Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERIA EN TECNOLOGÍA DE TELECOMUNICACIONES

TRABAJO FIN DE GRADO

IMPLEMENTACIÓN, ADAPTACIÓN Y REDISEÑO DE UNA APLICACIÓN WEB

DOCUMENTO – *MEMORIA*

Alumno/Alumna: Fernández, Guerrero, Pedro

Director/Directora: Ferro, Vázquez, Armando

Curso: 2019-2020

Fecha: Bilbao, 18, Julio, 2020

RESUMEN

Las aplicaciones Web ofrecen a los usuarios distintos tipos de utilidades. Para que una aplicación sea práctica es necesario que cumpla unos requisitos mínimos en lo que se refiere a la optimización de velocidad de servicio y diseño de una interfaz cómoda. Para ello, la implementación de dichas aplicaciones juega un papel fundamental, así como el diseño de las mismas dentro del cumplimiento de estos aspectos. Para favorecer el acceso a la información que ofrecen es imprescindible una buena adaptabilidad a cualquier dispositivo del que disponga el usuario. Todos estos temas serán analizados a lo largo de este documento a partir de un código ya facilitado.

ABSTRACT

Web applications offer users different types of utilities. For an application to be practical, it needs to meet minimum requirements for service speed optimization and the design of a good interface that allows the user to get the information in a comfortable way. To do this, the implementation of these applications plays a fundamental role, as well as the design of them within the fulfillment of these aspects. To promote access to the information they offer, good adaptability is essential to any device available to the user. All these topics will be analyzed throughout this document from an already provided code.

LABURPENA

Web-aplikazioek hainbat motatako utilitateak eskaintzen dizkiete erabiltzaileei. Aplikazio bat praktikoa izan dadin, gutxieneko baldintza batzuk bete behar dira zerbitzu-abiadura optimizatzeari eta interfaze eroso bat diseinatzeari dagokienez. Horretarako, aplikazio horiek inplementatzea funtsezkoa da, bai eta horiek diseinatzea ere, alderdi horiek betez. Eskaintzen duten informaziorako sarbidea errazteko, ezinbestekoa da erabiltzaileak duen edozein gailutara ondo egokitzea. Gai horiek guztiak dokumentu honetan aztertuko dira, emandako kode batean oinarrituta.

PALABRAS CLAVE

- Aplicación
- Implementación
- Servidor
- Etiqueta
- Interfaz

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

MEMORIA

1 Introducción

La implementación de un software por medio de un servidor es un paso imprescindible para ofrecer las ventajas de una aplicación previamente diseñada a cualquier usuario en cualquier parte. Dicha implementación puede variar en función de aspectos como el lenguaje de programación utilizado, el método de comunicación de dicha aplicación con el servidor y por supuesto, del tipo servidor que se decida emplear. Este proceso tendrá como objetivo garantizar una buena comunicación con la aplicación diseñada para que la entrega de información y servicios sea lo más eficaz posible.

Otro factor a tener en cuenta dentro de la optimización de la aplicación web es la reducción de los costes de alojamiento. Esto supone una bajada en los tiempos de carga de la web lo que ofrece al cliente una conexión más rápida y eficiente. El lenguaje *Cascading Style Sheet* (CSS) toma parte dentro de esta optimización mediante la reducción del código. Su función principal es separar el diseño gráfico del *script* logrando así, una disminución del tamaño del código realizado. Esto se traduce en dichas mejoras descritas entre otras secundarias que se mencionaran en el documento.

Estas ventajas tendrán mayor impacto en la aplicación web si se facilita al usuario un acceso desde cualquier dispositivo sin perder la comodidad que ofrece un buen diseño web en el ordenador. Por esta razón, también será importante que el trasvase de la estructura de la página web entre dispositivos se realice de la forma adecuada y que la interfaz mantenga su accesibilidad sea cual sea el origen de ejecución.

2 Contexto

La implementación de la aplicación web que se va a tratar se realizará sobre un servidor Apache utilizando el sistema operativo de Linux. Este software ha liderado el mercado durante casi dos décadas y aunque debido a la gran cantidad de alternativas que han ido surgiendo ha perdido algo de popularidad, continúa siendo uno de los servidores web de código abierto más conocidos hoy en día dentro de este ámbito. Su crecimiento se debió a unas altas prestaciones de funcionamiento y rendimiento y aunque hoy en día existe alguna opción más robusta, a nivel de Linux Apache sigue teniendo un mayor apoyo de la comunidad de código abierto, lo cual lo hace ideal para proyectos particulares. Además, las exigencias de la web a implementar no son altas lo cual da mayor libertad a la hora de elegir el software a utilizar y la flexibilidad de Apache lo convierte en una opción optima.

En el apartado de la codificación del diseño gráfico el lenguaje que se empleará en el diseño de la aplicación web de este proyecto será CSS que interactúa con HTML. Hace años que no se realizan mezclas entre el diseño y codificación de la aplicación dentro del mismo *script* salvo casos de páginas extremadamente simples y, aun así, no es lo

más conveniente. Para solucionar los problemas provocados por esta unión de códigos surge CSS. Su tarea es la de separar ambas partes en dos *scripts* diferentes e interaccionando con HTML. De esta forma se reduce la carga de trabajo de los desarrolladores gráficos de páginas web y con ello se reducen los costes del proceso. En la actualidad, desde el punto de vista del programador, aunque haya algún otro lenguaje para funciones graficas parecidas, CSS se postula como la única opción real en su formato de hoja de estilo.

3 Objetivos y Alcance

El objetivo principal de este proyecto es la puesta en marcha de una aplicación web facilitada que realiza un seguimiento del trabajo de los alumnos a lo largo del curso. Para ello habrá que implementarla con un servidor determinado en el localhost para realizar las pruebas de la aplicación. Posteriormente se realizará el rediseño de la misma. Seguido de esto se pretende acomodar la interfaz de dicha aplicación a cualquier dispositivo para ofrecer el acceso desde móvil sin que afecte al diseño de la página y al contenido de la misma.

