se va a guardar.

Tras una reunión con personal de Jefatura de Estudios de la ENM, se averiguan los datos de los que disponemos antes de empezar a programar, así como las expectativas que tiene la Escuela para la generación de estadísticas. La Escuela puede proporcionar información de alumnos, profesores,

asignaturas, grupos de clase y motivos de faltas. A cambio esperan estadísticas que permitan filtrar las faltas por alumno, por profesor y por asignatura.

Con toda esta información se decide crear las tablas de la Figura 3-6.





Motivo	
ID motivo	Entero
Nombre	Cadena de caracteres variable hasta 80 caracteres

Introducidos	
ID	Entero con incremento automático
ID grupo	Entero (de tabla Grupos)

Novedades	
ID alumno	Entero (de tabla Alumnos)
ID grupo	Entero (de tabla Grupos)
ID profesor	Entero (de tabla Profesor)
ID motivo	Entero (de tabla Motivo)
Creación	Grupo fecha/hora
Hora	Entero
Doble	Booleano sí o no

Guardar	
Nombre alumno	Cadena de caracteres variable hasta 128 caract.
Nombre grupo	Cadena de caracteres variable hasta 128 caract.
Nombre profesor	Cadena de caracteres variable hasta 128 caract.
Nombre motivo	Cadena de caracteres variable hasta 128 caract.
Creación	Grupo fecha/hora
Hora	Entero
Doble	Booleano sí o no

Figura 3-6 Tablas ideadas para la creación de la BDR

La tabla Pertenece, relaciona la permanencia de cada alumno a distintos grupos. De forma similar, la tabla Imparte, relaciona a cada profesor con las asignaturas que imparte.

Es importante destacar que el concepto de Grupo, difiere del concepto de grupo de clase de la ENM explicado con anterioridad. Se considerará un grupo la unidad básica de alumnos que van siempre juntos a todas las asignaturas, y existirá un grupo por asignatura. Es decir, para cada asignatura hay tantos grupos como corresponda en la brigada, más grupos, que pueden ser unipersonales, de alumnos que hayan repetido curso o hayan convalidado asignaturas y tengan una sábana diferente.

Además se idean las tablas de Novedades y Guardar. En la primera tabla se guardan los identificadores de la novedad de cada día, con un apartado para el grupo fecha hora (terminología militar que refiere el instante y la fecha en la que se realiza una acción) en la que se introdujo la

novedad, un apartado que guarda la hora correspondiente al horario normal de la ENM (1ª hora, 2ª hora, etc) y un último apartado de valor "verdadero" o "falso" para registrar las horas prácticas, que suponen dos horas de clase seguidas. Este último apartado evita que el profesor tenga que interrumpir la clase para volver a realizar la introducción de la novedad. La segunda tabla traduce la primera tabla de identificadores a una tabla equivalente de nombres, más sencilla para interpretar en Jefatura de Estudios.

También se genera la tabla Introducidos, que no es más que una tabla temporal que almacena durante el uso de la aplicación los grupos cuya novedad ha sido introducida.

Por otro lado, se realiza un primer borrador de la estructura de las distintas vistas de la aplicación web, la idea inicial de la página se incluye en la Figura 3-7, Figura 3-8 y Figura 3-9. Al abrir la aplicación se mostraría la pantalla de índice, una vez el profesor introduce su código de acceso le redirige a otra página en la que escanea los códigos de los grupos, para a continuación seleccionar la novedad de cada alumno.

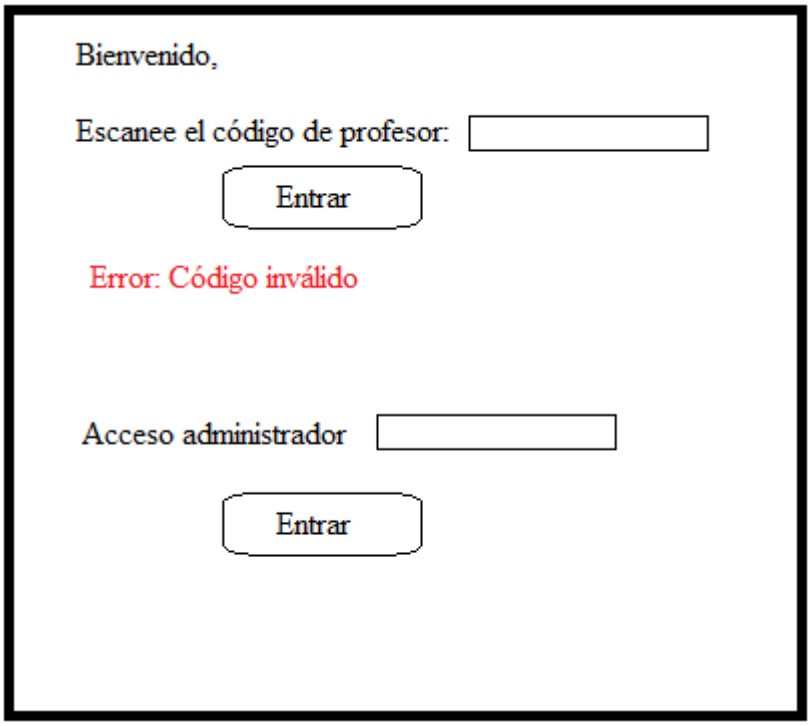


Figura 3-7 Idea de la presentación de inicio

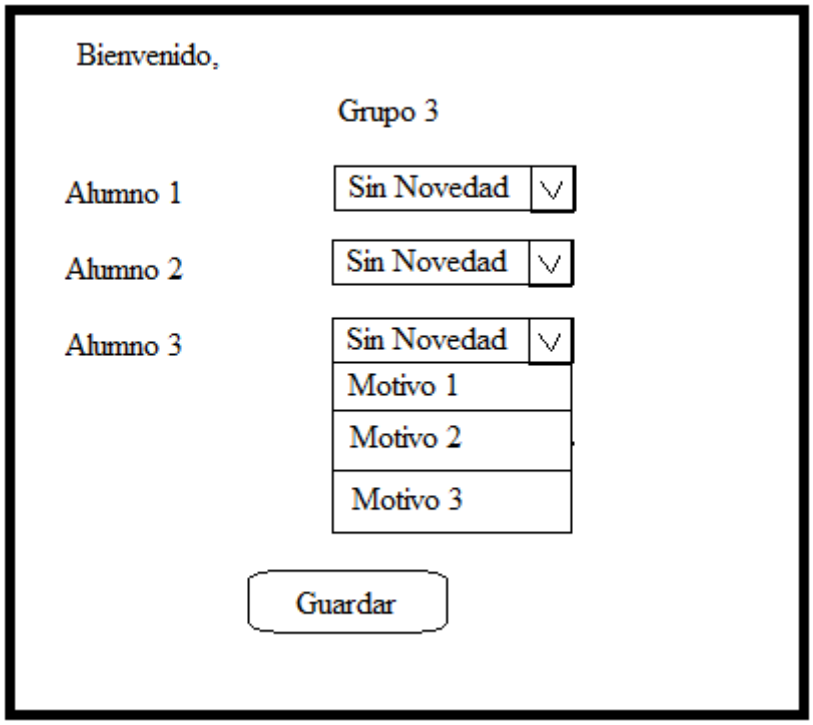


Figura 3-9 Idea de la presentación para introducir novedades

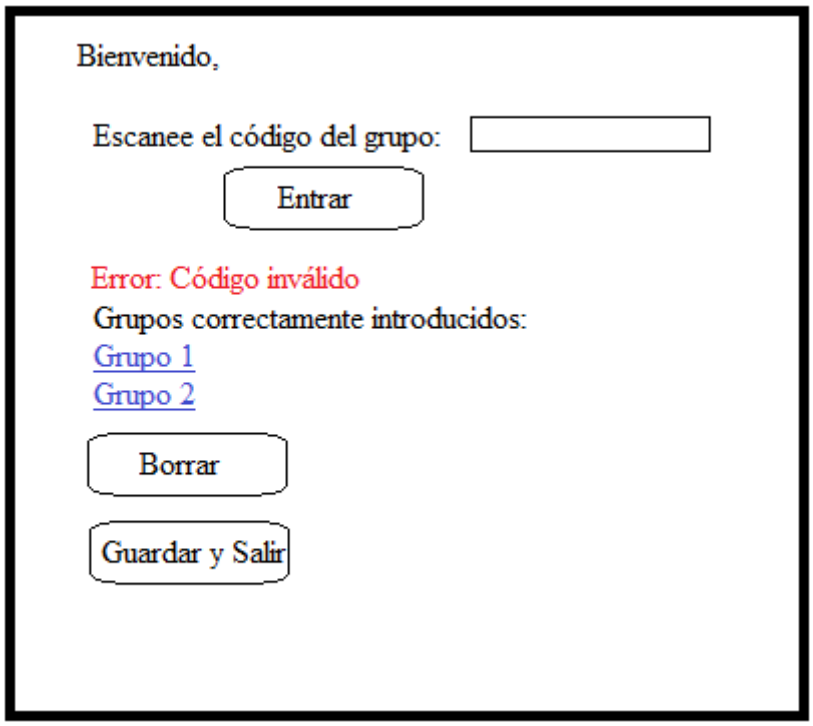


Figura 3-8 Idea de la vista principal de la aplicación

Cada vez que se introduce una novedad, se vuelve a la página de introducción de grupos, la cual permite introducir un nuevo grupo, ver las novedades introducidas hasta el momento, borrar alguna novedad o guardar todo y salir. Por otro lado, el administrador puede acceder con su código, desde la pantalla de inicio, para descargar o borrar la base de datos de novedades.

Una vez validada la idea inicial, se procede a su escritura, empezando por la base de datos.

3.5 Programación de la base de datos con MySQL

Una vez diseñada sobre el papel la base de datos, se procede a crear las tablas necesarias en MySQL. Es importante crear las tablas con los campos adecuados para no provocar errores a la hora de trabajar con los datos, en caso de haber creado mal una tabla, es posible borrar o editar cualquiera de sus campos.

En primer lugar se expondrán las definiciones necesarias para entender los campos de cada tabla. Solo se definen los campos que han sido utilizados para la creación de alguna tabla [18]:

- *int*: Número entero positivo o negativo de 24 bits (de -2^{23} a $+2^{23}-1$).
- *varchar (n)*: Cadena de longitud variable, de n caracteres como máximo.
- *Timestamp*: Guarda la fecha y hora del sistema en el momento de la introducción de los datos.
- *Boolean*: Variable que sólo puede tomar dos valores: Verdadero o Falso. Representa un valor verdadero con un 1, y uno falso con un 0.
- *Primary key*: Una clave primaria garantiza que no habrá dos filas en la tabla con el mismo valor en las columnas que componen la clave. Además no permite que una columna con clave primaria quede vacía. Sólo se permite una clave primaria por tabla.
- *Foreign key*: Una clave externa garantiza que los datos añadidos en la columna de la clave existen previamente en otra tabla, la columna de la tabla en la que debe existir la clave externa se especifica mediante el comando *References*.
- *Auto_increment*: establece que cada vez que se introduzca un dato en la tabla, el valor en cuestión se incrementará en una unidad. Así la tabla queda de forma muy sencilla ordenada de forma cronológica de introducción.
- *Engine=InnoDB*: InnoDB es un mecanismo de almacenamiento que no viene instalado por defecto, pero que es necesario para habilitar la función de *foreign key*.
- *Default charset=utf8*: Define el mecanismo de codificación de caracteres por defecto. UTF8 permite el alfabeto occidental añadiendo tildes y la letra eñe con lo que resulta válido para el castellano. Es importante introducir el alfabeto adecuado para que no se produzcan errores al introducir apellidos o nombres españoles.

Se abre el terminal de Ubuntu, y se carga MySQL. Tras acceder con la cuenta de usuario "Astorga" se selecciona la base de datos "trabajo" y se procede a la creación de las tablas expuestas en el la **¡Error! No se encuentra el origen de la referencia.**, Figura 3-8 y Figura 3-9. También se incluye una tabla más, "Administrador", que aunque no estaba contemplada en el diseño inicial, se considera necesaria al avanzar en el desarrollo del programa, para permitir identificarse como administrador del sistema.

En el Anexo I: Comandos a introducir para crear la base de datos, se recoge la secuencia introducida. En caso de que el código no se introduzca correctamente, la tabla no será creada. El comando "*show tables;*" muestra las tablas que forman parte de la base de datos. En este caso obtenemos las tablas de la Figura 3-10.

```
Database changed
mysql> show tables;
+--------------------+
| Tables_in_trabajo  |
+--------------------+
| Administrador      |
| Alumnos            |
| Asignatura         |
| Grupos             |
| Guardar            |
| Imparte            |
| Introducidos       |
| Motivo             |
| Novedades          |
| Pertenece          |
| Profesor           |
+--------------------+
11 rows in set (0.00 sec)
```

Figura 3-10 Presentación de MySQL de las tablas utilizadas

A continuación se comprueba que los valores han sido bien introducidos mediante el comando "*describe*". En la Figura 3-11 se muestra el resultado de ejecutar el comando "*describe Novedades;*":

```
mysql> describe Novedades;
+------------+------------+------+-----+-------------------+-----------------------------+
| Field      | Type       | Null | Key | Default           | Extra                       |
+------------+------------+------+-----+-------------------+-----------------------------+
| idalumno   | int(11)    | YES  | MUL | NULL              |                             |
| idgrupo    | int(11)    | YES  | MUL | NULL              |                             |
| idprofesor | int(11)    | YES  | MUL | NULL              |                             |
| idmotivo   | int(11)    | YES  | MUL | NULL              |                             |
| creado     | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| hora       | int(11)    | YES  |     | NULL              |                             |
| doble      | tinyint(1) | YES  |     | NULL              |                             |
+------------+------------+------+-----+-------------------+-----------------------------+
7 rows in set (0.01 sec)
```

Figura 3-11 Descripción de la tabla Novedades

Si se han creado las tablas de forma correcta, se está en situación de empezar a introducir datos en las tablas, recorrerlas para obtener información o borrar datos mediante los distintos comandos de control de MySQL [18].

En primer lugar será necesario introducir datos en las tablas, lo que se realiza con el comando "*Insert Into*". Este comando permite insertar datos en una tabla, como en el ejemplo:

- *Insert Into Alumnos Values (1234, "Alumno de Ejemplo", "12345678A");*

También permite introducir datos sólo en algunas columnas, sin olvidar que siempre ha de estar introducida la columna que sea clave primaria. Suponiendo que no se conoce el DNI del alumno de ejemplo anterior, se podría ejecutar:

- *Insert Into Alumnos(idalumno, nombre) Values (1234, "Alumno de Ejemplo");*

En la columna de DNI se almacenará un valor nulo, o *NULL*.

El caso contrario puede ser querer borrar los valores de una tabla, lo que se haría con el comando "*Delete*". En caso de querer borrar todos los datos de una tabla, se introduciría la orden:

- *Delete from Alumnos;*

De forma análoga se puede seleccionar los campos a borrar, añadiéndolos a continuación de la palabra *Delete*. Sin embargo si no se quiere borrar todos los datos de una columna, sino un dato en concreto hay que emplear el comando "*Where*". "*Where*" permite seleccionar una característica concreta del campo a seleccionar. Suponiendo que se quiere borrar el DNI de un alumno, cuyo identificador es conocido:

- *Delete DNI from Alumnos Where idalumno=1234;*

La última funcionalidad principal de MySQL para la gestión de los datos en las tablas es el comando "*Select*". Este comando permite seleccionar información concreta de una tabla y devolverla al programa. Se puede desde seleccionar toda la información de una tabla, hasta un único elemento mediante el comando "*Where*". En los ejemplos a continuación seleccionamos todos los elementos de Novedades, después sólo los identificadores de profesores, y por último sólo la novedad de un alumno.

- *Select * from Novedades;*
- *Select idprofesor from Novedades;*
- *Select * from Novedades where idalumno=1234;*

En este punto se cuenta con la base de datos preparada y con las herramientas necesarias para trabajar con ella. El siguiente paso es preparar la interfaz con la interactuará el usuario, es decir, las distintas páginas web de las que constará la aplicación.

3.6 Presentación web con HTML

3.6.1 Utilidad del lenguaje HTML

Una vez la BDR está configurada correctamente, hay que centrarse en la interfaz con la que va a interactuar el usuario del programa. La presentación en páginas web, como se explica en el punto 2.2.1 corre a cargo de las etiquetas HTML. Una página HTML puede contar con gran número de etiquetas, estas permiten llevar a cabo las funciones necesarias para la correcta presentación del programa por pantalla. El lenguaje HTML permite, entre otras cosas, organizar el texto, crear tablas, añadir efectos visuales o crear formularios para insertar texto. En este trabajo se han utilizado las etiquetas necesarias para hacer la aplicación ordenada y sencilla, y estas se detallan a continuación.

3.6.2 Etiquetas HTML

Toda página HTML [7] se inicia con la etiqueta *<DOCTYPE! html>*, que define el tipo de documento como HTML. A continuación le siguen las etiquetas que dividen la cabecera (*head*) del documento, y el cuerpo (*body*). La cabecera contiene información relevante para cargar e interpretar correctamente la página. Contiene el título de la página, la definición de los estilos y el alfabeto aceptado. Por otro lado, el cuerpo define el contenido visible por el usuario.

En HTML se recogen las etiquetas en distintos grupos, teniendo en cuenta la función que desempeñan. En este trabajo se han utilizado cuatro grupos principales.

3.6.2.1 Etiquetas de definición del documento

Las siguientes etiquetas definen el documento, sin embargo ninguna de la información que contienen es percibida por el usuario.

- *<!DOCTYPE html>:* Establece el tipo de documento como HTML.
- *<html> </html>:* Encuadra todo el código para que las etiquetas se interpreten como HTML.
- *<head> </head>:* Cabecera del documento.
- *<title> </title>:* Establece el título de la página y se suele mostrar en la barra del navegador.
- *<body> </body>:* Encuadra la parte visible de la página.
- *<link>:* Abre de manera automática una página para cargar datos, se ha usado para cargar la hoja de estilos, que se verá más adelante.
- *<meta charset>:* Establece el alfabeto aceptado, en nuestro caso el occidental con tildes y la letra eñe, propias del idioma español.