Las características que se buscan con la puesta en marcha de este proyecto son las siguientes:

- Implementación de la página web que garantice una comunicación optima con el usuario de la aplicación. Evitando fallos de conexión para su puesta en activo.
- Rediseño de la página con el objetivo de renovar y mejorar la interfaz original, ofreciendo un diseño fácil de asimilar por parte del usuario. De esta forma evitar una interacción complicada a la hora de navegar a través de ella.
- Adaptación de la página web a los dispositivos móviles con el fin de facilitar el acceso a dicha página a aquellos que soliciten sus servicios desde cualquier parte.
- Por último, puesta en marcha de la página web para que los usuarios que comiencen el curso puedan monitorizar su trabajo a lo largo del mismo.

4 Beneficios del trabajo

Como se ha mencionado el trabajo se basa en la implementación, rediseño y adaptación a interfaz móvil de una aplicación web. Con ello se pretende establecer las bases para realizar aplicaciones web con diseños y adaptaciones más complicadas con muchas más páginas de código con las que interactuar y modificaciones de estilo más elaboradas.

Se busca mejorar la eficiencia respecto al modelo anterior y establecer las bases para un incremento en el rendimiento y seguridad, así como capacidad de mejorar la aplicación en futuras remodelaciones de la misma.

5 Descripción de requerimientos

5.1 Descripción de la aplicación

La aplicación que se va a implementar es un tipo de base de datos donde se almacena a los participantes de una asignatura, profesores y alumnos, y se les va haciendo un seguimiento de las prácticas y los exámenes realizados para poder monitorizar el trabajo que se va realizando.

El objetivo a futuro es implementar un método de autenticación basado en formulario donde tanto alumnos como profesores tienen acceso a la aplicación, aunque a cada uno se les otorgara unos permisos diferentes. Los alumnos no podrán acceder a la lista donde se encuentran todas las personas que están matriculadas, sino que solo tendrán acceso a su propio trabajo mientras que los profesores tendrán acceso a toda la información de los participantes.

5.2 Servidor Web

En el desarrollo y puesta en marcha del proyecto es indispensable la utilización del servidor web para implementar el protocolo "*Hypertext Transfer Protocol*" (HTTP). El servidor web es un software que se instala en el ordenador con motivo de mantenerlo permanentemente activo en espera de peticiones de usuarios externos. Es importante diferenciar lo que es un servidor web con un servidor o *server* dado que, aunque ambos están estrechamente relacionados, el servidor web es el programa o *software* que forma parte de dicho servidor y este representa la máquina en la que son alojados todos estos servicios y aplicaciones.

El proceso de solicitar información comienza cuando los usuarios realizan las peticiones a través de un navegador web y realizará una búsqueda DNS. El servidor web permanece en espera de estas peticiones HTTP de los clientes y su trabajo será servir el contenido solicitado actuando como intermediario con estas máquinas de los distintos clientes través del navegador. El objetivo de esta función es procesar la aplicación que se encuentra en el lado del servidor para que ésta tenga a su vez acceso a servidores físicos. Esto le permite acceder a archivos que se encuentran en los mismos y después

se generará una respuesta en el lado del cliente que se ha comentado. Los contenidos devueltos por el servidor suelen ser interpretados por los navegadores web de los clientes, que los muestran por pantalla acorde a unas fuentes, colores, disposición de textos y objetos, por lo que el servidor se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de ella.

En este proyecto se va a realizar sobre Apache HTTP que es uno de los servidores más utilizados para la implementación de páginas web por su confiabilidad, sencillez y versatilidad. Un matiz importante que hay que tener en cuenta es que Apache tiene una estructura basada en módulos lo que le convierte en un servidor con una alta capacidad de personalización dado que permiten a los administradores tener el control de distintas funcionalidades adicionales y activarlas o desactivarlas en función de las diversas necesidades de cada uno. Por lo que su utilización dentro de este proyecto está perfectamente justificada, a pesar del mejor rendimiento de otros *softwares*, puesto que se trata de un servidor bastante confiable. Aun así, su rendimiento es bastante satisfactorio, aunque puede provocar algún problema en caso de páginas con demasiado tráfico. Debido a su arquitectura basada en procesos gasta RAM y CPU de forma innecesaria dado que genera un hilo por cada solicitud. Esto provoca sobrecarga cuando la tasa es alta. No obstante, teniendo en cuenta que el tráfico de la aplicación a implementar es relativamente bajo, esto no supone un problema real para este proyecto.

Apache no se trata de un servidor físico en sí, sino que es un software que se ejecuta en un servidor. Como su propio nombre indica Apache HTTP proporciona una comunicación HTTP entre cliente y servidor, cuya garantía de fiabilidad y seguridad en esta comunicación tendrá que estar garantizada por el propio software.

5.3 Sistemas de comunicación con el servidor

5.3.1 Lado del cliente

La petición de datos y recursos se realiza mediante el protocolo HTTP mediante el esquema que se muestra en la figura. Estos recursos pueden ser por ejemplo documentos HTML. Es la base de cualquier intercambio de datos en la web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador web, y nunca por el servidor. Así, una página web completa resulta de la unión de distintos documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web como CSS, el texto, las imágenes, scripts, etc...

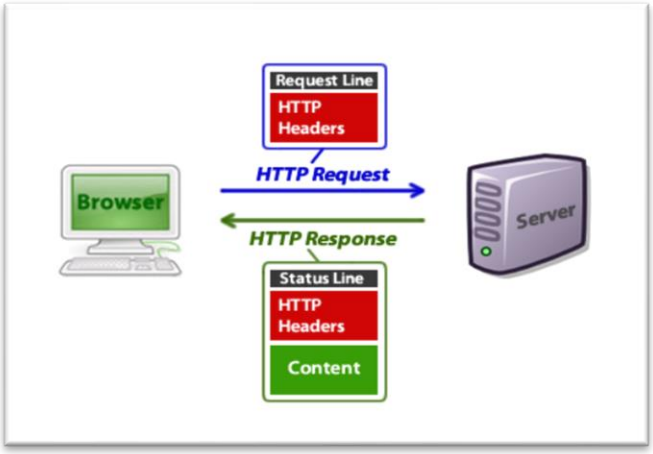


Figura 5.1 Esquema de comunicación HTTP

La arquitectura de este protocolo se basa en el principio cliente-servidor donde las solicitudes de información son enviadas por un agente usuario por medio del navegador web. Cada petición individual se envía a un servidor, el cual la gestiona y responde. Entre cada *petición* y *respuesta*, hay intermediarios, denominados proxies, que cumplen distintas funciones como la de *gateways* o caché. Además de proxies, hay mas elementos intermedios como *routers* aunque estos son transparentes al servidor y al cliente debido a la arquitectura en capas de la web ya que el protocolo HTTP se apoya en las capas de red y transporte.

Como se ha mencionado, el cliente iniciará la comunicación realizando una petición al servidor. En el caso de una página web, se enviará una petición de documento HTML. Cuando procesa el documento continúa mandando más peticiones como scripts u hojas de estilos para posteriormente unirlos y componer la página web. Se pueden realizar más peticiones de datos a medida que el navegador ejecuta los scripts puesto que es posible

que dentro de la misma haya enlaces a los que puede accederse. En consecuencia, se puede decir que el navegador actualizará y gestionará la página web.

5.3.2 Lado del servidor

Al otro lado del sistema de comunicación se encuentra el servidor. Como ya se ha comentado, este es el encargado de servir los datos que el cliente solicita por medio de peticiones. Se ve como a un único equipo, aunque puede estar formado por varios elementos que, por ejemplo, se reparten la carga de solicitudes o gestionan bases de datos.

Según su función se dividen en dos grupos:

- Servidor dedicado: Gestionan únicamente las peticiones de la red, dedicando toda su funcionalidad a la red.
- Servidor no dedicado: Comparte la potencia entre los clientes y las operaciones locales que se puedan desarrollar dentro del equipo local.

Por otro lado, para obtener la información de la aplicación que le ha solicitado el cliente se hace uso de una *Application Programming Interface* (API) web. Esto permite a los servidores web una mayor interactuación con aplicaciones de servidor para realizar distintas interacciones como un *Log in* de usuarios sin necesidad de conocer lo que hay detrás. Cada recurso que se consulta tiene un identificador único, que será un *uniform resource identifier* (URI). Este recurso es consultado por medio de la Uniform Resource Locator (URL) que es una URI que apunta a recursos, los cuales pueden variar en el tiempo.

Esta información se puede enviar mediante GET y POST. El primero de ellos envía la URL con separación de los campos mediante signos de interrogación '?'. El otro método envía directamente la información al servidor haciéndola invisible en la URL, lo que permite mayor cantidad de información. El GET no permitirá grandes cantidades de información, pero a cambio simplifica bastante el *script* utilizando solo una URL.

5.4 Separación del diseño y contenido

La separación del diseño gráfico y el contenido del código se debe a unas determinadas necesidades para los programadores. Es decir, no es algo que se haga porque sí, sino que responde a una serie de razones concretas.

Primero de todo se va a definir a que se refiere con contenido y diseño. El contenido es aquello que define el funcionamiento de la aplicación, así como los apartados e información que la complementan. La presentación o diseño hace referencia a como se

muestra esta estructura a los usuarios de la aplicación, puede ser por ejemplo que las imágenes tengan borde, el tamaño del texto mostrado o la posición de los elementos dentro del layout de la página entre otras muchas cosas.

Cuando ambas partes se juntan pueden existir diferentes inconvenientes de programación. Si por ejemplo en algún momento el desarrollador decide realizar algún cambio de estilo de la página web, el hecho de que diseño y contenido estén dentro de la misma página lo obliga a editar cada uno de los archivos donde se encuentre dicha especificación. Esto provoca que en sitios web con muchas páginas un cambio simple pueda conllevar mucho tiempo de edición. Estos problemas sugieren la necesidad de un sistema que optimice todo el proceso y es ahí donde surge el lenguaje CSS que tiene como objetivo reducir el coste del proceso que conlleva realizar todos estos cambios de diseño dentro de una web cuyo contenido y diseño esta englobado dentro de un mismo HTML. El objetivo principal del lenguaje CSS es la separación del contenido y el diseño o presentación del mismo. Esta separación se debe a unas necesidades concretas para los programadores y es la razón de ser de dicho código. Es decir, no es algo que se haga porque sí, sino que responde a una serie de razones concretas.

El lenguaje CSS se trata de una hoja de estilo que, como se ha mencionado, separa el contenido de la presentación. CSS son las siglas del inglés “Cascading Style Sheets” que se traduce como “hojas de estilo en cascada”. Esto quiere decir que su código se lee de arriba abajo. Por poner un ejemplo muy simple, si en un momento del código se define el color de una cabecera como rojo y más adelante se vuelve a declarar, pero en este caso como azul, el color final de la cabecera será azul dado que la ejecución se realiza en cascada.

Hoy en día es la forma más optima de implementar el diseño de una aplicación web dado que ofrece unas ventajas y facilidades que agilizan el proceso de la programación de dicho diseño gráfico, aumentando la adaptabilidad de la web y además garantizando que los usuarios puedan ver la web de la forma deseada por lo que su uso en este aspecto es incuestionable. Una de las razones principales de su implementación es una mayor facilidad de mantenimiento del código y una reducción en los tiempos de renderización. Esto hace referencia a lo que se ha mencionado anteriormente de las modificaciones de los elementos de la página web. Cuando se cambia una componente gráfica concreta el desarrollo mediante CSS evita una modificación en cada una de las páginas del código que componen la página web. De hecho, a mayor contenido tenga la aplicación web, mayor cantidad de tiempo se ahorrará con CSS. Esto se traduce en una mayor optimización del tiempo de trabajo del desarrollador y un mantenimiento más sencillo del código lo que es una ventaja muy importante ya que además se garantiza que todas las páginas utilizadas para conformar la aplicación web tengan un estilo coherente. Esto también produce una reducción del espacio de almacenamiento ya que al separar el contenido del diseño está provocando una reducción del tamaño de transferencia del archivo. Por un lado, se tendrá el documento CSS y por el otro lado, el código de la web. Esto significa que cuando un usuario accede al contenido de la página

web, a este código CSS solo se accede una vez para cargar su contenido en vez de tener que acceder a cada una de las páginas de la web. De esta forma se reducen los costes de alojamiento de la web al reducirse los tiempos de carga y por lo tanto requerir un menor ancho de banda, ofreciendo así al cliente una conexión más rápida y eficiente.