3.6.2.2 Etiquetas de texto

A continuación se muestran las etiquetas utilizadas para la presentación de texto por pantalla. En la Figura 3-12 se puede ver un ejemplo de cómo representa el navegador un código muy sencillo:

- *<h1> </h1>:* Título de texto.
- * :* Párrafo de texto.
- *
:* Salto de línea.

```
<body>
<h1>Esto es un título de texto</h1>
<p>Esto es un párrafo de texto en el cual<br> se ha introducido un salto de línea</p>
</body>
</html>
```

Figura 3-12 Ejemplo de etiquetas de texto HTML y resultado

3.6.2.3 Etiquetas de interacción con el usuario

Estas etiquetas, permiten al usuario interactuar con el programa para realizar consultas o introducir datos. Vemos un sencillo ejemplo de cada campo en la Figura 3-13.

- *<a href> :* Crea un enlace a otra ventana.
- *<form> </form>:* Abre un formulario.
- *<input>:* Elementos de un formulario, puede ser de varios tipos:
  - *<input type="text">:* Permite introducir en un campo de texto.
  - *<input type="submit">:* Envía los datos del formulario.

- *<input type="button">:* Botón que realiza una acción, que se puede especificar, al ser pulsado.
- *<input type="hidden">:* Al enviar el formulario se manda un campo de datos que no es visible por el usuario.
- *<select> </select>:* Presenta un desplegable con varias opciones predefinidas para elegir.
- *<option> </option>:* Define las opciones disponibles de un desplegable.

```
<body>
<a href="páginadeejemplo.com">Creamos un enlace</a> a una página de ejemplo. <br>
<form>
<input type="text" /><br>
<input type="submit" valuee="Submit" /><br>
<input type="button" value="Botón" /><br>
<select><option selected='selected'>Ejemplo</option></select>
<input type="hidden">
</form>
</body>
```

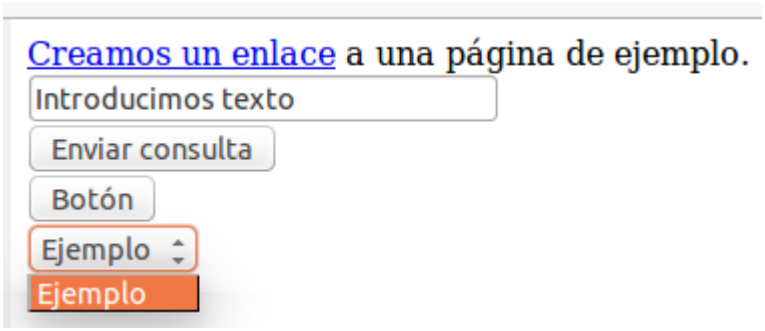


Figura 3-13 Ejemplo de etiquetas de interacción con HTML y resultado

3.6.2.4 Etiquetas para crear tablas

Por último se utilizan etiquetas que crean tablas. Estas tablas permiten organizar la disposición de la página y se utilizan para conseguir una presentación de datos ordenada y lo más estructurada posible. Vemos una sencilla tabla en el ejemplo de la Figura 3-14.

- *<table></table>:* Se utilizan para crear una tabla.
- *<tr></tr>:* Abren y cierran respectivamente una fila de la tabla.
- *<td></td>:* Se utilizan dentro de cada fila para especificar una nueva columna.

```
<body>
<table>
<tr><td>Casilla 1</td><td>Casilla 2</td></tr>
<tr><td>Casilla 3</td><td>Casilla 4</td></tr>
</table>
</body>
```

Casilla 1	Casilla 2
Casilla 3	Casilla 4

Figura 3-14 Ejemplo de etiquetas de creación de tablas y resultado

3.6.3 CSS

También es destacable la existencia de la etiqueta <*style*>, pues permite dotar a toda la presentación de mayor atractivo visual. La etiqueta cambia los estilos de la página web, incorporando colores de fondo, cambiando la fuente, el tamaño o el color del texto, o permitiendo diferentes tamaños o formas dentro de las tablas.

Se pretende que la presentación web sea agradable para el usuario, para hacer del programa una herramienta atractiva, fomentando así su uso. Por esto, se introduce el concepto de CSS (*Cascading Style Sheets)* u Hojas de Estilo en Cascada. Un archivo CSS recoge en un solo documento, todas las etiquetas de estilos que se desea aplicar al diseño web.

Para ello se construye un archivo CSS, que reúne los estilos deseados para los distintos elementos HTML. Se puede equiparar a un archivo que recoge todas las etiquetas <*style*> que se utilizarían en el código. Para que todo nuestro código se vea afectado por la hoja CSS simplemente es necesario llamarla en la cabecera del documento, mediante la etiqueta <*link*>. De esta manera se logra homogeneizar la presentación de las diferentes vistas y reutilizar el código de presentación.

La hoja de estilos utilizada se puede encontrar íntegra en el Anexo II: Hoja de estilos .

3.6.4 Interacción del usuario con PHP a través de HTML

En la aplicación cobran especial importancia los formularios, pues permiten comunicarse con PHP mediante llamadas al control y, junto a los enlaces, son la pieza fundamental de interacción entre el usuario y el programa.

Se consideran la pieza básica de la aplicación, y su diseño es la función más importante de HTML en el programa. Conocer las opciones que ofrecen los formularios HTML y saber configurarlos correctamente es requisito imprescindible para el buen funcionamiento de la aplicación.

El objetivo de los formularios añadidos en el código es que realicen acciones concretas, haciendo la aplicación dinámica. Los distintos campos de introducción del formulario se definen con atributos que facilitan su utilización, como añadirles un nombre o que salgan marcados por defecto.

3.7 Programación de acciones con PHP

A diferencia de HTML, PHP no es un lenguaje de marcado, sino que es propiamente dicho un lenguaje de programación. El código PHP es invisible y por tanto pasa desapercibido totalmente para el cliente, sin embargo es imprescindible para que la página web sea dinámica e interactiva [18] [24].

3.7.1 Características principales del lenguaje PHP

Como se ha mencionado anteriormente, el código PHP se incluye en las páginas HTML delimitado por las etiquetas "<*?php*" y "*?*>". Dentro de estas etiquetas cada instrucción se termina con un punto y coma ";", en caso contrario se producirá un error que no permitirá ejecutar el código. Para introducir código HTML en una función PHP se utilizará la función "*echo*", como en el ejemplo:

- *echo" Insertar texto HTML aquí ";*

El programa diseñado se basa en tres herramientas básicas del lenguaje, que son las variables, las estructuras de control y las funciones.

3.7.2 Variables

En PHP se permite guardar datos en la memoria, asociándolos a códigos sencillos de texto. Estos códigos pueden representar una constante, sin embargo, en esta aplicación se ha trabajado siempre con variables. Una variable se nombra añadiendo el símbolo del dólar ($) delante del nombre y es un espacio en la memoria cuyo valor es legible y modificable por la aplicación.

Las variables se organizan en dos grupos, pueden ser escalares o compuestas. Las variables escalares permiten almacenar números o cadenas de caracteres, también se incluyen en este grupo las variables de tipo verdadero o falso, también llamadas "*Boolean*".

El otro tipo de variables que se ha utilizado en el código son las tablas. Una tabla es una lista de elementos ordenada en pares clave/valor y se define con el código *array*. Las claves suelen ser de tipo entero o cadena de caracteres. En función del tipo de clave se distinguen lostipos de tablas. Si la clave es numérica se habla de una tabla numérica, en caso de una clave cadena se considera una tabla asociativa. Además, el valor asociado a la clave puede llegar incluso a tener el formato de tabla o *array*, en tal caso, se considera una tabla multidimensional.

En este programa se utilizan variables de tipo tabla para almacenar la información de las tablas de la BDR y así operar con ella.

3.7.3 Estructuras de control

Las estructuras de control permiten que las acciones del programa no se realicen secuencialmente de manera automática, sino que se añaden condiciones requisito para la ejecución de una acción en concreto. También facilitan la ejecución de acciones en bucle como el recorrido de todos los campos de una tabla, entre otras funciones.

La más sencilla es la estructura *if, else* traducible como "si", "si no,". Permite añadir una condición necesaria para que se ejecute una acción. Si la condición no se cumple, se ejecutará la acción definida por la palabra *else*. El añadir *else* es opcional, si no se incluye, en caso de que no se cumpla condición requerida, simplemente no se hará nada.

La siguiente estructura de control utilizada es *while* o "mientras", que como indica el nombre, permite ejecutar en bucle una acción mientras la condición sea verdadera.

La otra estructura utilizada para ejecutar instrucciones de manera iterativa es *for*, que permite controlar las iteraciones de la acción mediante tres expresiones.

- *for (expresión1; expresión2; expresión3) { instrucciones; }*

La "expresión1" se ejecuta solo una vez al iniciarse el bucle. La "expresión2" se ejecuta al inicio de cada iteración y su resultado se interpreta como verdadero o falso, si el resultado se evalúa como verdadero, se ejecutan las "instrucciones", si el resultado se evalúa como falso, se interrumpe el bucle. La "expresión3" se ejecuta al final de cada iteración. En nuestra aplicación se utiliza el *for* de la forma más común, en la que "expresión1" inicia un contador, "expresión2" evalúa el valor del contador y "expresión3" incrementa el valor del contador.

3.7.4 Funciones

PHP incluye una amplia variedad de funciones incorporadas. Una función ejecuta acciones diseñadas de forma automática. Se puede encontrar la descripción de todas las funciones existentes en la página web de PHP [24].

Un ejemplo de función es *echo*, que imprime un texto por pantalla. La función *echo* funciona como se muestra en la Figura 3-15.

```php
<?php
$variable = 'cadena de texto que permite ser editada';
echo "Texto que puede incluir una $variable";
?>
```

Texto que puede incluir una cadena de texto que permite ser editada

Figura 3-15 Ejemplo de código y resultado de la función *echo*

Existen otras funciones prediseñadas que se utilizan en la programación de este trabajo. En primer lugar la función *isset*, que indica si una variable está o no definida. Se emplea también la función *unset*, que elimina una variable de la memoria. Por otro lado se utiliza la función *count*, que cuenta la cantidad de elementos guardados en una variable de tipo tabla.

Por otro lado, el lenguaje PHP permite al programador crear sus propias funciones. Las funciones normalmente son parte del modelo y permiten la realización de una acción concreta. En una función se puede entrar con parámetros y devolver un resultado, o simplemente ejecutar una acción de forma automática. Una función se define con *function {}* y realiza todas las acciones incluidas entre las llaves, como se muestra en a continuación. En la Figura 3-16 se utiliza una función para definir una variable, se puede apreciar que antes de definirla la variable está vacía.

```php
<?php
echo "antes de la funcion la variable es $variabledefinida <br>";
$variabledefinida = definirlavariable();
function definirlavariable($entrandoconvariable){
$variabledefinida = 1;
return $variabledefinida;
}
echo "ahora la funcion devuelve del valor $variabledefinida";
?>
```

antes de la funcion la variable es
ahora la funcion devuelve del valor 1

Figura 3-16 Ejemplo de código y resultado de una función

3.7.5 Sesiones

Además PHP permite almacenar datos en sesión, de forma que estén disponibles en las diferentes páginas de código por las que está navegando un mismo usuario, añadiendo simplemente dos líneas de código al inicio de cada documento:

- *session_name("TFG");*
- *session_start();*

Nótese que el nombre de la sesión ha de ser el mismo en todas las páginas de un mismo programa para que los datos estén disponibles.

Para guardar una variable en sesión, se utiliza el *array* asociativo $_SESSION[], en el cual se define el nombre bajo el que se quiere guardar una variable. También permite cargar en otra página el contenido almacenado en sesión de la siguiente manera:

- Almacenar una variable para ser usada más tarde:
  $_SESSION[“nombre”] = $variable;

- Recuperar una variable de la sesión:
  $variable = $_SESSION[“nombre”];

De forma similar, se puede utilizar el comando $_REQUEST para obtener los datos introducidos en un formulario. De esta manera, se logra establecer una interacción entre el usuario y las funciones PHP mediante la introducción de texto o uso de las distintas opciones del formulario. En el siguiente ejemplo se accede en una página B a la información introducida en un cuadro de texto de la página A. Nótese que para poder usar dicha información adecuadamente es necesario guardarla en una variable:

- Página A, en la que se introduce el texto:
  <form method=”post”>
  <input type=”text” name=”texto”>
  <input type=”submit”>
  </form>

- Página B, en la que se trabaja con el texto previamente introducido:
  $texto = $_REQUEST[“texto”];

Una vez se ha introducido el lenguaje PHP, conociendo sus principales funcionalidades y herramientas para la creación de páginas web, se procede a la explicación del programa realizado para este trabajo.

3.8 Definiendo la aplicación web

El programa se ha escrito teniendo en cuenta la arquitectura MVC descrita en el apartado 3.3. La aplicación cuenta con una página de inicio y cinco vistas adicionales. Todo el control de la arquitectura se realiza en una única página, mientras que el modelo cuenta con tres archivos, uno de los cuales incluye todas las funciones necesarias para el correcto funcionamiento del programa, mientras que los otros dos contienen la función necesaria para exportar las novedades a Excel.

En este apartado se expondrá la estructura del programa, seguido de una explicación más detallada de las páginas que lo conforman, señalando las principales dificultades que presentan para su programación.

3.8.1 Estructura del programa

Al acceder a la aplicación, se abre la ventana del navegador mostrando el índice, es decir, “index.php”. Desde esta página se puede acceder a la “vista1.php” o a la “vista5.php” en función del formulario que se decida cubrir.

La “vista1.php” en su primera visita sólo ofrece dos opciones, rellenar el formulario para avanzar a la “vista2.php.” o salir, de vuelta a “index.php”. En la “vista2.php” se introducirán los datos de cada grupo de clase, y se volverá a la “vista1.php”.

La segunda y siguientes veces que se acude a la “vista1.php” permite tres acciones nuevas. La primera de ella es pinchar un enlace que dirige a la “vista4.php”, donde se muestra la novedad completa del grupo seleccionado. La siguiente opción es pulsar un botón que dirige a la “vista3.php”, página en la que se permite borrar una de las novedades previamente introducidas. La última opción que aparece es un botón que permite almacenar las novedades en la base de datos y volver a “index.php”.

El programa cuenta también con la página “modelo.php” la cual contiene todas las funciones que requiere la ejecución de la lógica de negocio de la aplicación, inclyendo las funciones necesarias para operar con las tablas de la base de datos. Los archivos “convertir.php” y “convertirids.php” exportan las tablas “Guardar” y “Novedades” respectivamente, a formato Excel.

Por último, el código de “control.php” contiene las relaciones y enlaces entre el modelo y las diferentes vistas, completando así la arquitectura MVC.

3.8.2 Vistas de la aplicación

En este apartado se explicarán los datos más relevantes acerca de las distintas vistas de la aplicación.

3.8.2.1 Index.php

El índice, como regla general para todas las aplicaciones informáticas, se guarda en un archivo de nombre “*index*” (índice en inglés), en este caso con la extensión *.php*. Si se realiza una consulta al servidor de Apache en la que no se especifique otra página de la aplicación, por defecto buscará la página nombrada *index.php*. Se puede ver el código completo en el Anexo III: index.php.

Esta vista es muy sencilla y su función es realizar una presentación clara de la aplicación y permitir a un profesor o administrador entrar en la herramienta para actualizar el estado de las novedades. Para ello cuenta con dos formularios diferenciados, el primero pide al profesor que escanee su tarjeta de identificación o introduzca su código para acceder a la siguiente página, mientras que el segundo formulario realiza una acción similar pero para que se identifique el administrador y acceda a la vista 5. Vemos el resultado de la presentación del índice en la Figura 3-17.

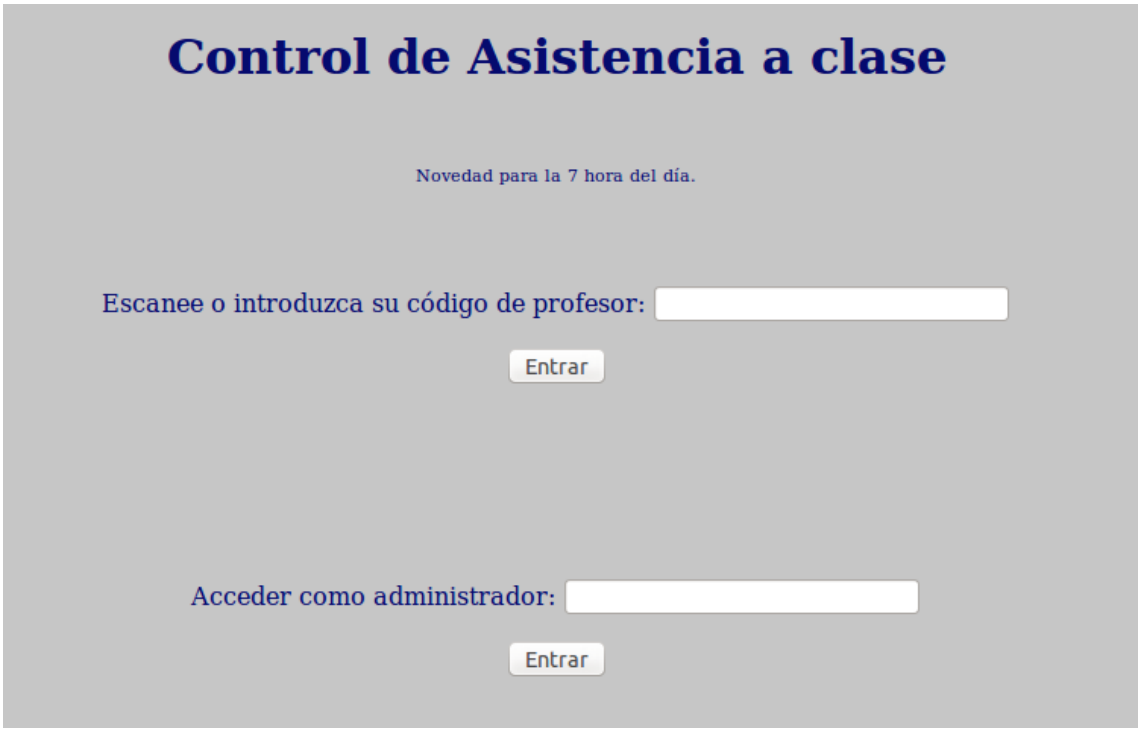


Figura 3-17 Presentación index.php

En el código de la página podemos ver que tiene también algunas líneas de PHP, utilizadas para vaciar la tabla de “Introducidos” de la base de datos, y una función *echo* que muestra por pantalla la hora del día dentro del horario de la ENM. Además incluye la posibilidad de mostrar un mensaje de error y uno de validación, que se muestran cuando se ha producido un error al introducir el código, o cuando el administrador borra las tablas de la base de datos.