Otra ventaja interesante que ofrece son sus opciones de visualización, las cuales, dentro de los aspectos de estilo mejoran con cada actualización, permitiendo algunos efectos que hasta ahora solo se podían aplicar con Photoshop. Y opciones de visualización dentro de los aspectos de adaptabilidad, permitiendo a la web acomodarse a distintos dispositivos móviles ofreciendo así un entorno cómodo a nivel de usuario, sea cual sea el dispositivo elegido para acceder al contenido de una web concreta.

5.5 Adaptación a dispositivos

A día de hoy tener una web adaptable es algo básico para la mayoría de las aplicaciones web dado que en los últimos años los dispositivos móviles son mucho más usados por parte del usuario medio que los ordenadores. No sólo se debe comprobar que se adapta correctamente en móviles y/o tablets, sino que, además, tener una web que lo permita supone que todos los elementos principales de dicha aplicación puedan ser usables sin perder la facilidad de navegación o al menos intentando que su comodidad se reduzca lo mínimo posible para que no afecte a la experiencia de los clientes de dichas aplicaciones.

Esto se puede realizar de distintas formas como se explicará a continuación en el análisis. Las formas que se plantearán en la evaluación de las tecnologías serán 3. Una opción sería realizar un diseño *responsive* o adaptable para poder utilizar dichas aplicaciones en estos dispositivos. Otra opción es la publicación dinámica en la que existen distintos códigos HTML en la misma URL. O directamente adaptar la URL donde todas ellas tendrán otra equivalente que proporciona contenido optimizado.

6 Análisis de alternativas

Una vez establecidas unas bases para poder realizar un análisis de las tecnologías a utilizar se procederá a evaluar las opciones disponibles para el desarrollo del proyecto. Se irán describiendo estas alternativas para finalmente escoger las más adecuadas para la realización de la solución propuesta.

6.1 Definición de las problemáticas

Problemas de la interfaz entre aplicación y servidor actual:

- Crea un nuevo proceso por cada petición al servidor. Lo cual lo hace un sistema ineficiente desde el punto de vista de recursos para la creación de estos nuevos procesos.
- Cada proceso obligará a establecer una nueva conexión a la base de datos lo cual no es beneficioso de cara a incluir una en el futuro a la aplicación.
- Normalmente el código interpretado requiere más tiempo. Dado que se produce una sobrecarga en la interpretación de scripts.
- En cuestión de seguridad, tampoco es la mejor opción dado que se ejecuta en el *Shell* del sistema operativo.

Problemas del diseño actual:

- Se trata de un diseño que muestra la información sin aprovechar las facilidades gráficas de la tecnología actual.
- Dificulta la navegación y la experiencia del usuario.
- Complica la estructuración de la página web.

Problemas de acceso desde dispositivos:

- Es necesario reescribir el diseño adaptándose al auge de los dispositivos móviles para aumentar el rango de utilidad de la aplicación.

6.2 Alternativas de interfaz

6.2.1 CGI

El CGI es una API de servidor web. Se trata de un sistema de comunicación que le indica al servidor cómo enviar y recibir datos de una aplicación a un cliente. Esto permite a los usuarios utilizar aplicaciones de servidor concretas. Los programas CGI se ejecutan en el servidor web en cualquier lenguaje que pueda procesar variables de entorno y standard input y escribir en standard output**.** No existe ningún tipo de limitación al lenguaje de programación que podemos utilizar para escribir un CGI.

Se trata de la interfaz que esta implementada en la aplicación por defecto por lo que sus inconvenientes se han explicado en las problemáticas a resolver. Debido a estas el CGI se encuentra un poco desfasado por la dificultad con la que se desarrollan los programas y la pesada carga que supone para el servidor que los ejecuta. Hay que añadir también,

que sí el número de usuarios que ponen en marcha un CGI al mismo tiempo multiplicarán por esa cantidad de usuarios los recursos que ocupe.

6.2.2 Frameworks

Los frameworks en el servidor son *softwares* que facilitan las tareas de escribir, mantener y escalar aplicaciones. En resumen, facilitan las tareas de desarrollo web. Estas tareas incluyen enrutado de URL, interactuación con bases de datos, soporte de sesiones y autorizaciones de usuario, formateado de la salida (por ejemplo, HTML, JSON, XML), y mejora de la seguridad contra los ataques web.

La mayoría de sitios proporcionan un gran número de recursos diferentes, accesibles a través de distintas URL. Los frameworks web proporcionan mecanismos simples para mapear patrones URL a funciones de gestión específicas. Esta aproximación tiene también beneficios en términos de mantenimiento, porque puedes cambiar el URL que se usa para entregar una característica en particular sin tener que cambiar el código.

En lo que a base de datos se refiere, los frameworks web proporcionan una capa de base de datos que abstrae las operaciones de lectura, escritura, consulta y borrado de la base (Mapeado de Objetos Relacionados). Esto permite:

- Reemplazar la base de datos subyacente sin tener necesariamente que cambiar el código que la usa.
- La validación básica de datos puede implementarse dentro del framework. De esta forma, las distintas comprobaciones de los datos serán más fáciles y seguras.

Uno de los frameworks de código abierto es Django que se trata de un framework web para Python que tiene en cuenta muchos de los problemas de desarrollo web y cumple con todo los descrito anteriormente. Es veloz, seguro y muy escalable. Al estar basado en Python, el código de Django es fácil de leer y de mantener.

A continuación, se presenta una matriz de priorización para comparar cualitativamente CGI con un framework tipo Django.

Tabla 6.1. Matriz de priorización CGI/Framework

PESO ALTERNATIVA	Velocidad P=1	Base de Datos P=1	Seguridad P=1	Pequeña aplicación P=2	TOTAL
CGI	V=1 V×P=2	V=1 V×P=1	V=2 V×P=2	V=3 V×P=6	11
Framework (Django)	V=3 V×P=6	V=2 V×P=2	V=3 V×P=3	V=1 V×P=2	13

En la tabla queda reflejada la superioridad del uso de un framework sobre CGI. La valoración principal se le ha aportado al apartado de pequeña aplicación dado que se considera crucial la optimización de costes y tiempo en cualquier proyecto de ingeniería. Aquí CGI es superior en sencillez, en cambio Django a veces puede ser demasiado avanzado para aplicaciones pequeñas.

En cuestión de seguridad, Django es superior contra ataques a la web que CGI dado que como se ha comentado se ejecuta en el Shell del sistema operativo.

La base de datos se ha considerado como una posible opción a futuro por lo que su peso es menor. En este caso, el framework estará por encima debido a que CGI obliga a tener que acceder a ella en cada proceso por lo que en una futura implementación su uso sería más optimo con Django.

Como se ha comentado en las problemáticas la velocidad de CGI no es muy alta por lo que también está por debajo en esta comparación. Sin embargo, al tratarse de una web de poco almacenamiento el peso que se le ha dado no ha sido alto.

6.3 Alternativas de diseño gráfico

En la resolución de problemas de disposición de aplicaciones web es necesario tratar las distintas formas de estructuración de las mismas. Para solucionar los problemas derivados del diseño gráfico existen múltiples opciones. Al tratarse de una pagina sencilla las alternativas que se presentan se refieren únicamente a aspectos como el menú, referencias en botones u otras posibilidades.

Se van a proponer tres opciones de estructuración partiendo de la base de que el contenido se encontrará distribuido en pestañas desplegables en cualquiera de las tres.

6.3.1 Diseño con menú inicial

Una opción que facilita la navegación a través de la página web es la disposición de un menú que estructure el contenido de la página en distintos apartados. De esta forma el contenido de la aplicación estará ordenado y su acceso y orientación será sencillo. Este menú llevara al usuario a una referencia concreta donde se almacena la información en su conjunto, es decir, todos los elementos de un mismo tipo dentro de un solo enlace.

6.3.2 Diseño unido

Otra opción puede ser, partiendo de la base de que todos los elementos ya se encuentran organizados en pestañas, colocar la información en una única página donde se distribuyen todos. En este caso, la disposición es menos ordenada, pero al estar todo archivado en desplegables no habrá que navegar en exceso para encontrar la referencia buscada y se ahorra al usuario el pasar por un menú intermedio.

6.3.3 Diseño Independiente

Esta opción está conformada por un menú inicial que como en el anterior se divide en distintas opciones, solo que a diferencia del primero que dirigía a una página donde se encontraban todos los elementos correspondientes a cada apartado, este menú será un desplegable que dirigirá hacia la referencia de uno en concreto. Por lo tanto, esta será la opción de navegación más rápida, aunque obligara al usuario a ir hacia atrás cada vez que se quiera ver otro elemento, aunque este pertenezca al mismo apartado.

Estas alternativas dependen del realizador del proyecto. A pesar de su subjetividad, es menester del ingeniero dejar claro al programador el diseño que desea obtener para la aplicación web que se va a realizar. La tecnología a utilizar, como la mencionada anteriormente (CSS), será decisión del programador de diseño gráfico.

6.4 Alternativas de adaptabilidad

6.4.1 Responsive

Este método está basado en una configuración que proporciona a todos los dispositivos un mismo HTML que se ajusta al tamaño de la pantalla. Esto quiere decir que el contenido de las aplicaciones web estará adaptado a cualquier dispositivo que se utilice para acceder a él.

En este caso, las URL no cambian por lo que a los usuarios les es más fácil compartir contenido y enlazarlo. Requiere menos tiempo a la hora de actualizar el contenido ya que si se actualiza el contenido de una página de la web, éste se actualiza también en las versiones móviles. Además, no se requieren redirecciones, lo cual aumentaría el tiempo de carga de tu web.

Una desventaja que presenta este método de adaptación es la dificultad de reedición. Esto quiere decir que, si se ha realizado el diseño de la web sin tener en cuenta el modo *responsive*, es probable que lo mejor sea empezar de cero.

6.4.2 Publicación dinámica

Como se ha comentado con anterioridad este formato se basa en la existencia de varios códigos HTML en la misma URL, de forma que en función del dispositivo que se utiliza cargara una u otra.

La publicación dinámica tiene prácticamente las mismas ventajas que la opción *responsive*, aunque habrá que realizar el proceso correctamente dado que sino las probabilidades de que no funcione son mayores.

Lo más complicado de este método es la detección de agentes de usuario, ya que es una técnica muy propensa a errores. Si no está bien desarrollada es muy probable que no se detecte correctamente desde qué dispositivo accede el usuario y muestre un contenido equivocado.

6.4.3 Adaptar la URL

Cada URL de la web tiene asignada una equivalente para la versión optimizada a los dispositivos. Para evitar que haya confusión por URL repetidas, se pueden utilizar etiquetas en HTML que indican que se trata de una adaptación y no de contenido repetido. Se trata del método más sencillo, aunque también se compromete la capacidad de uso de la página web.

Es importante que el redireccionamiento sea correcto, dado que una cantidad excesiva del mismo puede perjudicar al tiempo de carga de la página web. Además, caso de una incorrecta configuración de estas redirecciones los usuarios no podrán ver ningún contenido. Si por ejemplo una página para ordenador se redirecciona a una página para otro dispositivo sin relación, el navegador no entenderá el contenido de la web.

A la hora de elegir la solución más adecuada, se ha realizado otra matriz de priorización donde se exponen los parámetros que se consideran más importantes en este sentido.

Tabla 6.2. Matriz de priorización de adaptabilidad

PESO ALTERNATIVA	Velocidad P=1	Coste de actualización P=2	Contenido P=1	Coste P=-1	TOTAL
Responsive	V=3 V×P=3	V=5 V×P=10	V=3 V×P=3	V=1 V×P=-1	15
Publicación dinámica	V=4 V×P=4	V=2 V×P=4	V=4 V×P=4	V=2 V×P=-2	10
Adaptar la URL	V=5 V×P=5	V=1 V×P=2	V=5 V×P=5	V=3 V×P=-3	9

Como se puede observar, la opción más ventajosa es Responsive. Esto se debe a que el parámetro con más peso, es decir, el coste de actualización, presenta el mejor comportamiento. Esto quiere decir que, a la hora de actualizar la aplicación, los cambios se aplican únicamente en CSS y no en HTML, a diferencia de las otras alternativas. El resto de aspectos a analizar obtienen una puntuación menor en el caso de Responsive, pero se ha considerado un peso más bajo debido a la propia naturaleza de la aplicación que se estudia en este proyecto concreto. La velocidad de navegación es la más baja y el contenido que se muestra en el móvil es igual al del ordenador (no se implementa una web específica para el móvil). El primer caso no supone un gran problema ya que la carga de datos que requiere es baja y la velocidad se podría arreglar mediante otros métodos más efectivos. Por otro lado, dado que la aplicación se usará principalmente en ordenador por los profesores (en versión móvil lo más probable es que solamente lo empleen los alumnos para comprobar sus datos), el peso del contenido también es menor, esto es, no resulta tan interesante una web dedicada para otros dispositivos. Por último, también se ha considerado el coste, que resulta menor en la alternativa elegida por ser la más sencilla.