La programación de esta página es muy sencilla y lo único destacable son las llamadas a las funciones para obtener la hora del día, programada en el modelo, y el borrado de la tabla de "Introducidos". Este borrado se realiza de manera automática al iniciar la aplicación, de forma que si la aplicación no se cerrase correctamente, la siguiente vez que se utilice no presentará datos erróneos en el campo de grupos cuya novedad ha sido introducida.

3.8.2.2 Vista1.php

Esta vista es el eje principal de la aplicación web. Desde ella se redirige a las demás vistas (a excepción de la de administración) y permite realizar varias acciones de interés. El código de esta vista se adjunta en el Anexo IV: vista1.php.

La primera vez que se accede a la vista, tras introducir correctamente el código del profesor, presenta, bajo la bienvenida al profesor, dos sencillos formularios, como se puede ver en la Figura 3-18. El primer formulario permite escanear o introducir el código de cualquiera de los grupos de clase para dirigirse a la siguiente página e introducir su novedad. El segundo formulario cuenta únicamente con un botón que permite salir de la vista, volviendo al índice.

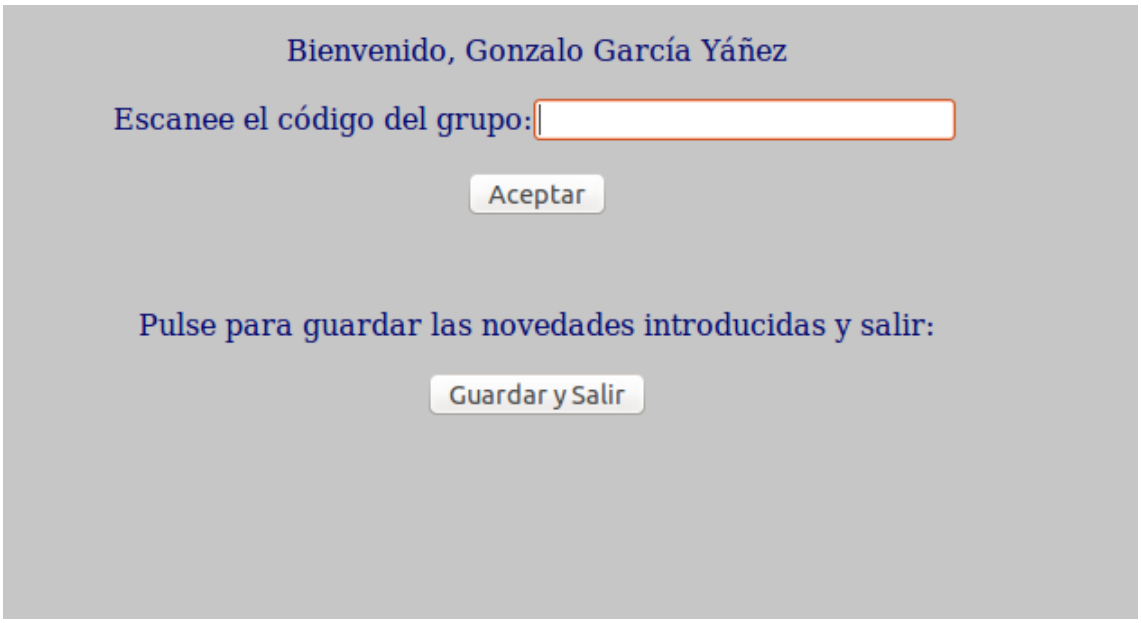


Figura 3-18 Presentación de vista1.php

Tras el primer formulario, aparece una ventana que obliga a seleccionar si la clase a impartir dura uno o dos periodos, antes de pasar a la siguiente vista. Podemos ver esta ventana en la Figura 3-19.

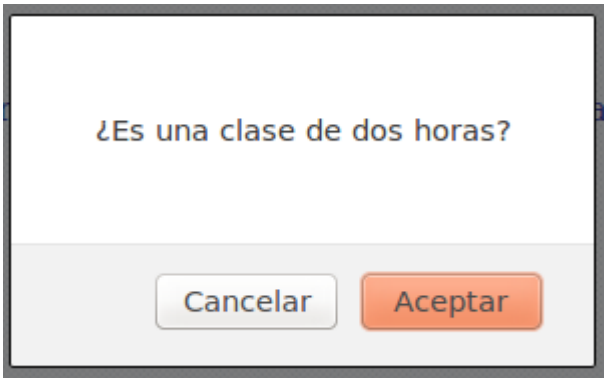


Figura 3-19 Ventana "*pop-up*" al introducir el grupo

Una vez la novedad de uno o varios grupos ha sido introducida, la presentación de la vista se completa, añadiendo dos nuevas funciones, como se aprecia en la Figura 3-20. Se muestran ahora los grupos cuya novedad ha sido introducida en la base de datos. Estos grupos se presentan en forma de enlace, de tal forma que con un sencillo pinchazo del ratón se puede consultar la novedad, en la vista 4.

Por otro lado, se presenta un nuevo botón que permite borrar alguna de las novedades introducidas dirigiéndose a la vista 3.

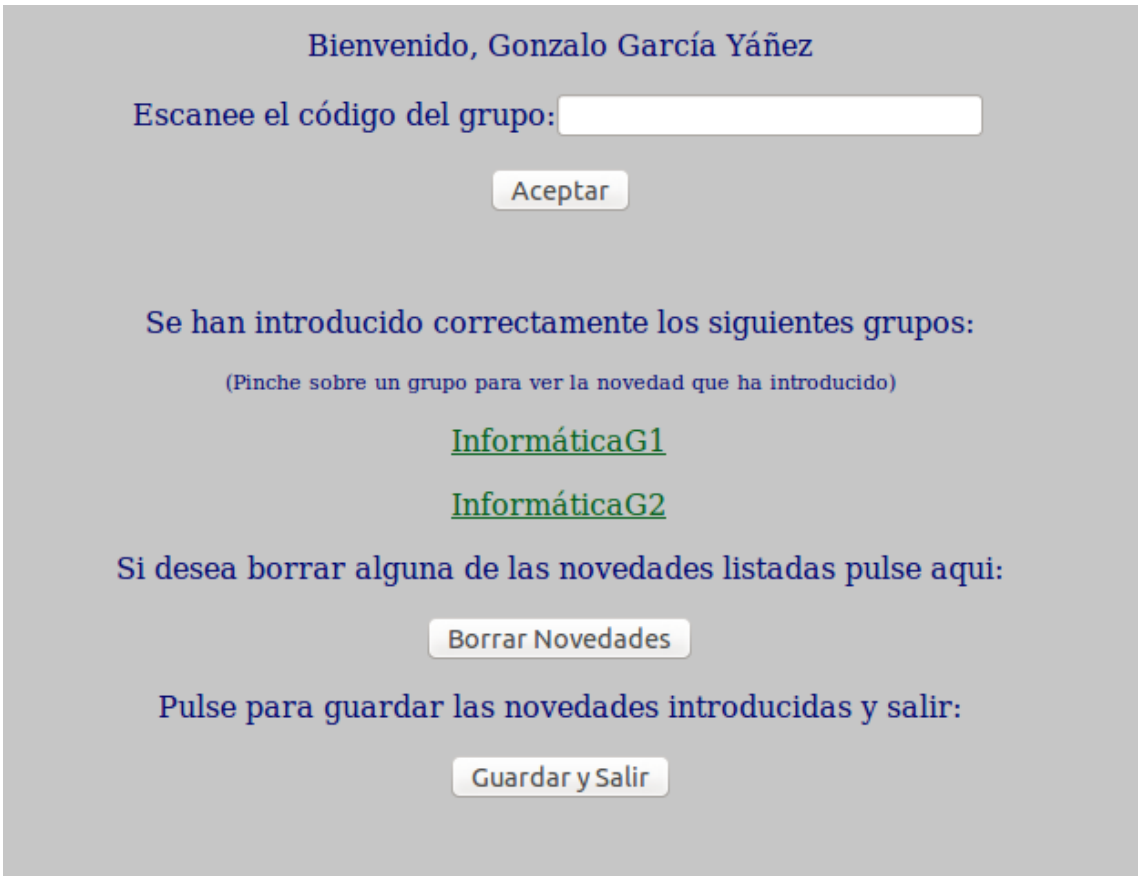


Figura 3-20 Presentación de vista1.php con grupos introducidos

El código que genera esta vista tiene dos aspectos destacables. El primero es la gran cantidad de código PHP que tiene para ser una vista. Esto es debido a que al ser una vista en la que se implementan varias funcionalidades, necesita comprobar condiciones y saber qué procesos se han realizado. El segundo aspecto, es la presencia de la etiqueta <*script*> seguida de código JavaScript, no utilizado hasta ahora en la memoria.

El objetivo de este código es la generación de la ventana que pide confirmación entre clase de una o dos horas. Estas ventanas, llamadas "*pop-up*", pueden ser presentadas por el navegador al interpretar el JavaScript.

La principal dificultad de la programación de esta página surge de la necesidad de almacenar novedades especificando si la clase dura uno o dos periodos. El objetivo es que el profesor no tenga que interrumpir su clase para introducir la novedad para la segunda hora. En un primer momento se optó por la creación de un campo de formulario que permitiese al profesor marcar la opción de hora doble, antes de escanear el grupo. Sin embargo, se comprobó que al introducir varios grupos seguidos era fácil olvidarse de marcar la opción en todos ellos, obligando a recurrir al borrado y nueva introducción de la novedad, proceso lento y tedioso.

Por ello se incorpora este sencillo código JavaScript [31], que muestra el *pop-up* de la Figura 3-19, permitiendo crear un sistema a prueba de despistes. Para ejecutar el *script*, se pulsa el botón del

formulario, por lo que es necesario inhabilitar en la introducción del texto, el envío automático con la tecla *Enter* del teclado.

3.8.2.3 Vista2.php

A la vista 2 se accede únicamente desde la vista 1 cuando se introduce un código de grupo válido. Como se puede ver en la Figura 3-21, esta vista presenta una tabla con todos los miembros de un grupo acompañada con un desplegable con todos los motivos de falta justificados que contempla el parte de clase actual, y que están recogidas en la tabla "Motivo" de la BDR. A continuación se envía la novedad a la base de datos y se vuelve a la vista 1.

Figura 3-21 Presentación de la vista2.php

En cuanto a la programación de esta vista, cuyo código se puede consultar en el Anexo V: vista2.php, cabe destacar la forma en la que PHP genera una tabla cuyo tamaño se define en función del número de alumnos del grupo.

La estructura de control *foreach* genera la tabla de la manera siguiente: Para cada elemento que contenga el *array "$nombrealumnos*", genera una columna con dos filas, la primera con el valor del elemento del *array* y la segunda con el desplegable.

El principal problema a la hora de programar esta página es realizar un guardado de datos que permita relacionar a cada alumno con su novedad, sin que se ordenen automáticamente. Se opta por incluir en cada fila de la tabla un contador, que aporta un identificador provisional a cada novedad. De esta forma se garantiza que las novedades se guarden en un *array* en el mismo orden en el que se guardan los alumnos. Esto resulta muy importante a la hora de introducir las novedades en la base de datos, como se verá más adelante.

3.8.2.4 Vista3.php

A la vista 3 también se accede desde la vista 1. Para poder llegar a la vista 3, es necesario antes haber introducido la novedad de algún grupo, en caso contrario no se mostrará el formulario que

incluye el botón que redirige a la vista 3. Se puede ver la estructura de control en el código del Anexo VI: vista3.php.

La presentación de la vista es muy sencilla, como se puede apreciar en la Figura 3-22. Contiene un campo de texto en el que se escanea o introduce el código del grupo cuya novedad se desee borrar, y un botón que permite volver a la vista anterior sin borrar ninguna novedad.

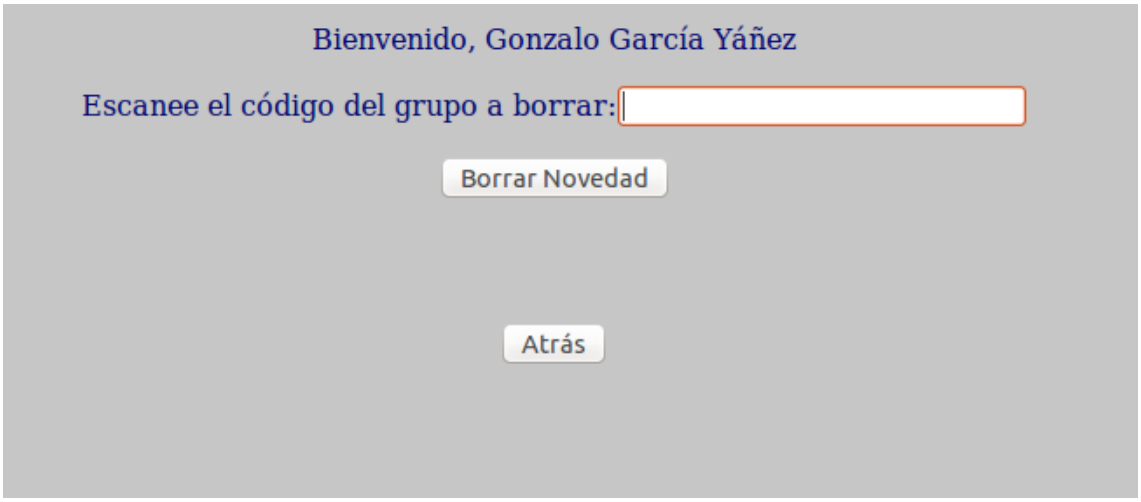


Figura 3-22 Presentación de vista3.php

Esta vista no tiene mayor dificultad en cuanto al código, pero sí que tiene aspectos complejos el efectuar un borrado de novedades que no produzca errores en la tabla. La solución para este problema se explica más adelante al describir el modelo.

3.8.2.5 Vista4.php

A esta página se llega desde la vista 1. Al igual que con la vista 3, a esta página sólo se puede acceder una vez ha sido introducida la novedad de algún grupo. A la vista se entra mediante un enlace con el nombre del grupo introducido, y presenta una tabla con la novedad de dicho grupo, como se puede ver en la Figura 3-23.

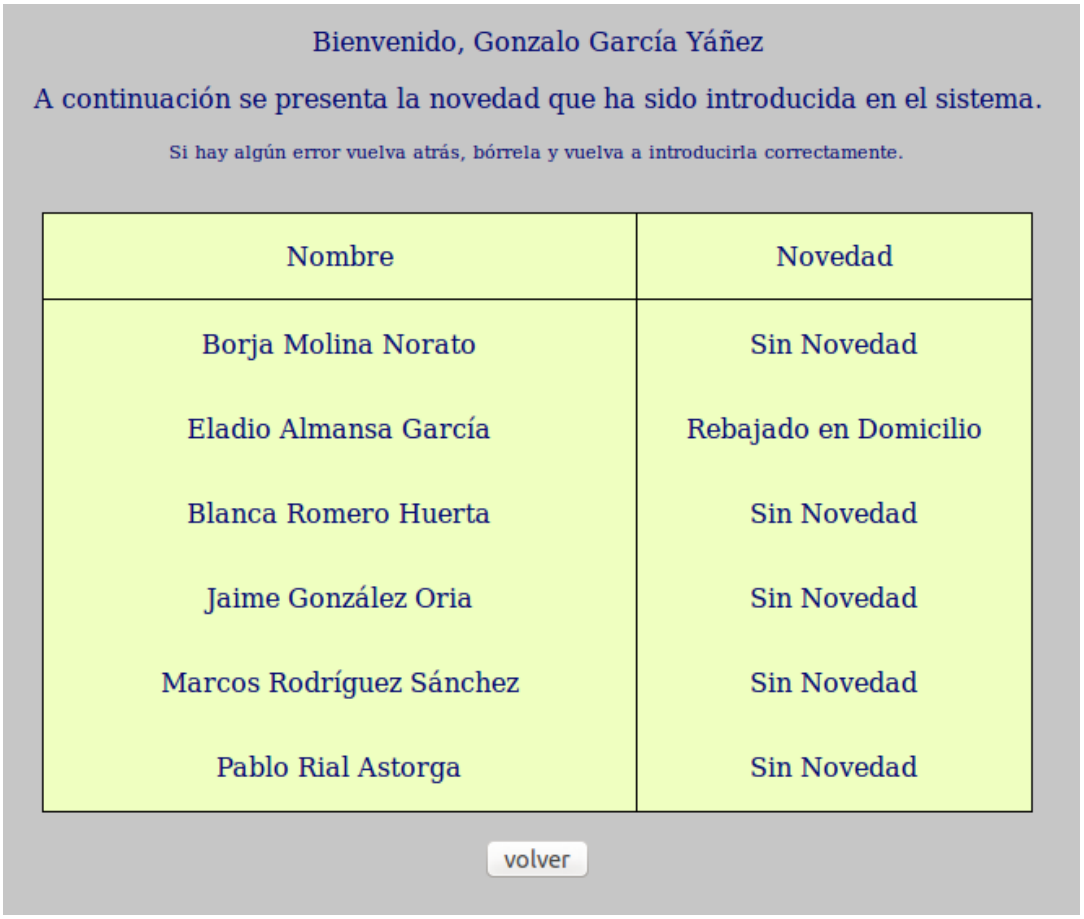


Figura 3-23 Presentación de vista4.php

El código, disponible en el Anexo VII: vista4.php, incluye una tabla de cuatro celdas. Las dos celdas superiores son los títulos de cada columna, mientras que en cada una de las celdas restantes se generan sendas tablas. Una tabla contiene todos los alumnos en orden, mientras que la segunda tabla representa las novedades introducidas, en el mismo orden. Ambas sub-tablas se dimensionan en función al número de alumnos en el grupo, de forma similar a la tabla de la vista 2.