6.5 Selección de las alternativas

Para el desarrollo de esta práctica a modo de introducción para futuras actualizaciones se han tenido en cuenta las alternativas que se han comentado en los apartados anteriores. Las alternativas que se van a emplear en el proyecto son como interfaz CGI, aunque su velocidad, seguridad e interacción con la base de datos sea menor, en el apartado de pequeña aplicación que además se ha priorizado sale ganando. La razón es que, al tratarse de una aplicación con poco riesgo de fallas en la seguridad, debido a que, aunque monitoree el trabajo no es crucial en la calificación de los alumnos, y a que

no tiene una carga de datos alta por lo que la velocidad no es un problema se ha priorizado la optimización de tiempos y coste en su realización.

Por otro lado, en el diseño, por consideraciones de estructuración, se ha optado por un diseño estructurado a partir de un menú que separe los tipos de elementos en prácticas, exámenes y resultados.

Por último, teniendo en cuenta la matriz de probabilidades y el coste de actualización como apartado de mayor peso, se ha optado por realizar una adaptación *responsive* dado que no hay gran interés en la realización de web dedicada para móvil ya que se considera suficiente realizar una adaptación de la página original de ordenador.

A pesar de estas elecciones, es importante destacar que a efectos de mejorar la aplicación en el futuro añadiendo mejoras como una base de datos, las opciones pueden variar y ser más optimo el uso de otra tecnología como Django que obviamente en niveles de rendimiento y seguridad es suprior a CGI. Por lo demás, se sigue considerando el menú inicial y la tecnología *responsive* como las opciones claras por las que decantarse.

7 Descripción solución propuesta

7.1 Instalación del servidor

7.1.1 Instalación en el SO

Para implementar la aplicación web lo primero que habrá que hacer será instalar un servidor concreto. En el análisis anterior se han propuesto opciones a tener en cuenta. De todas ellas se va a escoger el servidor web de Apache por su sencillez de configuración y su gran apoyo en la comunidad de código abierto en Linux. Aunque no es el servidor con mayor rendimiento, la aplicación a desarrollar no va a tener que soportar una alta tasa de tráfico por lo que este será un problema menor.

Este servidor es fácilmente descargable desde su página web o directamente desde la terminal de Linux como ha sido el caso. Por lo tanto, con un sencillo comando "*apt install*" será posible descargar "*apache2*" en su versión 2.4. Antes de dar por finalizado el proceso de instalación del *software* en el equipo, solo hay que realizar las convenientes comprobaciones de que todo está correcto. Para ello lo primero es comprobar la configuración del cortafuegos para garantizar el acceso desde fuera a los puertos web por defecto. Después de eso, hay que poner en marcha el servidor en la IP o "localhost" por medio del correspondiente comando.

Una vez se tiene el software operativo hay que adaptarlo para que pueda ejecutar la aplicación por defecto en el puerto que se le indique. Para ello, habrá que acceder al fichero que viene por defecto sobre la configuración de Apache, *apache2.conf* y modificarlo como se indica en la imagen.

Originalmente, Apache solo lee los archivos que se encuentran en el directorio "*var/www/"* por lo tanto hay que modificar esa dirección en el fichero como se puede ver a continuación utilizando el directorio del proyecto:

```
<Directory /home/pedrofernandez/projects/web-monitor>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Figura 7.1. Directorio de acceso de Apache

La opción "*Indexes FollowSymLinks*" lo que hace es que seguir un link simbólico en el directorio especificado. En este caso, buscara un "*.html*". Además, se niega el acceso a los archivos *".htacces*" y permite el acceso a casi cualquier archivo del servidor Linux.

Una vez hecho esto, hay que indicar los puertos que se van a utilizar para implementar la página web. Para ello hay que acceder al fichero de configuración de puertos e indicarle que puerto se va a utilizar. En el caso del proyecto, se empleará el puerto *8080.*

Ya solo queda configurar los ficheros "*sites-available*" y "*sites-enabled*". El directorio "*/etc/apache2/sites-available"* contiene archivos de configuración para *Virtual Hosts* de Apache. Los hosts virtuales permiten que Apache se configure para múltiples sitios que tienen configuraciones separadas. Por defecto, habrá un archivo de hosts virtual llamado "*000-default.conf*" que ayuda a cargar la página predeterminada del servidor web Apache a petición. En este archivo se incluirá el puerto y el directorio donde se encuentra el proyecto que se especificará como documento raíz. Aunque no se ha utilizado en este caso, se puede añadir también el nombre del dominio y la dirección del administrador del servidor. También hay que incluir el documento raíz dentro del archivo "*default.conf*". Con esto ya se tienen configurados los *Virtual Host* de Apache. En directorio "*sites-enabled*" se encuentra el mismo archivo "*000-default.conf*", por lo que no es necesario configurarlo dado que cambia cuando se modifica en "*sites-available*". Para que un sitio sea accesible, se debe crear un enlace a su archivo de configuración en el directorio "*/etc/apache2/sites-enabled"*. Esto se hace usando el comando *a2ensite* en la terminal.

7.1.2 Habilitar CGI

El primer paso es activar la función de CGI de Apache. Para ello, hay que habilitar el mod_cgid que es el encargado de permitir que apache2 ejecute archivos CGI. En la imagen adjuntada a continuación se muestra las modificaciones que habrá que realizar en el archivo de configuración del servidor "*apache2.conf*":

```
ServerName localhost
Alias "/web-monitor/" "/home/pedrofernandez/projects/web-monitor/"
Options +ExecCGI
AddHandler cgi-script .cgi .pl .py
```

Figura 7.2. Configuración para CGI para apache2

Lo primero es declarar el directorio donde se quiere que Apache acceda y trate los archivos como archivos CGI. Para ello, se tienen dos directivas que se pueden ejecutar, *ScriptAlias* y *Alias*. Ambas direccionan a la URL donde se encuentra el archivo y tratan el contenido que no está directamente bajo el "*DocumentRoot*" como si fuera un árbol del archivo a ejecutar. En la directiva ScriptAlias, todos los documentos que se encuentren dentro del directorio, en esta configuración el directorio es "*web-monitor*", serán tratados como archivos CGI. En este caso, se va a usar la directiva Alias puesto que, aunque el archivo a ejecutar haga uso de un *script* que utiliza CGI, el ejecutable que llama a ese archivo es un HTML, por lo que no se busca tratar todo el contenido como ficheros CGI.

A continuación, hay que definir en la siguiente línea la opción de ejecutable CGI. Esto se hace porque Apache no ejecutara un *script* CGI salvo que este se encuentre en un directorio que tenga la opción descrita activada. Para ello hay que activarla mediante la directiva *Options.* Si no existe la directiva *Options* en ese directorio habrá que añadir un "+".

Al final es importante añadir la directiva "*AddHandler cgi-script*" para indicar que clase de archivos se consideran como un archivo "*cgi*". Como se puede observar en esa configuración se incluyen los archivos en formato Python. Por último, queda configurar el archivo "serve-cgi-bin.conf" que se encuentra en los directorios "*conf-enabled*" y "*conf-available*".

En la imagen 7.3, se muestra el código que hay que realizar para finalizar el proceso de habilitación.

Como se puede observar, cuando se ejecuta la directiva Alias, se activa el mod_alias. En este momento, se realizarán las siguientes sentencias. Se comprueba que el

mod_cgid, el cual se ha mencionado previamente está activo y si lo está se definirá el "*ENABLE_USR_LIB_CGI_BIN*" y por consiguiente entrará en la siguiente sentencia que aplicará el Alias al directorio indicado.

Además, como se ha comentado anteriormente, habrá que activar la opción de ejecutable CGI a dicho directorio.

```
<IfModule mod_alias.c>
    <IfModule mod_cgi.c>
        Define ENABLE_USR_LIB_CGI_BIN
    </IfModule>

    <IfModule mod_cgid.c>
        Define ENABLE_USR_LIB_CGI_BIN
    </IfModule>

    <IfDefine ENABLE_USR_LIB_CGI_BIN>
#       Alias /cgi-bin/ /usr/lib/cgi-bin/
#       <Directory "/usr/lib/cgi-bin">
#           AllowOverride None
#           Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
#           Require all granted
#       </Directory>

                Alias /web-monitor/ /home/pedrofernandez/projects/web-monitor/
    <Directory "/home/pedrofernandez/projects/web-monitor/">
        AllowOverride None
        Options +ExecCGI
    </Directory>

    </IfDefine>
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figura 7.3. Configuración de serve-cgi-bin

Una vez hecho esto, bastara con acceder al *localhost* de identificación de la máquina y añadir el puerto que se le ha asignado a la aplicación para su puesta en marcha.

7.2 Rediseño de la aplicación

7.2.1 Comunicación entre aplicación y servidor

El proyecto define una comunicación mediante el uso de CGI. Una vez instalada la aplicación en el servidor, se cargan los valores en forma *input* en el archivo definido para trabajar como CGI mediante un *fieldStorage* y a continuación se va obteniendo esa información por medio de *getvalue* para utilizarla en el código. De esta forma se interactúa con el servidor en forma GET y solicitándole la información donde la primera ejecución será la del statsDNI que, en primera instancia, verificará al usuario.

7.2.2 Rediseño gráfico

- **Menú inicial de la aplicación**

El primer paso es crear un menú con referencias a las partes de información más importantes de la aplicación. Para ello lo que se ha hecho es cargar la información "*input*" mediante el *script* HTML en un fichero CGI que se ha denominado *"Pagina.py"*. Se trata de un pequeño código Python utilizado para generar una página entre el menú

de registro y la información que se almacena en el fichero origen facilitado. Es necesario que se trate de un fichero CGI y que importe el fichero de verificaciones "*statsDNI*" para que pueda leer y almacenar la información solicitada al servidor dado que sino la operación de registro será inviable y cualquiera podrá acceder sea el DNI auténtico o no. Además de que no cargaría la información. Una vez realizado este paso, hay que relacionar el fichero con una hoja de estilos CSS para darle el formato deseado. Abajo se adjuntan unas imágenes del mismo. Se trata de un menú sencillo que facilita la navegación de los usuarios a través de la página para que la búsqueda de servicios solicitados sea más accesible.

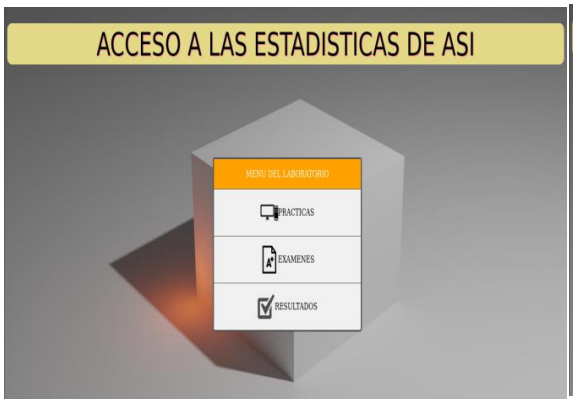


Figura 7.4. Menú de Aplicación

Figura 7.5. Menú (Selección "Práctica")

Figura 7.6. Menú (Selección "Exámenes")

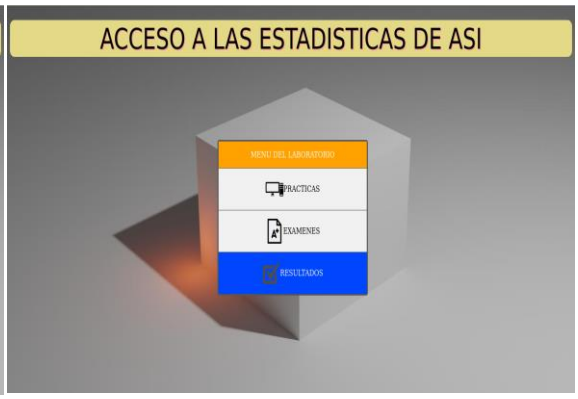


Figura 7.7. Menú (Selección "Resultados")