3.8.2.6 Vista5.php

A la vista 5, cuyo código se encuentra recogido en el Anexo VIII: vista5.php, se accede desde el índice, rellenando el formulario que solicita un código de administrador. En esta vista se muestran dos enlaces que sirven para exportar las tablas Guardar y Novedades a formato Excel. La diferencia entre estas tablas se explica en la vista: La tabla Guardar contiene las novedades con los datos de nombres de alumnos, grupos, profesores y motivos; mientras que la tabla Novedades contiene la misma información pero con los códigos numéricos de identificación. Como se puede ver en la Figura 3-24, a continuación se incluyen las instrucciones de uso de la herramienta, que incluyen el borrado de las tablas de novedades, acción que se puede hacer con pulsar un botón.

Figura 3-24 Presentación de vista5.php

3.8.3 Estructura de control

Las vistas se comunican con el modelo a través del control, de acuerdo con la arquitectura MVC. El control se recoge en el documento "*control.php*", que se puede encontrar en el Anexo IX: control.php.

El funcionamiento del control se basa en los valores que toma la variable acción. En función del valor de la variable *$accion* se ejecutará una u otra secuencia de funciones. Como se puede apreciar en el código, el primer paso es requerir al modelo, para que se ejecuten las funciones descritas en este. A continuación se carga el valor de *$REQUEST["acción]* de los distintos formularios, comprueba que la variable ha sido definida y ejecuta diferentes funciones dependiendo de los valores de la variable.

Las funcionalidades principales que ejecuta este código se repiten en distintas acciones e incluyen la llamada a variables de formularios con *$_REQUEST*, la llamada a funciones con el código visto anteriormente, o el uso de estructuras de control como *for*, utilizada en varias ocasiones para recorrer todos los puestos de un *array*. También se realizan muchas cargas y guardados de variables en sesión, utilizados para entrar en funciones, o para accederr los resultados de las funciones invocadas en otras páginas.

En la Figura 3-25 se muestra el código de la primera de las posibilidades, en la cual la variable *$accion* tiene el valor "comprobacion". La variable toma este valor mediante el campo tipo *hidden* del formulario en el que el profesor se identifica con su código.

```
if ($accion == "comprobacion") {
        $idprof = $_REQUEST["idprofesor"];
        $idprofesor = comprobacion($idprof);
        $nombreprofesor = nombreprofesor($idprofesor);
        $_SESSION["nombreprofesor"] = $nombreprofesor;

        if ($idprofesor != NULL) {
                $_SESSION["idprofesor"] = $idprofesor;
                redirect("vista1.php");
        } else {
                redirect("index.php?accion=error");
        }
}
```

Figura 3-25 Segmento de código de control.php

Este bloque recupera del mismo formulario el valor del código del profesor y con la primera función comprueba que existe ese valor en la base de datos. Si existe, utiliza el identificador del profesor para obtener el nombre del profesor, lo guarda para ser utilizado más adelante, y redirige a la vista 1. En caso contrario vuelve a la vista del índice, añadiendo la acción de error.

3.8.4 Estructura de modelo

El modelo lo constituyen prácticamente todas las funciones que se ejecutan durante el uso del programa. Es el código más largo, que puede encontrarse en el Anexo X: modelo.php.

3.8.4.1 Modelo.php

En el modelo se encuentran más de veinte funciones, la mayoría interactúan con las tablas de la BDR, mediante las distintas variables de PHP. A continuación se explica brevemente qué hace cada función y cuándo se ejecuta:

- *function comprobacion*: Se ejecuta al introducir el código del profesor y comprueba que el código existe en la tabla Profesor de la BDR.
- *function comprobacionadministrador*: De manera análoga, comprueba la existencia del código introducido en el formulario de administrador en la tabla de Administrador.
- *function nombreprofesor*: Una vez se comprueba la existencia del identificador del profesor, se obtiene el nombre asociado en la tabla.
- *function nombregrupo*: Una vez en el formulario se introduce un código de grupo, se recupera de la tabla correspondiente su nombre.
- *function sacaridasignatura*: Hace la misma acción para obtener el identificador de la asignatura de la tabla Grupos.
- *function sacaridalumnos*: Guarda en un *array* los identificadores de todos los alumnos que pertenecen al grupo del código introducido.
- *function nombrealumnos*: Recorre el *array* generado por la función anterior, creando otro con los nombres de los alumnos correspondientes.
- *function guardarnovedad*: Esta función guarda la novedad de cada grupo una vez introducida en la vista 2. Se guarda con sus identificadores en la tabla de "Novedades".

- *function introducir*: Al introducir una novedad, añade el grupo en cuestión a la tabla "Introducidos", que permite saber los grupos cuya novedad ha sido introducida pero aún no guardada.
- *function sacarid*: Realiza la acción inversa, lee los grupos que hay guardados en la tabla "Introducidos", para poder generar los enlaces que dirigen a la vista 4.
- *function nombreintroducidos*: Obtiene los nombres de los grupos asociados a los identificadores obtenidos mediante la función anterior.
- *function borrarnovedad*: Borra la novedad del grupo cuyo código se introduce en la vista 3. Cabe destacar que esta función es la que más limita el funcionamiento del programa, como se verá en el apartado 3.9.
- *function salir*: Devuelve a la página de inicio, borrando los datos de la tabla de grupos "Introducidos".
- *function sacaridgrupo*: Realiza la acción contraria a la función *nombregrupo*, a partir del nombre de un grupo, obtiene el identificador asociado, para listar el grupo en la vista 3.
- *function idnovedad*: Obtiene de entre las novedades de la tabla, el identificador de la novedad de cada alumno, para un profesor y una hora del día concreta.
- *function nombrenovedad*: De la tabla de motivos selecciona el nombre asociado a los identificadores de la función anterior.
- *function sacarhora*: Esta función permite leer el grupo fecha hora del sistema y asociar un valor a cada intervalo horario. Se utiliza con el horario de la Escuela Naval para conocer la hora del día en base a la sábana de régimen interior. En caso de utilizar la aplicación fuera de los horarios habituales de clase, esta función devuelve el valor cero (0).
- *function vaciarnovedades*: Vacía, una vez lo ordena el administrador, las tablas de "Novedades" y "Guardar".
- *function convertirtabla*: Esta función se ejecuta de manera automática cuando un usuario accede a la zona reservada para la administración (vista 5). Esta función obtiene todos los campos de la tabla de Novedades, traduce los identificadores a sus nombres y los vuelca en la tabla Guardar.

3.8.4.2 Convertir.php y convertirids.php

Además del archivo "*modelo.php*", los archivos "*convertir.php*" y "*convertirids.php*" forman parte del modelo del programa. En ellos se encuentran las funciónes que exportan tablas de MySQL a formato Excel [18]. El primero convierte la tabla Guardar, mientras que el segundo traduce la tabla Novedades. Su código se puede revisar en el Anexo XI: convertir.php y el Anexo XII: convertirids.php.

3.8.5 Lectura, introducción y borrado de datos en la BDR

La principal dificultad del proyecto es que se produzca una interacción entre la página web y la base de datos MySQL que no conlleve errores. Como se puede apreciar en las funciones del modelo, la carga, descarga y consulta de información de las distintas tablas son procesos muy repetidos a lo largo de la ejecución del programa.

Es importante asegurarse de que no se producen errores en estas operaciones, que causarían que la aplicación fuese un estorbo en lugar de una ayuda para la gestión de la asistencia del alumnado.

Los principales errores que se produjeron durante la creación del proyecto estaban relacionados con el orden en el que se obtenían los datos de las diferentes tablas, o el orden en el que se volcaban.

Por ejemplo, al introducir diez alumnos, las diez novedades asociadas se guardaban en la tabla en otro orden. Este error es inadmisible pues supone una falsificación de los datos de asistencia.

Para solucionar estos errores es necesario programar las funciones que interactúan con MySQL con las condiciones suficientes para que la extracción y volcado de datos se realice de la manera deseada por el desarrollador.

Una vez resueltos estos errores, todos los datos de novedades se pueden gestionar desde la aplicación con total seguridad, sin embargo los datos de alumnos, profesores, grupos, asignaturas y motivos no se pueden modificar directamente a través de la aplicación.

Es necesario saber introducir datos en las tablas de la BDR mediante comandos MySQL, lo que puede ser demasiado trabajo si se realiza de manera manual. Sin embargo, se puede aliviar esta carga de trabajo al importar archivos Excel con los campos necesarios directamente a la base de datos.

Se crea un archivo Excel con los mismos campos que la tabla MySQL, en el mismo orden, y se guarda en formato "**.csv*". Este formato guarda la tabla como un código de texto en el cual se separa cada celda por una coma (,). Para cargar este archivo a la tabla se utiliza el comando:

- *LOAD DATA LOCAL INFILE '/home/alumno/Escritorio/grupos.csv' INTO TABLE Grupos FIELDS TERMINATED BY ',';*

En el ejemplo, se muestra la ruta del archivo con la información de los distintos grupos de clase, el nombre de la tabla en la que se desean guardar estos datos, y el símbolo que separa las distintas celdas de la tabla, en este caso la coma.

De esta manera se cargarán las tablas con la información de la Escuela Naval para poder implantar el uso de la aplicación.

3.9 Uso y gestión de la aplicación

Se ha diseñado la aplicación para que su uso sea lo más sencillo posible, sin embargo es necesario que el administrador de la aplicación conozca sus limitaciones y posibles causas de error.

La aplicación ha sido diseñada para sustituir el actual sistema de partes de clase, y puede almacenar grandes cantidades de novedades. El programa funciona con la siguiente secuencia:

Cada profesor recibe una tarjeta con su código personal y un código de barras, lo que le permite identificarse para acceder a la aplicación. En caso de que el escáner de códigos no funcione, o que se quiera acceder desde un dispositivo móvil, se puede introducir el código con el teclado. A continuación, los jefes de grupo se acercan con la ficha de grupo y escanean su código, para que el profesor introduzca la novedad de los alumnos faltos. Se guarda la novedad y se cierra la aplicación para dar comienzo a la clase. Ambas propuestas de documento se pueden encontrar en el Anexo XIV: Tarjetas de profesor y ficha de clase. Además permite que el profesor visualice las novedades que ha introducido, e incluso borrarlas para volver a introducirlas en caso de que las haya introducido mal.

A fin de actividades, cuando normalmente se entregan todos los partes de clase, el administrador debe acceder con su código y descargar las novedades en formato Excel. A continuación se recomienda borrar la base de datos para tenerla vacía para un nuevo día.

La razón de borrar cada día la base de datos de novedades se debe a una de las posibles causas de error del programa. En la función *borrarnovedad* se permite borrar la novedad de un grupo que haya sido mal introducida por el profesor. El objetivo es evitar que se borre de la base de datos otra novedad que no sea la deseada, esto se logra borrando la novedad que tenga el mismo identificador de profesor que el que haya accedido al programa, el mismo identificador de grupo que el introducido y la misma hora del día dentro del horario. Es decir, si se supone que el grupo 1 de Informática tiene clase con el mismo profesor por la mañana de teoría y por la tarde de seminario, y que por la tarde el profesor

introduce mal la novedad y quiere borrarla. Si la borra no se produce ningún error en la aplicación, pues aunque coindicen el profesor y el grupo, no coincide la hora del horario, y por lo tanto se borraría la novedad de la tarde.

Sin embargo sí que se produce un error cuando el mismo profesor imparte una clase un día de la semana a una cierta hora, y otra clase de la misma asignatura otro día a la misma hora. Al borrar la novedad del grupo el segundo día, involuntariamente y sin ser consciente de ello, borrará la novedad del día anterior, pues coinciden tanto el profesor, como el grupo, como la hora del día.

Esta es toda la gestión que requiere la aplicación, además de la carga de datos a las tablas de apoyo, que se puede hacer de forma directa con el código descrito en el apartado anterior, por lo que se puede considerar que la administración del programa es muy sencilla.

4 VALIDACIÓN

4.1 Generación de una base de datos para pruebas en Excel

En primer lugar es necesario crear una base de datos ficticia que contenga alumnos distribuidos en grupos, diferentes profesores y asignaturas. Los motivos han sido proporcionados por Jefatura de Estudios de la ENM, y son los mismos que contempla hoy en día el parte de clase. Esta base de datos se puede consultar en el Anexo XIII: Tablas excel de la base de datos para la realización de pruebas de validación.

Para ello se diseña una base de datos que contempla las posibles situaciones que podrían confundir al programa. Se generan los datos ficticios de 43 alumnos, repartidos en tres cursos. Dentro de un curso hay tres grupos, dos del Cuerpo General de la Armada y uno de Infantería de Marina, mientras que los otros dos cursos cuentan con un grupo cada uno. Además, se han introducido tres alumnos con situaciones especiales. Dos alumnos que repiten curso y uno que ha convalidado asignaturas, por lo que tienen horarios únicos, y forman grupos unipersonales.

Además se generan tres asignaturas por curso, aunque los alumnos en situación extraordinaria están matriculados de asignaturas de cursos superiores. Se idean también los datos ficticios de diez profesores y diez asignaturas. Los profesores se relacionan con las asignaturas de forma que un profesor puede impartir más de una asignatura, al igual que una asignatura puede ser impartida por varios profesores.

4.2 Lectura y registro de datos de Excel en la base de datos.

A continuación se cargan los datos en las correspondientes tablas con el comando mostrado en el apartado 3.8.5, y se comprueba que los datos han sido correctamente interpretados e introducidos. En la Figura 4-1 se ve como, efectivamente se han cargado los datos de los profesores correctamente, de acuerdo a la base de datos ficticia. Se realiza la comprobación de manera análoga en el resto de tablas y se aprecia que los datos han sido cargados.

```
mysql> Select * from Profesor;
+------------+---------------------------+--------+-----------------------+
| idprofesor | nombre                    | codbar | correo                |
+------------+---------------------------+--------+-----------------------+
|          1 | Gonzalo García Yáñez      |      1 | garcía@ejemplo.com    |
|          2 | Eugenia Rodríguez Alonso  |      2 | rodriguez@ejemplo.com |
|          3 | Engracia Villalba García  |      3 | villalba@ejemplo.com  |
|          4 | Francisco Polo Hernández  |      4 | polo@ejemplo.com      |
|          5 | Miriam Vila Rodríguez     |      5 | vila@ejempo.com       |
|          6 | Alberto Rial Freire       |      6 | rial@ejemplo.com      |
|          7 | Sofía Dapena Iglesias     |      7 | dapena@ejemplo.com    |
|          8 | Ernesto Miguélez Cuenca   |      8 | miguelez@ejemplo.com  |
|          9 | Ignacio Pascual Cerrada   |      9 | pascual@ejemplo.com   |
|         10 | Claudia Mateos Cruz       |     10 | mateos@ejemplo.com    |
+------------+---------------------------+--------+-----------------------+
10 rows in set (0.00 sec)
```

Figura 4-1 Presentación en MySQL de los datos de la tabla Profesor

De esta manera hemos conseguido validar el funcionamiento de los comandos de importación de datos a las tablas de la BDR desde archivos Excel.

4.3 Prueba de funcionamiento de la aplicación

4.3.1 Uso normal sin errores

Se realizará la prueba de la aplicación en la que la profesora Miriam Vila Rodríguez, (id 5), imparte una clase de Informática a los grupos “G1” (id 1) y “G2” (id 2) y a un repetidor (id 3). En esta clase, el alumno Luis Astorga Boccherini (id 1) está de guardia, mientras que el alumno Ignacio Rodrigáñez Ribas (id 10) se encuentra hospitalizado. Por último, el administrador descarga la novedad en formato Excel.

El proceso de la aplicación se puede seguir con las Figura 4-2, Figura 4-3, Figura 4-4, Figura 4-5 y Figura 4-6:

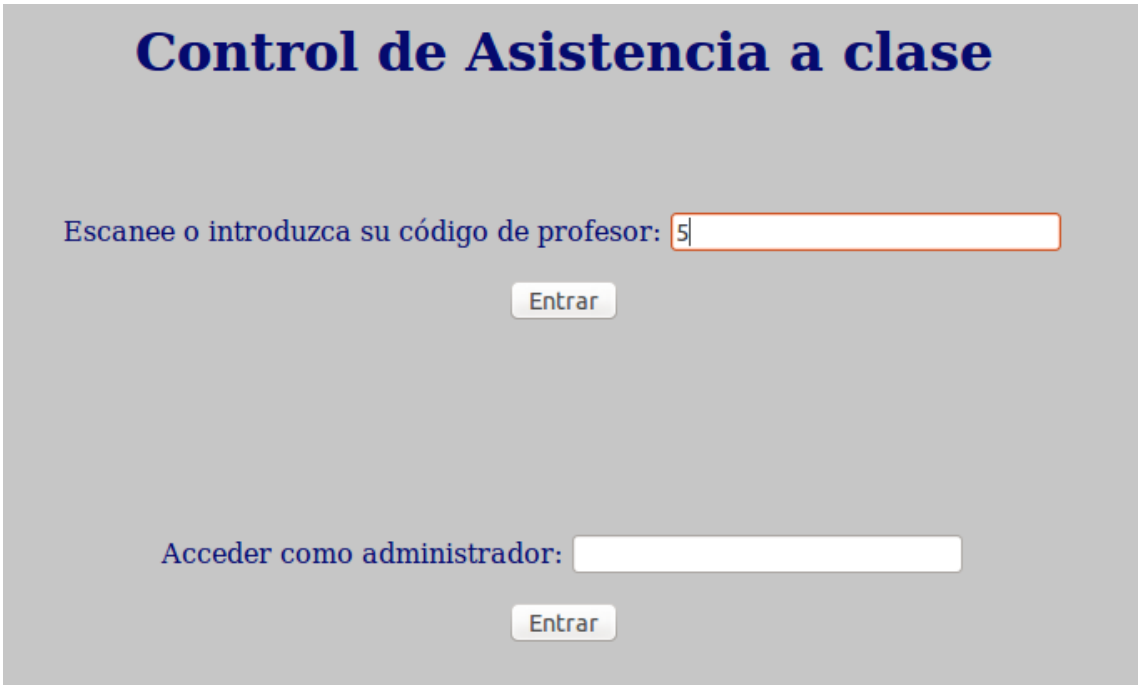


Figura 4-2 Presentación de index.php al introducir un código

Figura 4-3 Presentación de vista1.php con el nombre del profesor

Figura 4-4 Presentación de vista2.php introduciendo la novedad de los alumnos

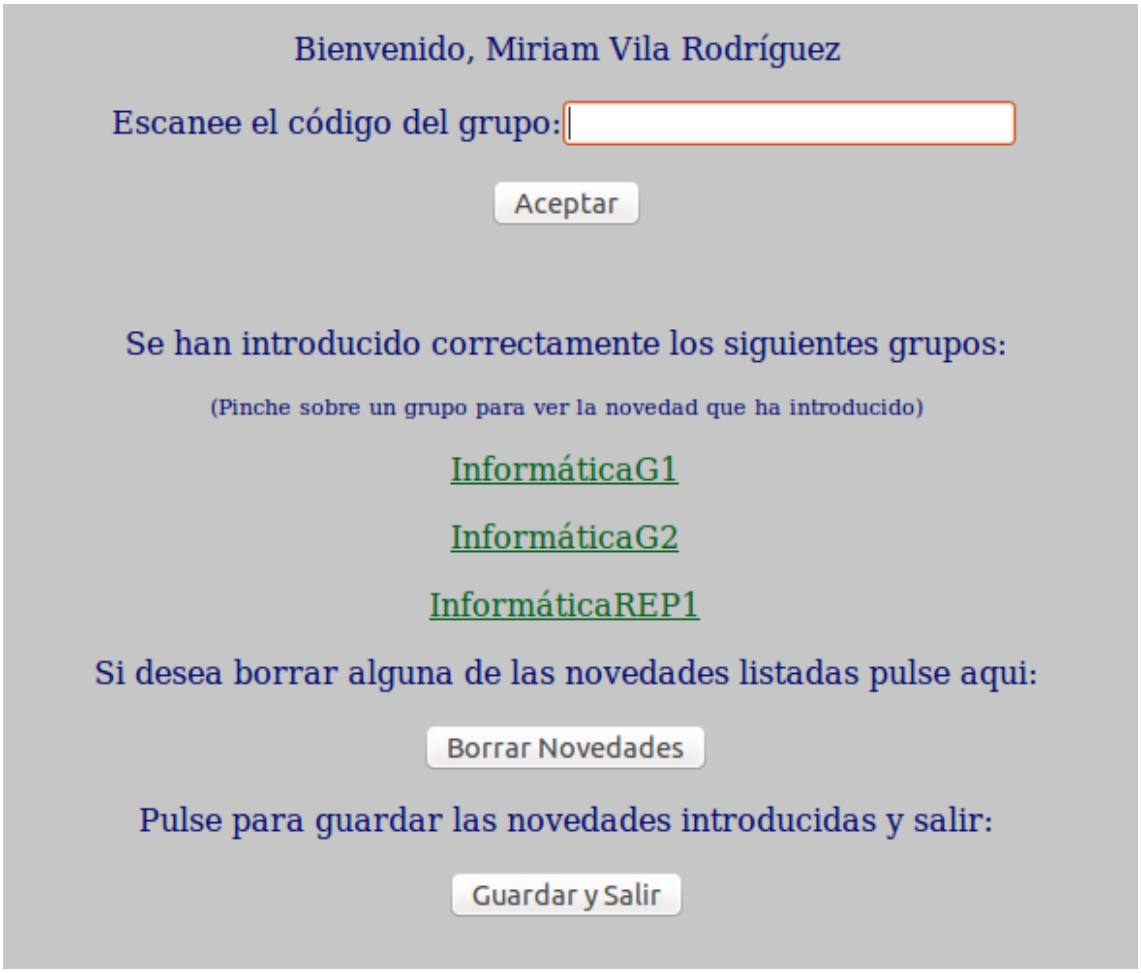


Figura 4-5 Presentación de vista1.php con los grupos introducidos

nombrealumno	nombregrupo	nombreprofesor	nombremotivo
Luis Astorga Boccherini	InformáticaG1	Miriam Vila Rodríguez	Guardia
Carlos Arenas Pérez-Seoane	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Carlos Martínez de Baños Martínez de Morentín	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Gerardo González-Aller Díaz del Río	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Carlos Bonaplata Hernández de Armijo	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Santiago Carlos Bausá Viseras	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Carlos Eustaquio Gallego Montero	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Jose Luis Burrueco Jiménez	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Guillermo Busto Cuiñas	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Ignacio Rodrigáñez Ribas	InformáticaG1	Miriam Vila Rodríguez	Hospitalizado
Juan Antonio Vidal Sáchez	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Gonzalo Peñuelas Pérez-Cuadrado	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Guillermo Gilabert Gamoa	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Cristina Martínez Merello Graña	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Juan de la Torre Díaz	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Enrique de la Torre Martínez	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Pablo Vega Vegas	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Óscar Vega Vegas	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Federico Luján Cuesta	InformáticaREP1	Miriam Vila Rodríguez	Sin Novedad

Figura 4-6 Resultado de la exportación a Excel de las novedades guardadas

Podemos observar que con el uso normal de la aplicación se guarda y exporta correctamente la novedad.

4.3.2 Uso forzando errores para la detección de fallos

En este caso, se realizará la misma prueba que en el caso anterior, con las siguientes diferencias: Antes de introducir su código, la profesora introduce un código erróneo, lo mismo sucede al intentar introducir el grupo G2. Además introduce mal la novedad del grupo G1, poniendo al alumno Carlos Martínez de Baños Martínez de Morentín (id 3) de guardia. La profesora borrará la novedad y la volverá a introducir correctamente. Por último, el administrador, tras equivocarse con su código, accede y descarga la novedad en formato Excel, donde se comprobará si la aplicación ha logrado registrar correctamente la novedad.

La Figura 4-7, Figura 4-8, Figura 4-9, Figura 4-10, Figura 4-11, Figura 4-12 y Figura 4-13 muestran las diferencias respecto al proceso anterior en este nuevo caso:

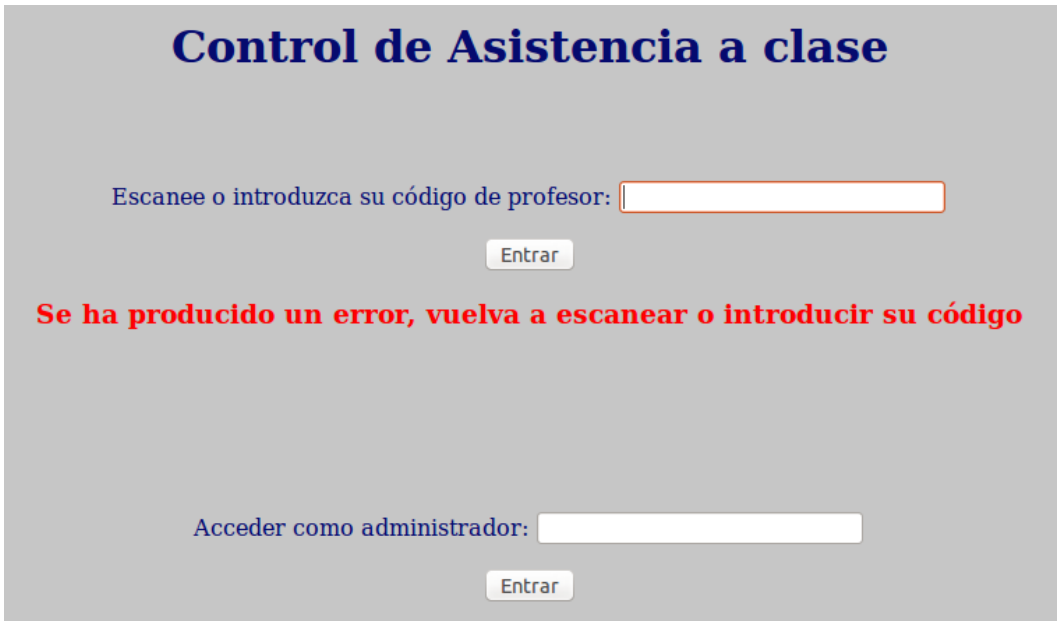


Figura 4-7 Presentación de index.php al introducir un código de profesor incorrecto

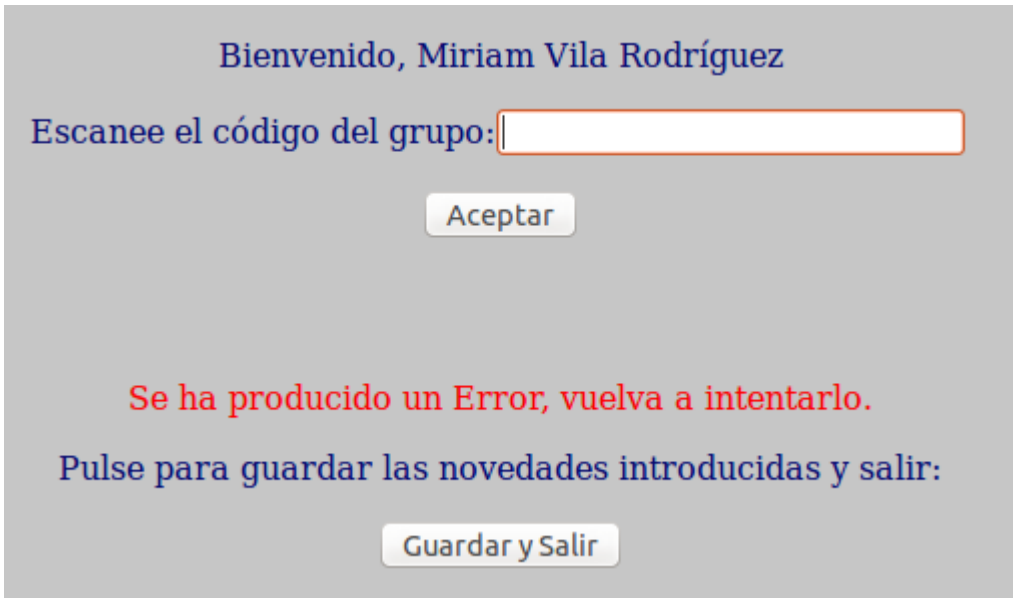


Figura 4-8 Presentación de vista1.php al introducir un código de grupo incorrecto

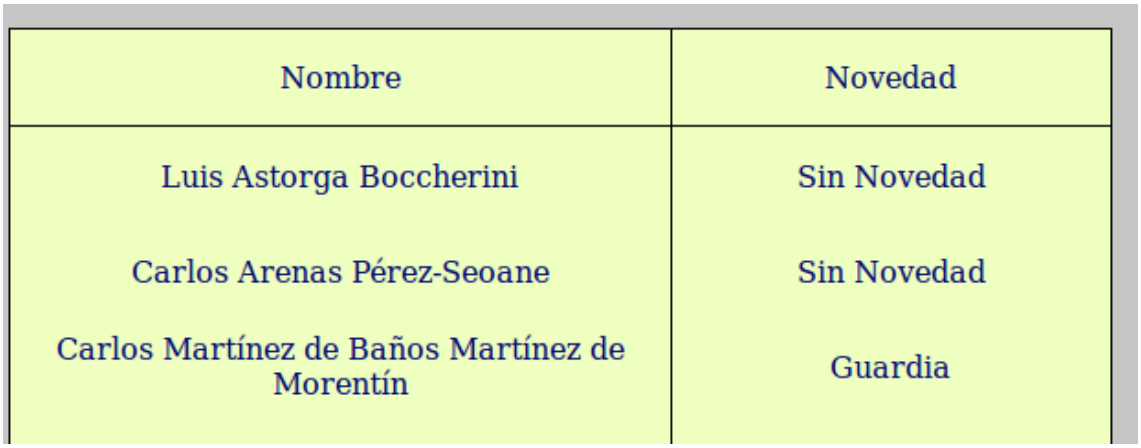


Figura 4-9 Sección de vista4.php en la que se aprecia el error al introducir la novedad

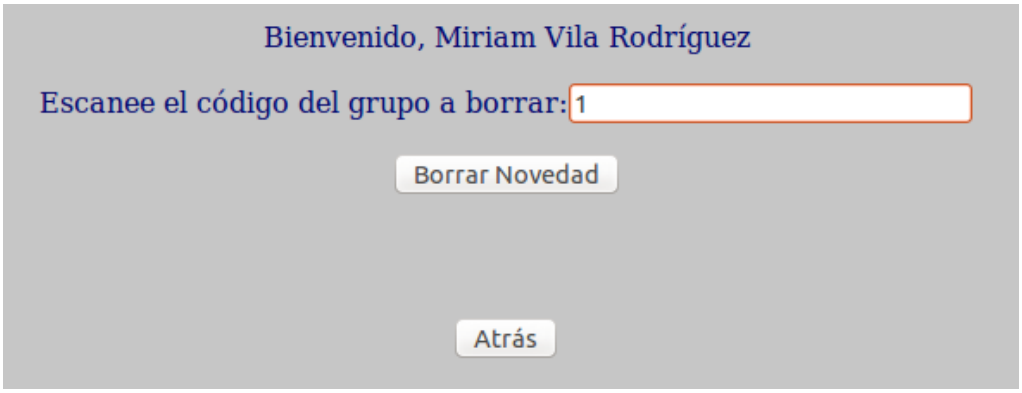


Figura 4-10 Presentación de vista3.php en la que se introduce el grupo a borrar

Figura 4-11 Sección de vista1.php en la que se aprecia el borrado del grupo InformáticaG1

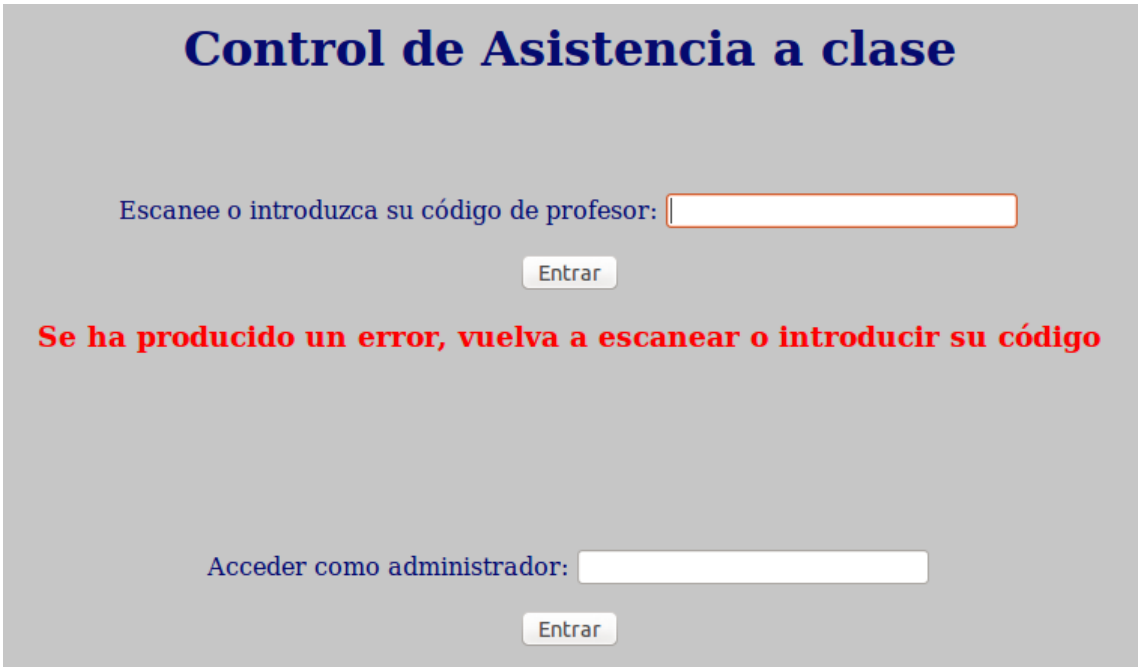


Figura 4-12 Presentación de index.php al introducir un código de administrador erróneo

nombrealumno	nombregrupo	nombreprofesor	nombremotivo
Luis Astorga Boccherini	InformáticaG1	Miriam Vila Rodríguez	Guardia
Carlos Arenas Pérez-Seoane	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Carlos Martínez de Baños Martínez de Morentín	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Gerardo González-Aller Díaz del Río	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Carlos Bonaplata Hernández de Armijo	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Santiago Carlos Bausá Viseras	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Carlos Eustaquio Gallego Montero	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Jose Luis Burrueco Jiménez	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Guillermo Busto Cuiñas	InformáticaG1	Miriam Vila Rodríguez	Sin Novedad
Ignacio Rodrigáñez Ribas	InformáticaG1	Miriam Vila Rodríguez	Permiso
Juan Antonio Vidal Sáchez	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Gonzalo Peñuelas Pérez-Cuadrado	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Guillermo Gilabert Gamoa	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Cristina Martínez Merello Graña	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Juan de la Torre Díaz	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Enrique de la Torre Martínez	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Pablo Vega Vegas	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Óscar Vega Vegas	InformáticaG2	Miriam Vila Rodríguez	Sin Novedad
Federico Luján Cuesta	InformáticaREP1	Miriam Vila Rodríguez	Sin Novedad

Figura 4-13 Resultado de la exportación a Excel de la novedad guardada

Se puede apreciar que la novedad se guarda correctamente incluso cuando se producen errores, o el borrado de datos y que el propio programa no permite trabajar con códigos que no sean válidos.

4.4 Almacenamiento de novedades en la base de datos y exportación a formato Excel

Como se presenta en los apartados anteriores, una de las grandes ventajas de la aplicación es la posibilidad de importar a Excel la tabla con las novedades almacenadas. Se comprueba, para el ejemplo anterior, que la tabla recoge las novedades correctamente, como se muestra en la Figura 4-14.

```
+-----+-------------------------------------------------+-----------------+----------------------+-------------+---------------------+----
--+-------+
| id  | nombrealumno                                    | nombregrupo     | nombreprofesor       | nombremotivo | creado              | hor
a | doble |
+-----+-------------------------------------------------+-----------------+----------------------+-------------+---------------------+----
--+-------+
| 129 | Luis Astorga Boccherini                         | InformáticaG1   | Miriam Vila Rodriguez | Guardia     | 2016-02-25 01:08:49 |
0 |     0 |
| 130 | Carlos Arenas Pérez-Seoane                      | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:08:50 |
0 |     0 |
| 131 | Carlos Martínez de Baños Martínez de Morentín   | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:08:50 |
0 |     0 |
| 132 | Gerardo González-Aller Díaz del Río             | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:08:50 |
0 |     0 |
| 133 | Carlos Bonaplata Hernández de Armijo            | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:08:50 |
0 |     0 |
| 134 | Santiago Carlos Bausá Viseras                   | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:08:50 |
0 |     0 |
| 135 | Carlos Eustaquio Gallego Montero                | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:08:50 |
0 |     0 |
| 136 | Jose Luis Burrueco Jiménez                      | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:08:50 |
0 |     0 |
| 137 | Guillermo Busto Cuiñas                          | InformáticaG1   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:09:00 |
0 |     1 |
| 138 | Ignacio Rodrigáñez Ribas                        | InformáticaG1   | Miriam Vila Rodriguez | Permiso     | 2016-02-25 01:10:50 |
0 |     0 |
| 139 | Juan Antonio Vidal Sáchez                       | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 140 | Gonzalo Peñuelas Pérez-Cuadrado                 | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 141 | Guillermo Gilabert Gamoa                        | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 142 | Cristina Martínez Merello Grana                 | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 143 | Juan de la Torre Díaz                           | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 144 | Enrique de la Torre Martínez                    | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 145 | Pablo Vega Vegas                                | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 146 | Óscar Vega Vegas                                | InformáticaG2   | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
| 147 | Federico Luján Cuesta                           | InformáticaREP1 | Miriam Vila Rodriguez | Sin Novedad | 2016-02-25 01:10:50 |
0 |     0 |
+-----+-------------------------------------------------+-----------------+----------------------+-------------+---------------------+----
--+-------+
19 rows in set (0.01 sec)
```

Figura 4-14 Datos de la tabla Guardar en MySQL

Lo mismo sucede con la tabla que guarda los identificadores.

Gracias a esta característica de la aplicación, se pueden procesar las faltas de los alumnos en cuestión de minutos, mediante filtros de búsqueda en el archivo Excel, ordenando la tabla por nombres, motivos, profesores o grupos.

A continuación estos datos se deben exportar a otro Excel que contengan todas las novedades del periodo que se quiera estudiar, con el fin de recabar las estadísticas deseadas.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Cumplimiento de objetivos

Una vez realizado el trabajo, se vuelve sobre los objetivos planteados en su inicio, en el apartado 1.5, y en base a estos, se valora la aplicación implementada y su utilidad.

Se definió el objetivo principal del trabajo como "la implementación de una herramienta informática que permita sentar las bases de un programa que gestione y automatice el control de asistencia a clase de los alumnos de la Escuela Naval Militar, proveyendo estadísticas personalizadas de los alumnos y profesores". Este objetivo se considera parcialmente cumplido, pues el programa ya gestiona el control de la asistencia a clase, y realiza el proceso de una forma automática, presentando a Jefatura de Estudios la novedad recopilada en formato Excel. Sin embargo, el aspecto de la obtención de estadísticas no ha sido completamente logrado, pues las estadísticas han de ser analizadas con Excel, y no se procesan de modo automático. Por otro lado, se ha diseñado una base de datos que permite trabajar sobre esta aplicación para incorporar más adelante la gestión estadística sin necesidad de empezar de cero.

El objetivo secundario se considera logrado, pues el programa se ha diseñado logrando que su uso sea sencillo, de forma que no supone ningún esfuerzo añadido con respecto al sistema actual, ni por parte del profesor ni de los alumnos. Con el sistema del parte de clase, el jefe de grupo tenía que recoger partes en blanco a primera hora de la mañana y entregarlo a última hora de la tarde, ahora sólo tiene que acudir a clase con su ficha de grupo.

Además, el sistema actual obliga al personal de Jefatura de Estudios a recopilar la información de hasta 60 partes de clase, interpretando la caligrafía de los alumnos, y tipificando el nombre de las asignaturas para su introducción a un Excel. Este paso sería completamente eliminado, aliviando de una gran carga de trabajo al personal de la ENM.

Se pretendía lograr tres ventajas con la implantación de la aplicación sobre la que versa este trabajo. Las ventajas, listadas en el apartado 1.5 y que cito textualmente son:

"La consecución de estos objetivos principales viene motivada por lograr las ventajas siguientes respecto al sistema actual:

- Eliminar la dependencia del parte de clase en papel para el control de asistencia.
- Reducir el consumo de papel, logrando también un ahorro material a medio y largo plazo.
- Acercar la Escuela Naval Militar a las herramientas tecnológicas que utilizan todos los centros educativos de prestigio, categoría a la que debe pertenecer la ENM."

Como se puede apreciar, se basan en eliminar el parte de clase en papel, con las consecuencias que ello conlleva. Esto se considera también cumplido, pues la aplicación ya puede ser usada para sustituir el actual parte de clase.

Sin embargo, la aplicación dista aún de ser una herramienta completa, por lo que se proponen avances que se pueden realizar sobre la base de este programa y que ayudarán a crear un sistema de control de asistencia mucho más completo y automático.

5.2 Generación automática de códigos de barras

5.2.1 Generación automática de códigos de barras

Los códigos de barras utilizados para las fichas de grupos y tarjetas de profesores requeridos para la validación del programa han sido creados manualmente con la herramienta gratuita Barcode Generator [32].

La introducción manual de todos los códigos de barras que identifican profesores y grupos a la escala que se maneja en la Escuela Naval es una tarea tediosa y puede llevar a errores.

Por ello se puede mejorar la aplicación creando un sistema automático de generación de códigos de barras. Leyendo los datos de la BDR actual, se puede hacer que las fichas de grupo y las tarjetas de profesor se generen automáticamente listas para ser impresas.

Existen ya librería de PHP que permiten la generación automática de códigos de barras, por lo que su implantación resultará sencilla [33].

Esto aportaría también flexibilidad a la aplicación a la hora de cambiar los grupos, o incorporar nuevos profesores a la plantilla docente.

5.2.2 Procesado y almacenamiento de estadísticas

La principal carencia de la aplicación consiste en no haber logrado un procesado automático de estadísticas. Por ello se propone como línea futura, sobre la base del presente trabajo, generar un sistema de creación de estadísticas, gráficos y búsqueda filtrada por alumnos, grupos, profesores, asignaturas o motivos.

Para ello también es necesario incluir un sistema que permita garantizar el almacenado de todas las estadísticas durante el periodo de tiempo que se considere necesario, sin que se pueda producir el borrado no deseado de alguna novedad.

5.2.3 Alertas a profesores por correo electrónico

Este programa ha sido diseñado como una herramienta de ayuda a profesores, jefes de grupo y personal de la ENM, no como una herramienta con fines disciplinarios, por lo que no contempla medidas de seguridad avanzadas en caso de intentar falsificar datos de novedades.

Una vez se valida la novedad, sólo un administrador puede borrarla, sin embargo, es posible que un alumno intente hacerse pasar por profesor para introducir novedades falsas en la base de datos.

Por ello se propone utilizar el campo de correo electrónico del profesor para generar un correo estándar que se envíe de manera automática al profesor una vez guarda la novedad del grupo. Este correo se enviaría al pulsar el botón "*Guardar y salir*" de la vista principal y permitirá detectar si un alumno introduce datos falsos en la tabla. Así se podrá avisar al administrador para que borre las novedades adulteradas de la tabla.

5.2.4 Lectura desde dispositivos móviles

Otra de las mejoras que se pueden implementar en la aplicación, es la creación de un formato de presentación web que se adapte al creciente uso de dispositivos móviles. El diseño actual del programa se presenta en una pantalla móvil de la misma manera que en el monitor del ordenador.

Sin embargo, mediante el uso de librerías como *Bootstrap* [14] [34] [35], se puede llegar a un diseño que adapte su presentación al tamaño de la pantalla del dispositivo utilizado. Esto permite usar la aplicación con mayor comodidad en las actividades prácticas como embarques, prácticas de campo, o instrucción militar, muy presentes en el horario de la Escuela.

Sería necesario para su correcto funcionamiento, la programación de una aplicación que introduzca en los formularios el valor de un código de barras leído por la cámara del dispositivo. Mientras se logra esa aplicación, de la que se pueden encontrar múltiples referencias en el mercado actual, como es el ejemplo de QuaggaJS [36], se puede seguir utilizando la herramienta, pero con el inconveniente de que se obliga al profesor a introducir el valor numérico de todos los datos de identificación a través del teclado.

5.3 Conclusión

El programa diseñado en este trabajo está listo para ser implantado en la Escuela Naval Militar, pasando por el proceso de validación que solicite el personal de la Jefatura de Estudios. El programa ya permitirá aliviar considerablemente la carga de trabajo que supone el control de asistencia a clase, especialmente para el personal encargado de procesar a diario los partes de clase.

Aunque la aplicación puede incorporar muchas funcionalidades, se ha sentado una base sólida sobre la que puede apoyarse el código de estas nuevas funciones sin necesidad de reestructurar la aplicación ni empezar un nuevo proyecto desde cero.

6 BIBLIOGRAFÍA

[1] A. Española, «Página oficial de la Armada,» [En línea]. Available: http://www.armada.mde.es/ArmadaPortal/page/Portal/ArmadaEspannola/personal_enm/prefLang_es/00_inicio. [Último acceso: 25 Febrero 2016].

[2] Centro Universitario de la Defensa, «Página del CUD de Vigo,» [En línea]. Available: https://www.cud.uvigo.es. [Último acceso: 2016 Febrero 25].

[3] LEXTREND, «Plataforma LEXTREND,» [En línea]. Available: http://lextrend.com/el-control-de-asistencia-digital-elimina-la-burocracias-de-las-escuelas/. [Último acceso: 2016 Febrero 25].

[4] LEXTREND, «Software Tracesign,» [En línea]. Available: http://lextrend.com/tracesign-software-de-control-de-asistencia-con-tecnologia-nfc/. [Último acceso: 25 Febrero 2016].

[5] Ender, «Ender, software Atenea,» [En línea]. Available: http://www.ender.es/software-para-empresas-de-formacion/gestion-de-academias-online/. [Último acceso: 2016 Febrero 25].

[6] Ender, «Empresa Ender,» [En línea]. Available: http://www.ender.es/2011/07/atenea-nuestro-sistema-web-para-la-gestion-de-academias-y-empresas-de-formacion/. [Último acceso: 25 Febrero 2016].

[7] w3schools, «w3chools HTML,» [En línea]. Available: http://www.w3schools.com/html/default.asp. [Último acceso: 23 Febrero 2016].

[8] webadictos, «webadictos historia HTML,» [En línea]. Available: http://webadictos.com/2012/12/30/breve-historia-del-html/ . [Último acceso: 23 Febrero 2016].

[9] W3C, «W3C Org,» [En línea]. Available: https://www.w3.org/. [Último acceso: 23 Febrero 2016].

[10] librosweb, «librosweb historia HTML,» [En línea]. Available: http://librosweb.es/libro/xhtml/capitulo_1/breve_historia_de_html.html. [Último acceso: 21 Febrero 2016].

[11] W3C, «W3C Berners Lee,» [En línea]. Available: https://www.w3.org/People/Berners-Lee/. [Último acceso: 23 Febrero 2016].

[12] Apache, «Apache org,» [En línea]. Available: http://www.apache.org/. [Último acceso: 23 Febrero 2016].

[13] Webopedia, «Webopedia LAMP,» [En línea]. Available: http://www.webopedia.com/TERM/L/LAMP.html. [Último acceso: 2016 Febrero 25].

[14] L. M. Cabezas Granado y F. J. González Lozano, Desrrollo Web con PHP y MySQL, Madrid: Anaya Multimedia, 20015.

[15] httpd, «HTTPD About Apache,» [En línea]. Available: https://httpd.apache.org/ABOUT_APACHE.html. [Último acceso: 25 Febrero 2016].

[16] Apache, «Apache Software Foundation,» [En línea]. Available: http://www.apache.org/foundation/. [Último acceso: 25 Febrero 2016].

[17] Cibernetia, «historia apache,» [En línea]. Available: http://www.cibernetia.com/manuales/instalacion_servidor_web/2_1_historia_apache.php. [Último acceso: 2016 Febrero 20].

[18] O. Heurtel, PHP y MySQL Domine el desarrollo de un sitio web dinámico e interactivo, Barcelona: Ediciones ENI, 2014.

[19] Oracle, «mysql,» [En línea]. Available: http://www.mysql.com/. [Último acceso: 25 Febrero 2016].

[20] Databasefriends, «History of MySQL,» [En línea]. Available: http://www.databasefriends.co/2014/02/history-of-mysql.html. [Último acceso: 25 Febrero 2016].

[21] Oracle, «Oracle Corporation,» [En línea]. Available: http://www.oracle.com/es/products/mysql/index.html. [Último acceso: 25 Febrero 2016].

[22] PHP, «PHP,» [En línea]. Available: http://php.net. [Último acceso: 26 Febrero 2016].

[23] Tiobe, «Índice Tiobe,» [En línea]. Available: http://tiobe.com/tiobe_index/. [Último acceso: 26 Febrero 2016].

[24] PHP, «PHP Manual,» [En línea]. Available: http://php.net/manual/es/history.php.php. [Último acceso: 26 Febrero 2016].

[25] Ecojoven, «Ecojoven RFID,» [En línea]. Available: http://www.ecojoven.com/dos/03/RFID.html. [Último acceso: 02 Marzo 2016].

[26] Ubuntu, «Ubuntu,» [En línea]. Available: http://www.ubuntu.com. [Último acceso: 26 Febrero 2016].

[27] G. González, «blogthinkbig,» [En línea]. Available: http://blogthinkbig.com/una-maquina-virtual-sirve/. [Último acceso: 26 Febrero 2016].

[28] Oracle, «Virtualbox,» [En línea]. Available: https://www.virtualbox.org/. [Último acceso: 26 Febrero 2016].

[29] M. Ángel Álvarez, «Desarrolloweb,» [En línea]. Available: http://www.desarrolloweb.com/articulos/que-es-mvc.html. [Último acceso: 26 Febrero 2016].

[30] Universidad de Alicante, «UA,» [En línea]. Available: http://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html. [Último acceso: 26 Febrero 2016].

[31] Stackoverflow, «Stack Overflow,» [En línea]. Available: http://stackoverflow.com/questions/895171/prevent-users-from-submitting-form-by-hitting-enter. [Último acceso: 02 Marzo 2016].

[32] Barcode Generator, «Barcode Generator,» [En línea]. Available: http://www.barcode-generator.org/. [Último acceso: 2016 Febrero 15].

[33] Sistema Summa, «Códigos de barra con php,» [En línea]. Available: http://sistemasumma.com/2014/12/02/codigos-de-barra-en-pdf-con-php/. [Último acceso: 02 Marzo 2016].

[34] Bootstrap, «Get Bootstrap,» [En línea]. Available: http://getbootstrap.com/ . [Último acceso: 27 Febrero 2016].

[35] W3 Schools, «W3 Schools Bootstrap,» [En línea]. Available: http://www.w3schools.com/bootstrap. [Último acceso: 27 Febrero 2016].

[36] Serratus, «QuaggaJS,» [En línea]. Available: https://serratus.github.io/quaggaJS/. [Último acceso: 02 Marzo 2016].

ANEXO I: COMANDOS A INTRODUCIR PARA CREAR LA BASE DE DATOS

- *mysql -u Astorga -p*

- *xxxxxxxx*

- *Use trabajo;*

- *CREATE TABLE Alumnos (idalumno int, nombre varchar(128), DNI varchar(80), primary key(idalumno)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Profesor (idprofesor int, nombre varchar(128), correo varchar(128), codbar int, primary key(idprofesor)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Asignatura (idasignatura int, nombre varchar(80), curso int, primary key(idasignatura)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Grupos (idgrupo int, nombre varchar(80), codbar int, asignatura int, primary key(idgrupo), foreign key(asignatura) references Asignatura(idasignatura)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Pertenece (idalumno int, idgrupo int, foreign key(idalumno) references Alumnos(idalumno), foreign key(idgrupo) references Grupos(idgrupo)) engine engine=InnoDB default charset=utf8;*

- *CREATE TABLE Imparte (idprofesor int, idasignatura int, foreign key(idprofesor) references Profesor(idprofesor), foreign key(idasignatura) references Asignatura(idasignatura)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Motivo (idmotivo int, nombre varchar(80), primary key(idmotivo)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Novedades (idalumno int, idgrupo int, idprofesor int, idmotivo int, creado timestamp, hora int, doble boolean, foreign key(idalumno) references Alumnos(idalumno), foreign key(idgrupo) references Grupos(idgrupo), foreign key(idprofesor) references Profesor(idprofesor), foreign key(idasignatura) references Asignatura(idasignatura)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Introducidos (id int auto_increment, idgrupo int, primary key(id)) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Guardar (nombrealumno varchar(128), nombregrupo varchar(128), nombreprofesor varchar(128), nombremotivo varchar(80), creado timestamp, hora int, doble boolean) engine=InnoDB default charset=utf8;*

- *CREATE TABLE Administrador (idadiministrador int, nombre varchar(128), primary key(idadministrador)) engine=InnoDB default charset=utf8;*

ANEXO II: HOJA DE ESTILOS CSS

```
body {
background-color: #C6C6C6;
}
h1 {
color:#050A6E;
text-align:center;
}
h3.error {
color:red;
text-align:center;
}
p {
color:#050A6E;
text-align:center;
}
p.error {
color:red;
text-align:center;
}
p.aclaracion {
font-size:11px;
color:#050A6E;
text-align:center;
}
table, tr, td {
color:#050A6E;
text-align:center;
background-color: #EFFFC0;
border:1px solid black;
border-collapse:collapse;
width:60%;
margin:auto;
height:50px;
}
table.interior, tr.interior, td.interior {
color:#050A6E;
text-align:center;
background-color: #EFFFC0;
border:0px solid black;
border-collapse:collapse;
width:100%;
margin:auto;
height:50px;
}
a:link {
color:#07611E;
}
a:visited {
```

```
color:#07611E;
}
a:hover {
color:#610707;
}
a:active {
color:#07611E;
}
```

ANEXO III: INDEX.PHP

```
<?php
session_name("TFG");
session_start();
require ("modelo.php");
?>
<!DOCTYPE html>
<html lang="en">
<html>
<head>
	<meta charset="utf-8">
	<link rel="stylesheet" href="style.css">
	<title>Acceso</title>
</head>
<body>
<h1>Control de Asistencia a clase</h1>
		<br>
<?php
$salir = salir();
$indice = sacarhora();
if ($indice != 0) {
echo "<p class='aclaracion'>Novedad para la $indice hora del día. </p><br>";
}
?>
		<br>
	<form action="control.php" method="post" autocomplete="off">
		<p>Escanee o introduzca su código de profesor: <input type="text" name="idprofesor"
autofocus="autofocus"></p>
		<input type="hidden" name="accion" value="comprobacion">
		<p><input type="submit" name="enviar" value="Entrar"></p>
	</form>
<?php
$accion = $_REQUEST["accion"];
if (isset($accion)) {


if ($accion == "error") {
echo "<h3 class='error'>Se ha producido un error, vuelva a escanear o introducir su código</h3>";
}
if ($accion == "vaciado") {
echo "<p class='error'>Base de datos de Novedades ha sido borrada correctamente</p>";
}
}
?>

<br><br><br><br><br>
	<form action="control.php" method="post" autocomplete="off">
		<p>Acceder como administrador: <input type="text" name="administrador"></p>
		<input type="hidden" name="accion" value="administrador">
```

```
            <p><input type="submit" name="enviar" value="Entrar"></p>
        </form>

</body>
</html>
```

ANEXO IV: VISTA1.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<link rel="stylesheet" href="style.css">
	<title>Bienvenido</title>
</head>
<body>
<?php
$nombreprofesor = $_SESSION["nombreprofesor"];

	echo "<p>Bienvenido, $nombreprofesor</p>";


?>
	<form action="control.php" method="post" autocomplete="off" id="formulario"
onkeypress="return event.keyCode != 13">
		<p><input type="hidden" id="horadoble" name="horadoble" value="">
		<p>Escanee el código del grupo:<input type="text" name="idgrupo"
autofocus="autofocus"><br></p>
		<input type="hidden" name="accion" value="grupo">
		<p><input type="button" id="enviar" name="enviar" value="Aceptar"></p>
	</form>
<br>
<?php
$accion = $_REQUEST["accion"];
if (isset($accion)) {

$count = $_SESSION["count"];

	if ($accion == "introducido") {
	$nombreintroducidos = $_SESSION["nombreintroducidos"];
	echo "<p>Se han introducido correctamente los siguientes grupos:</p>";
	echo "<p class='aclaracion'>(Pinche sobre un grupo para ver la novedad que ha
introducido)</p>";
	foreach ($nombreintroducidos as $row){
		echo "<p><a
href=control.php?accion=listargrupo&grupo=".$row["nombre"].">".$row["nombre"]."</a><br></p>";
	}
	echo "<p>Si desea borrar alguna de las novedades listadas pulse aqui:</p>";
	echo "<form action='control.php' method='post' autocomplete='off'>";
	echo "<p><input type='submit' name='quieroborrar' value='Borrar Novedades'></p>";
	echo "<p><input type='hidden' name='accion' value='iraborrar'></p>";
	echo "</form>";
```

```
        }
        if ($accion == "error") {
        echo "<br> <p class='error'>Se ha producido un Error, vuelva a intentarlo.</p>";
        }
}

?>
        <form action="control.php" method="post">
                <p>Pulse para guardar las novedades introducidas y salir:</p>
                <p><input type="submit" name="salir" value="Guardar y Salir"></p>
                <input type='hidden' name='accion' value='salir'>
        </form>

        <script src="jquery.min.js"></script>
  <script type="text/javascript">
        // Función jQuery para pedir confirmación de si es una hora o dos
        $(document).ready(function(){
      $("#enviar").click(function(){
        // Aparece una ventana de confirmación
        var c = confirm("¿Es una clase de dos horas?");
        // En función del valor del confirm ponemos una hora o dos en el formulario
        if (c) {
                $("#horadoble").attr("value","doble");
        } else {
                $("#horadoble").attr("value","simple");
        }
        // Enviar el formulario al servidor
      $("#formulario").submit();
    });
   });

  </script>
</body>
</html>
```

ANEXO V: VISTA2.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
<html>
<head>
	<meta charset="utf-8">
	<link rel="stylesheet" href="style.css">
	<title>Grupo</title>
</head>
<body>
<?php
$nombreprofesor = $_SESSION["nombreprofesor"];
$nombregrupo = $_SESSION["nombregrupo"];
	echo "<p>Bienvenido, $nombreprofesor <br></p>";
	echo "<p>Introduzca la novedad del grupo $nombregrupo <br><br></p>";


		if (isset($_SESSION["nombrealumnos"])) {
			$nombrealumnos = $_SESSION["nombrealumnos"];
			$numalumno = 0;
			if (count($nombrealumnos) > 0) {
				echo "<form action='control.php' method='post' autocomplete='off'>";
				echo "<table>";
				echo "<tr><td>Nombre</td><td>Novedad</td></tr>";
				foreach ($nombrealumnos as $row) {

					echo "<tr><td>".$row."</td><td>
<select name='novedad$numalumno'>
 <option value='0' selected='selected'>Sin Novedad</option>
 <option value='9001'>Hospitalizado</option>
 <option value='9002'>Revista Médica</option>
 <option value='9003'>Permiso</option>
 <option value='9004'>Comisión</option>
 <option value='9005'>Rebajado en Domicilio</option>
 <option value='9006'>Rebajado de Ejercicio y/o Instrucción</option>
 <option value='9008'>Tutoría</option>
 <option value='9007'>Guardia</option>
 <option value='9009'>Otro motivo</option>
</select>
</td></tr>";
$numalumno = $numalumno + 1;
$_SESSION["numalumno"] = $numalumno;
				}
				echo "</table>";
```

```
                                echo "<p><input type='submit' name='enviar' value='Aceptar
Novedad'></p>";
                                echo "<input type='hidden' name='accion' value='novedades'>";
                                echo "</form>";
                        } else {
                                echo "<h3>Grupo vacio</h3>";
                                $accion = 'error';
                        }
                }
        ?>

<?php
$accion = $_REQUEST["accion"];
if (isset($accion)) {

if ($accion == "error") {
echo "<h3 style='color:red;'>Se ha producido un Error, vuelva a intentarlo.</h3>";
}
}
?>
</body>
</html>
```

ANEXO VI: VISTA3.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<link rel="stylesheet" href="style.css">
	<title>Borrar Novedad</title>
</head>
<body>
<?php
$nombreprofesor = $_SESSION["nombreprofesor"];

	echo "<p>Bienvenido, $nombreprofesor</p>";


?>
	<form action="control.php" method="post" autocomplete="off">
		<p>Escanee el código del grupo a borrar:<input type="text" name="idgrupoaborrar"
autofocus="autofocus"></p>
		<input type="hidden" name="accion" value="borrar">
		<p><input type="submit" name="borrar" value="Borrar Novedad"></p>
	</form>
<br><br>
<form action="control.php" method="post">
		<p><input type="submit" name="borrar" value="Atrás"></p>
		<input type="hidden" name="accion" value="borrar">
	</form>

</body>
</html>
```

ANEXO VII: VISTA4.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<link rel="stylesheet" href="style.css">
	<title>Novedades Grupo</title>
</head>
<body>
<?php
$nombreprofesor = $_SESSION["nombreprofesor"];

	echo "<p>Bienvenido, $nombreprofesor</p>";

?>
<p> A continuación se presenta la novedad que ha sido introducida en el sistema.</p>
<p class='aclaracion'>Si hay algún error vuelva atrás, bórrela y vuelva a introducirla correctamente.
<br> </p>
<br>
<table><tr><td>Nombre</td><td>Novedad</th></tr><tr><td>
<?php
$nombrealumnosgrupo = $_SESSION["nombrealumnosgrupo"];

echo "<table class='interior'>";
foreach ($nombrealumnosgrupo as $row) {
echo "<tr class='interior'><td class='interior'>".$row."</td></tr>";
}
echo "</table>";
?>
</td><td>
<?php
$nombrenovedad = $_SESSION["nombrenovedad"];
echo "<table class='interior'>";
foreach ($nombrenovedad as $row) {
echo "<tr class='interior'><td class='interior'>".$row."</td></tr>";
}
echo "</table>";
?>
</td></tr></table>
	<form action="control.php" method="post">
		<p><input type="submit" name="Volver Atrás" value="volver"></p>
		<input type='hidden' name='accion' value='introducido'>
	</form>
```

```
</body>
</html>
```

ANEXO VIII: VISTA5.PHP

```
<?php
session_name("TFG");
session_start();
require ("modelo.php");
?>
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<link rel="stylesheet" href="style.css">
	<title>Administrador</title>
</head>
<body>
<?php

	echo "<p>Bienvenido</p>";
$descargar = convertirtabla();
	echo "<p>Pulse sobre este link para guardar la Novedad del día en su equipo</p><p><a
href='convertir.php'>Pulse aquí para guardar las novedades completas</a></p><p><a
href='convertirids.php'>Pulse aquí para descargar las novedades por códigos</a></p><p>Una vez
guardada no olvide borrarla de la base de datos para que esté vacía para mañana</p>";

?>
	<form action="control.php" method="post">
		<p>Pulse el botón a continuación para borrar la tabla de Novedades del sistema.</p>
		<p class='aclaracion'>Cuidado, una vez borrada no podra recuperarla, <font
color="red">asegurese de tenerla ya guardada en el sistema</font></p>
		<input type="hidden" name="accion" value="borrarnovedades">
		<p><input type="submit" name="borrar" value="Borrar Novedades"></p>
	</form>
<br><br>
<form action="control.php" method="post">
		<p><input type="submit" name="borrar" value="Salir sin Borrar"></p>
		<input type="hidden" name="accion" value="salirdeadmin">
	</form>

</body>
</html>
```

ANEXO IX: CONTROL.PHP

```
<?php
session_name("TFG");
session_start();

function redirect($url, $statusCode = 303) {
	header('Location: ' . $url, true, $statusCode);
	die();
}

require ("modelo.php");

$accion = $_REQUEST["accion"];

if (isset($accion)) {

	if ($accion == "comprobacion") {
		$idprof = $_REQUEST["idprofesor"];
		$idprofesor = comprobacion($idprof);
		$nombreprofesor = nombreprofesor($idprofesor);
		$_SESSION["nombreprofesor"] = $nombreprofesor;

		if ($idprofesor != NULL) {
			$_SESSION["idprofesor"] = $idprofesor;
			redirect("vista1.php");
		} else {
			redirect("index.php?accion=error");
		}
	}

	if ($accion == "grupo") {
		$idgrupo = $_REQUEST["idgrupo"];
		$_SESSION["idgrupo"] = $idgrupo;
		$horadoble = $_REQUEST["horadoble"];
		$_SESSION["horadoble"] = $horadoble;
		$nombregrupo = nombregrupo($idgrupo);
		$_SESSION["nombregrupo"] = $nombregrupo;
		$idasignatura = sacaridasignatura($idgrupo);
		$_SESSION["idasignatura"] = $idasignatura;
		$nombreasignatura = nombreasignatura($idasignatura);
		$_SESSION["nombreasignatura"] = $nombreasignatura;
		$idalumnos = sacaridalumnos($idgrupo);
		$_SESSION["idalumnos"] = $idalumnos;
		$nombrealumnos = nombrealumnos($idalumnos);
		$_SESSION["nombrealumnos"] = $nombrealumnos;


		if ($nombrealumnos != NULL) {
```

```
                redirect("vista2.php");
            } else {
                redirect("vista1.php?accion=error");
            }
        }

        if ($accion == "novedades") {
            $idalumnos = $_SESSION["idalumnos"];
            $numalumno = $_SESSION["numalumno"];
            $longitud = count($idalumnos);
            $i = 0;
            $count = $_SESSION["count"];
            $novedadi = array();
            for ($i=0; $i<$longitud; $i++){
            $novedadi[$i] = $_REQUEST["novedad$i"];
            }
            for ($i=0; $i<$longitud; $i++){
            $novedad = $novedadi[$i];
            $idalumno = $idalumnos[$i];
            $idgrupo = $_SESSION["idgrupo"];
            $idprofesor = $_SESSION["idprofesor"];
            $horadoble = $_SESSION["horadoble"];
            if (isset($horadoble) && ($horadoble == "doble")) {
                $doble = 1;
            } else {
                $doble = 0;
            }
            $indice = get_time_slot();
            $novedades = guardarnovedad($novedad, $idalumno, $idgrupo, $idprofesor, $doble,
$indice);
            }
            $introducido = introducir($idgrupo);
            $sacarid = sacarid();
            $nombreintroducidos = nombreintroducidos($sacarid);
            $_SESSION["nombreintroducidos"]=$nombreintroducidos;
            redirect("vista1.php?accion=introducido");
        }
        if ($accion == "introducido") {
        redirect("vista1.php?accion=introducido");
        }
        if ($accion == "salir") {
            $salir = salir();
            unset ($_SESSION[""]);
            redirect("index.php");
        }
        if ($accion == "borrar") {
        $idgrup = $_REQUEST["idgrupo"];
        $idgrupo = $_REQUEST["idgrupoaborrar"];
        $idprofesor = $_SESSION["idprofesor"];
        $indice = $indice = get_time_slot();
        $count = $_SESSION["count"];
```

```
        $borrar = borrarnovedad($idprofesor, $idgrupo, $indice);
        $sacarid = sacarid();
        $nombreintroducidos = nombreintroducidos($sacarid);
        $_SESSION["nombreintroducidos"] = $nombreintroducidos;
        redirect("vista1.php?accion=introducido");
        }
        if ($accion == "iraborrar") {
        redirect("vista3.php");
        }

        if ($accion == "listargrupo") {
        $grupo = $_REQUEST["grupo"];
        $indice = get_time_slot();
        $idprofesor = $_SESSION["idprofesor"];
        $idgrupo = sacaridgrupo($grupo);
        $idalumno = sacaridalumnos($idgrupo);
        $nombrealumnosgrupo = nombrealumnos($idalumno);
        $idnovedad = idnovedad($idalumno, $idprofesor, $indice);
        $nombrenovedad = nombrenovedad($idnovedad);
        $_SESSION["grupo"] = $grupo;
        $_SESSION["sacaridgrupo"] = $idgrupo;
        $_SESSION["alumnosgrupo"] = $idalumno;
        $_SESSION["nombrenovedad"] = $nombrenovedad;
        $_SESSION["nombrealumnosgrupo"] = $nombrealumnosgrupo;
        redirect("vista4.php");
        }
        if ($accion == "administrador") {
        $idadmin = $_REQUEST["administrador"];
        $idadministrador = comprobaradministrador($idadmin);

                if ($idadministrador != NULL) {
                        $_SESSION["idadministrador"] = $idadministrador;
                        redirect("vista5.php");
                } else {
                        redirect("index.php?accion=error");
                }
        }
        if ($accion == "borrarnovedades") {
        $borrarnovedad = vaciarnovedades();
        redirect("index.php?accion=vaciado");
        }
        if ($accion == "salirdeadmin") {
        redirect("index.php");
        }
}


?>
```

ANEXO X: MODELO.PHP

```
<?php
session_name("TFG");
session_start();

function comprobacion($idprof) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
	$sql = "SELECT idprofesor FROM Profesor WHERE idprofesor=$idprof";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0) {
		while($row = mysqli_fetch_assoc($result)) {
			$idprofesor = $row["idprofesor"];
		}
		return $idprofesor;
	} else {
		return NULL;
	}
	mysqli_close($conn);
}

function comprobaradministrador($idadmin) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
	$sql = "SELECT idadministrador FROM Administrador WHERE idadministrador=$idadmin";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0) {
		while($row = mysqli_fetch_assoc($result)) {
			$idadministrador = $row["idadministrador"];
		}
		return $idadministrador;
	} else {
		return NULL;
	}
```

```
	mysqli_close($conn);
}

function nombreprofesor($idprofesor) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "SELECT nombre FROM Profesor WHERE idprofesor=$idprofesor";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0) {
		while($row = mysqli_fetch_assoc($result)) {
			$nombreprofesor = $row["nombre"];
		}
	return $nombreprofesor;
	}
	mysqli_close($conn);
}

function nombregrupo($idgrupo) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "SELECT nombre FROM Grupos WHERE idgrupo=$idgrupo";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0) {
		while($row = mysqli_fetch_assoc($result)) {
			$nombregrupo = $row["nombre"];
		}
	return $nombregrupo;
	}
	mysqli_close($conn);
}

function sacaridasignatura($idgrupo) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
```

```
	$dbname = "trabajo";
	$idasignatura = array();
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "SELECT idasignatura FROM Grupos WHERE idgrupo=$idgrupo";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0) {
		while($row = mysqli_fetch_assoc($result)) {
			$idasignatura = $row["idasignatura"];
		}
	return $idasignatura;
	}
	mysqli_close($conn);
}

function nombreasignatura($idasignaturas) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$nombreasignatura = array();
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "SELECT nombre FROM Asignatura WHERE idasignatura=$idasignaturas";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0){
		while($row = mysqli_fetch_assoc($result)){
			$nombreasignatura[] = $row["nombre"];
		}

		return $nombreasignatura;
	}
	mysqli_close($conn);
}

function sacaridalumnos($idgrupo) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$idalumno = array();
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
```

```
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "SELECT idalumno FROM Pertenece WHERE idgrupo=$idgrupo";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0){
		while($row = mysqli_fetch_assoc($result)){
			$idalumno[] = $row["idalumno"];
		}

		return $idalumno;
	}
	mysqli_close($conn);
}

function nombrealumnos($idalumnos) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$nombrealumno = array();
	$longitud = count($idalumnos);
	$i = 0;

	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
	for ($i=0; $i<=$longitud; $i++){
	$idalumno = $idalumnos[$i];
	$sql = "SELECT nombre FROM Alumnos WHERE idalumno=$idalumno";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$nombrealumno[] = $row["nombre"];
			}
		}
	}
		return $nombrealumno;

	mysqli_close($conn);
}

function guardarnovedad($novedad, $idalumno, $idgrupo, $idprofesor, $doble, $indice) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
```

```
	$longitud = count($idalumnos);
	$i = 0;
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "INSERT INTO Novedades (idalumno, idmotivo, idgrupo, idprofesor, hora, doble)
VALUES (?, ?, ?, ?, ?, ?)";
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_bind_param($stmt, "ssssss", $idalumno, $novedad, $idgrupo, $idprofesor,
$indice, $doble);
	mysqli_stmt_execute($stmt);

	mysqli_close($conn);

}

function introducir($idgrupo){
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$introducido = array();
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
	$sql = "INSERT INTO Introducidos (idgrupo) VALUES (?)";
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_bind_param($stmt, "s", $idgrupo);
	mysqli_stmt_execute($stmt);
	mysqli_close($conn);
}

function sacarid(){
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$sacarid = array();
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
	$sql = "SELECT idgrupo FROM Introducidos";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0){
```

```
                while($row = mysqli_fetch_assoc($result)){
                        $sacarid[] = $row["idgrupo"];
                }

                return $sacarid;
        }
        mysqli_close($conn);
}

function nombreintroducidos($sacarid){
        $servername = "localhost";
        $username = "Astorga";
        $password = "xxxxxxx";
        $dbname = "trabajo";
        $longitud = count($sacarid);
        $nombreintroducidos = array();
        $conn = mysqli_connect($servername, $username, $password, $dbname);
        if (!$conn) {
                die("Conexión fallida:".mysqli_connect_error());
        }
        mysqli_set_charset($conn, "utf8");
        for ($i=0; $i<$longitud; $i++){
        $sacada = $sacarid[$i];
        $sql = "SELECT nombre FROM Grupos WHERE idgrupo='$sacada'";
        $result = mysqli_query ($conn, $sql);
        if (mysqli_num_rows($result) > 0){
                while($row = mysqli_fetch_assoc($result)){
                        $nombreintroducidos[$i] = $row;
                }
        }
        }
                return $nombreintroducidos;



        mysqli_close($conn);
}

function borrarnovedad($idprofesor, $idgrupo, $indice) {
        $servername = "localhost";
        $username = "Astorga";
        $password = "xxxxxxx";
        $dbname = "trabajo";
        $i = 0;
        $conn = mysqli_connect($servername, $username, $password, $dbname);
        if (!$conn) {
                die("Conexión fallida:".mysqli_connect_error());
        }
        mysqli_set_charset($conn, "utf8");

        $sql = "DELETE FROM Novedades WHERE (idprofesor, idgrupo, hora) = (?, ?, ?)";
```

```
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_bind_param($stmt, "sss", $idprofesor, $idgrupo, $indice);
	mysqli_stmt_execute($stmt);

	$sql = "DELETE FROM Introducidos WHERE (idgrupo) = (?)";
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_bind_param($stmt, "s", $idgrupo);
	mysqli_stmt_execute($stmt);
	mysqli_close($conn);

}

function salir() {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "DELETE FROM Introducidos";
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_execute($stmt);
	mysqli_close($conn);
}

function sacaridgrupo($grupo) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "SELECT idgrupo FROM Grupos WHERE nombre='$grupo'";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0) {
		while($row = mysqli_fetch_assoc($result)) {
			$sacaridgrupo = $row["idgrupo"];
		}
	return $sacaridgrupo;
	}
	mysqli_close($conn);
}
```

```
function idnovedad($idalumno, $idprofesor, $indice) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$idnovedad = array();
	$longitud = count($idalumno);
	$i = 0;

	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
	for ($i=0; $i<=$longitud; $i++){
	$alumno = $idalumno[$i];
	$sql = "SELECT idmotivo FROM Novedades WHERE (idalumno, idprofesor, hora) =
($alumno, $idprofesor, $indice)";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$idnovedad[] = $row["idmotivo"];
			}
		}
	}
		return $idnovedad;

	mysqli_close($conn);
}

function nombrenovedad($idnovedad) {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$nombrenovedad = array();
	$longitud = count($idnovedad);
	$i = 0;

	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
	for ($i=0; $i<=$longitud; $i++){
	$novedad = $idnovedad[$i];
	$sql = "SELECT nombre FROM Motivo WHERE idmotivo=$novedad";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){
```

```
                        while($row = mysqli_fetch_assoc($result)){
                                $nombrenovedad[] = $row["nombre"];
                        }
                }
        }
                return $nombrenovedad;

        mysqli_close($conn);
}

function sacarhora() {
  date_default_timezone_set("Europe/Madrid");


  $horas[1]["begin"] = date('H:i', strtotime("08:30"));
  $horas[1]["end"] = date('H:i', strtotime("09:20"));
  $horas[2]["begin"] = date('H:i', strtotime("09:25"));
  $horas[2]["end"] = date('H:i', strtotime("10:15"));
  $horas[3]["begin"] = date('H:i', strtotime("10:35"));
  $horas[3]["end"] = date('H:i', strtotime("11:25"));
  $horas[4]["begin"] = date('H:i', strtotime("11:30"));
  $horas[4]["end"] = date('H:i', strtotime("12:20"));
  $horas[5]["begin"] = date('H:i', strtotime("12:25"));
  $horas[5]["end"] = date('H:i', strtotime("13:15"));
  $horas[6]["begin"] = date('H:i', strtotime("13:30"));
  $horas[6]["end"] = date('H:i', strtotime("14:20"));
  $horas[7]["begin"] = date('H:i', strtotime("15:55"));
  $horas[7]["end"] = date('H:i', strtotime("16:45"));
  $horas[8]["begin"] = date('H:i', strtotime("16:50"));
  $horas[8]["end"] = date('H:i', strtotime("17:40"));

  $now = date('H:i');


  $indice = 0;
  for ($i=1; $i <= 8; $i++) {
    if (($now >= $horas[$i]["begin"]) && ($now <= $horas[$i]["end"])) {
      $indice = $i;
    }
  }
  return $indice;
}

function vaciarnovedades() {
        $servername = "localhost";
        $username = "Astorga";
        $password = "xxxxxxx";
        $dbname = "trabajo";
        $i = 0;
        $conn = mysqli_connect($servername, $username, $password, $dbname);
```

```
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");

	$sql = "DELETE FROM Novedades";
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_execute($stmt);

	$sql = "DELETE FROM Guardar";
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_execute($stmt);

	mysqli_close($conn);
}

function convertirtabla() {
	$servername = "localhost";
	$username = "Astorga";
	$password = "xxxxxxx";
	$dbname = "trabajo";
	$idmotivos = array();
	$nombremotivos = array();
	$idgrupos = array();
	$nombregrupos = array();
	$idalumnos = array();
	$nombrealumnos = array();
	$idprofesores = array();
	$nombreprofesores = array();
	$horas = array();
	$dobles = array();
	$creados = array();
	$i = 0;
	$conn = mysqli_connect($servername, $username, $password, $dbname);
	if (!$conn) {
		die("Conexión fallida:".mysqli_connect_error());
	}
	mysqli_set_charset($conn, "utf8");
//Sacar idalumnos

	$sql = "SELECT idalumno FROM Novedades";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$idalumnos[] = $row["idalumno"];
			}
		}

//Sacar nombrealumnos
	$longitud = count($idalumnos);
```

```
	for ($i=0; $i<=$longitud; $i++){
	$idalumno = $idalumnos[$i];
	$sql = "SELECT nombre FROM Alumnos WHERE idalumno=$idalumno";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$nombrealumnos[] = $row["nombre"];
			}
		}
	}
//Sacar idgrupos
	for ($i=0; $i<=$longitud; $i++){
	$idalumno = $idalumnos[$i];
	$sql = "SELECT idgrupo FROM Novedades WHERE idalumno=$idalumno";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$idgrupos[] = $row["idgrupo"];
			}
		}
	}

//Sacar nombregrupos
	$longitud = count($idgrupos);
	for ($i=0; $i<=$longitud; $i++){
	$idgrupo = $idgrupos[$i];
	$sql = "SELECT nombre FROM Grupos WHERE idgrupo=$idgrupo";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$nombregrupos[] = $row["nombre"];
			}
		}
	}

//Sacar idmotivos
	for ($i=0; $i<=$longitud; $i++){
	$idalumno = $idalumnos[$i];
	$sql = "SELECT idmotivo FROM Novedades WHERE idalumno=$idalumno";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$idmotivos[] = $row["idmotivo"];
			}
		}
	}
```

```
//Sacar nombremotivos
	$longitud = count($idmotivos);
	for ($i=0; $i<=$longitud; $i++){
	$idmotivo = $idmotivos[$i];
	$sql = "SELECT nombre FROM Motivo WHERE idmotivo=$idmotivo";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$nombremotivos[] = $row["nombre"];
			}
		}
	}
//Sacar idprofesores
	for ($i=0; $i<=$longitud; $i++){
	$idalumno = $idalumnos[$i];
	$sql = "SELECT idprofesor FROM Novedades WHERE idalumno=$idalumno";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$idprofesores[] = $row["idprofesor"];
			}
		}
	}
//Sacar nombreprofesores
	$longitud = count($idprofesores);
	for ($i=0; $i<=$longitud; $i++){
	$idprofesor = $idprofesores[$i];
	$sql = "SELECT nombre FROM Profesor WHERE idprofesor=$idprofesor";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$nombreprofesores[] = $row["nombre"];
			}
		}
	}
//Sacar horas

	$sql = "SELECT hora FROM Novedades";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$horas[] = $row["hora"];
			}
		}
//Sacar dobles

	$sql = "SELECT doble FROM Novedades";
```

```
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$dobles[] = $row["doble"];
			}
		}
//Sacar creados

	$sql = "SELECT creado FROM Novedades";
	$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0){

			while($row = mysqli_fetch_assoc($result)){
				$creados[] = $row["creado"];
			}
		}
//Meter en tabla Guardar
	$longitud = count($nombrealumnos);
	for ($i=0; $i<$longitud; $i++) {
	$sql = "INSERT INTO Guardar (nombrealumno, nombremotivo, nombregrupo,
nombreprofesor, creado, hora, doble) VALUES (?, ?, ?, ?, ?, ?, ?)";
	$stmt = mysqli_prepare($conn, $sql);
	mysqli_stmt_bind_param($stmt, "sssssss", $nombrealumnos[$i], $nombremotivos[$i],
$nombregrupos[$i], $nombreprofesores[$i], $creados[$i], $horas[$i], $dobles[$i]);
	mysqli_stmt_execute($stmt);
	}
	mysqli_close($conn);

}
?>
```

ANEXO XI: CONVERTIR.PHP

```
<?php
header("Content-type: application/vnd.ms-excel" ) ;
header("Content-Disposition: attachment; filename=novedad.xls" ) ;
$servidor="localhost";
$user="Astorga";
$pass="xxxxxxx";
$db="trabajo";
$tabla="Guardar";
mysql_connect($servidor,$user,$pass) ;
mysql_select_db($db) ;
$qry=mysql_query("select * from $tabla" ) ;
$campos = mysql_num_fields($qry) ;
$i=0;
echo "<table><tr>";
while($i<$campos){
echo "<td>". mysql_field_name ($qry, $i) ;
echo "</td>";
$i++;
}
echo "</tr>";
while($row=mysql_fetch_array($qry)){
echo "<tr>";
for($j=0; $j<$campos; $j++) {
echo "<td>".$row[$j]."</td>";
}
echo "</tr>";
}
echo "</table>";
?>
```

ANEXO XII: CONVERTIRIDS.PHP

```
<?php
header("Content-type: application/vnd.ms-excel" ) ;
header("Content-Disposition: attachment; filename=novedad.xls" ) ;
$servidor="localhost";
$user="Astorga";
$pass="xxxxxxx";
$db="trabajo";
$tabla="Novedades";
mysql_connect($servidor,$user,$pass) ;
mysql_select_db($db) ;
$qry=mysql_query("select * from $tabla" ) ;
$campos = mysql_num_fields($qry) ;
$i=0;
echo "<table><tr>";
while($i<$campos){
echo "<td>". mysql_field_name ($qry, $i) ;
echo "</td>";
$i++;
}
echo "</tr>";
while($row=mysql_fetch_array($qry)){
echo "<tr>";
for($j=0; $j<$campos; $j++) {
echo "<td>".$row[$j]."</td>";
}
echo "</tr>";
}
echo "</table>";
?>
```

ANEXO XIII: TABLAS EXCEL DE LA BASE DE DATOS PARA LA REALIZACIÓN DE PRUEBAS DE VALIDACIÓN

PROFESOR

IDPROFESOR	NOMBRE	CODBAR	CORREO
1	Gonzalo García Yáñez	1	garcía@ejemplo.com
2	Eugenia Rodríguez Alonso	2	rodriguez@ejemplo.com
3	Gracia Villalba García	3	villalba@ejemplo.com
4	Francisco Polo Hernández	4	polo@ejemplo.com
5	Miriam Vila Rodríguez	5	vila@ejempo.com
6	Alberto Rial Freire	6	rial@ejemplo.com
7	Sofía Dapena Iglesias	7	dapena@ejemplo.com
8	Ernesto Miguélez Cuenca	8	miguelez@ejemplo.com
9	Ignacio Pascual Cerrada	9	pascual@ejemplo.com
10	Claudia Mateos Cruz	10	mateos@ejemplo.com

ASIGNATURA

IDASIGNATURA	NOMBRE	CURSO
1	Informática	5
2	Cálculo	5
3	Materiales	5
4	Estadística	5
5	Máquinas	4
6	Maniobra	4
7	Termodinámica	4
8	Medioambiente	3
9	Formación Física	3
10	Navegación	3

MOTIVO

IDMOTIVO	NOMBRE
0	Sin Novedad
9001	Hospitalizado
9002	Revista Médica
9003	Permiso
9004	Comisión
9005	Rebajado en Domicilio
9006	Rebajado en Ejercicio/Instrucción
9007	Guardia
9008	Tutoría
9009	Otro

ALUMNOS		
IDALUMNO	NOMBRE	DNI
1	Luis Astorga Boccherini	53199958S
2	Carlos Arenas Pérez-Seoane	54879537T
3	Carlos Martínez de Baños Martínez de Morentín	54879324O
4	Gerardo González-Aller Díaz del Río	12349322H
5	Carlos Bonaplata Hernández de Armijo	12559687M
6	Santiago Carlos Bausá Viseras	24651525Y
7	Carlos Eustaquio Gallego Montero	36952712A
8	Jose Luis Burrueco Jiménez	08745188R
9	Guillermo Busto Cuiñas	02587557T
10	Ignacio Rodrigáñez Ribas	6701239W
11	Juan Antonio Vidal Sáchez	15436985K
12	Gonzalo Peñuelas Pérez-Cuadrado	98326547L
13	Guillermo Gilabert Gamoa	12875438O
14	Cristina Martínez Merello Graña	19635874N
15	Juan de la Torre Díaz	986312547E
16	Enrique de la Torre Martínez	00215478D
17	Pablo Vega Vegas	95162384X
18	Óscar Vega Vegas	32587461C
19	Miguel Ruiz García	95147854P
20	Miguel Ruiz García	63635958U
21	Pedro López Montoya	51448445G
22	Jose Javier Hernández Serrano	94541587D
23	Jose María Núñez Urdiales	98774896R
24	Gonzalo Gestoso Rodríguez	85484744E
25	Francisco García Ruiz	99632254T
26	Antonio Piñeiro Filgueira	12144425U
27	Luis Carlos Figueirido Filgueira	14522320Y
28	Miguel Mata Peña	99500215J
29	Rubén Magadán Tomás	11215436G
30	Carlos González Rubio	95632505F
31	Pedro Casado Miranda	00216508Q
32	Luis Felipe de la Puente	95515628Z
33	Pablo Sánchez Quemado	97543219N
34	Sergio Núñez Reus	06999085R
35	Borja Molina Norato	16498873M
36	Eladio Almansa García	97415895P
37	Blanca Romero Huerta	17994361O
38	Jaime González Oria	42158965Y
39	Marcos Rodríguez Sánchez	67558031K
40	Pablo Rial Astorga	96548875C
41	Federico Luján Cuesta	89421814D
42	Paloma Hernández de la Flor	98741547V
43	Ricardo Delgado Pérez	98465487R

GRUPOS

IDGRUPO	NOMBRE	CODBAR	IDASGIN
1	InformáticaG1	1	1
2	InformáticaG2	2	1
3	InformáticaREP1	3	1
4	InformáticaCON1	4	1
5	CálculoREP2	5	2
6	CálculoG1	6	2
7	CálculoG2	7	2
8	CálculoIM1	8	2
9	CálculoREP1	9	2
10	MaterialesG1	10	3
11	EstadísticaIM1	11	4
12	MaterialesG2	12	3
13	MaterialesIM1	13	3
14	Medioambiente1	14	8
15	FormaciónFísicaG1	15	9
16	NavegaciónG1	16	10
17	MáquinasG4	17	5
18	ManiobraG4	18	6
19	TermodinámicaG4	19	7
20	TermodinámicaREP1	20	7
21	TermodinámicaCON1	21	7
22	TermodinámicaREP2	22	7
23	ManiobraCON1	23	6
24	ManiobraREP2	24	6

ANEXO XIV: TARJETAS DE PROFESOR Y FICHA DE CLASE

TARJETA DE PROFESOR	TARJETA DE PROFESOR
Gonzalo García Yáñez	Eugenia Rodríguez Alonso
 00000001	 00000002

FICHA DE CLASE.

CURSO: 5º CUERPO GENERAL GRUPO G1

Informática	Cálculo
 00000001	 00000006
Materiales	
 00000010	