Para programar ese menú, lo primero y más importante ha sido editar referencias. Es necesario que cada una de las partes del menú referencien al lugar indicado. Para ello se han utilizado las etiquetas de tipo "<a>". A continuación, se agrupan las opciones en una etiqueta de lista no ordenada (<ul>) y a cada elemento se le añade un indicador de ítem de lista (<li>). Para poder dar a cada apartado un color diferente de tal forma que el menú sea más dinámico, lo que se ha hecho es dividir cada uno de los elementos

etiquetados como ítem en clases distintas. Todas estas características se aprecian en la imagen adjuntada a continuación.

```
        print "<ul>"
        print "<li><a class='active' href='#'>MENU DEL LABORATORIO</a></li>"
        print "<li><a class='pr' href='../cgi-bin/practice.py?path=%s&DNI=%s&nprac=1
es/15375.ico'>PRACTICAS</a></li>" % (webpath,DNI)
        print "<li><a class='ex' href='../cgi-bin/practice.py?path=%s&DNI=%s&nexam=1
=9&nexam=10&nexam=11&nexam=12'><img src='../../images/103339.ico'>EXAMENES</a></li>"
        print "<li><a class='re' href='../cgi-bin/practice.py?path=%s&DNI=%s&nresult
S</a></li>" % (webpath,DNI)
        print "</ul>"
```

Figura 7.8. Codificación del menú

Como se puede ver, todos los ítems tienen una referencia excepto el de clase activa. Ese hace referencia al título del menú y se le ha asignado una referencia inexistente para que no acceda a ningún sitio. Se mantiene fijo puesto que solo es el título. Como se ha mencionado, a cada elemento se le ha asignado una clase para que los colores sean distintos cuando el cursor pasa por encima. Del mismo modo la clase “active” del menú se mantiene en el mismo color de forma permanente. Para ejecutar estas sentencias ha habido que editar la hoja de estilos. En la imagen añadida abajo se indica el procedimiento que se ha seguido.

```
33 li a {
34   display: block;
35   color: #000;
36   padding: 16px 32px;
37   text-decoration: none;
38 }
39
40
41 li a.active {
42   background-color: #ffa200;
43   color: white;
44 }
45
46 li a.pr:hover:not(.active) {
47   background-color: #ff2800;
48   color: white;
49 }
50
51 li a.ex:hover:not(.active) {
52   background-color: #1ad600;
53   color: white;
54 }
55
56 li a.re:hover:not(.active) {
57   background-color: #0046ff;
58   color: white;
59 }
```

Figura 7.9. Hoja de estilos del menú

Como se puede ver, en la primera llave el elemento más importante es “*display: block*” esto quiere decir que muestra cada elemento “
<a>” en una nueva línea. De esta forma se está disponiendo que el menú sea vertical. Lo demás son elementos de decoración como el “*padding”* que determina el tamaño de las casillas. En el resto del código hay que fijarse en la directiva “*:hover:not(.active)*”. Esta sentencia “*:hover*” indica que cuando el cursor está sobre el elemento etiquetado como por ejemplo “<a class=”pr”>”, este adquiere las características indicadas entre llaves. En este caso, cuando el cursor esta sobre “PRÁCTICA”, el fondo se vuelve rojo y las letras blancas.

Cuando no, retoma las características definidas en “li” (no se muestran en la imagen: fondo blanco, letras negras). El “*:not(.active)*” le niega esta característica de cambio de color al elemento de la clase “*active*”

En la imagen no se han mostrado las estructuras de “<ul>” y “<li>” dado que solo contienen información como el color de relleno, tamaño de la tabla, posición de la tabla o del texto. Además, hay que añadir que para situar los iconos a la izquierda se ha utilizado la etiqueta “<img>”, dentro de la cual se ha añadido la fuente o directorio en el que se encuentra la imagen. Si se ha entendido lo básico de la metodología de CSS, es fácil deducir que esta deberá colocarse dentro de la etiqueta de multivínculo en la posición en la que se quiera colocar, en este caso antes del texto. Al estar la sentencia dentro de “<ul>” las imágenes estarán dentro de la tabla. Por lo tanto, es visible como es el orden de ejecución las etiquetas y en función de eso se van estableciendo las características en la hoja de estilos. Para acabar de alinearla hay que aplicar un “*vertical-align: middle;*”. Esta sentencia colocará la imagen centrada con el texto.

Para acabar con el menú, simplemente añadir que la imagen de fondo se coloca aplicando en la hoja de estilos como indicador de la directiva “*background*” la URL del directorio en el que encuentra la imagen colocada. Es importante añadir que, si la imagen no cubre el tamaño del monitor y no se añade la directiva de no repetición, esta se repetirá de forma indefinida hasta rellenar el fondo.

A continuación, se va a acceder a la parte de “PRÁCTICAS”. El diseño de esta parte es igual que el de exámenes y resultados por lo que solo se explicará una vez para no ser repetitivo.

- **Entrada en la aplicación (alumno)**

Como se mostrará en el anexo, el punto de partida era un conjunto de información mostrado de arriba hacia abajo separado por líneas. No se le había aplicado ninguna hoja de estilos y los detalles se habían realizado utilizando directamente HTML. Lo que se ha buscado es simplificar el diseño y agrupar lo máximo posible, pero manteniendo la coherencia de la aplicación para no perder efectividad a la hora de realizar una aplicación cómoda y funcional. De esta forma, los usuarios podrán acceder más rápidamente a la información que desean y sin necesidad de ir buscando a lo largo de la página la práctica que buscan. Como se indica en el punto, esta es la parte de la aplicación referente al alumno, es igual que la del profesor solo que hay algunos detalles que los diferencian dado que no disponen de los mismos permisos por lo que no podrán acceder a las tablas de usuarios que se mostrarán en el paso siguiente. Se adjuntan dos imágenes globales para poder explicar más en detalle cómo se han ido realizando los cambios. Se puede observar en ellas las agrupaciones de prácticas y exámenes de la asignatura. Se trata de un diseño simple dada la naturaleza de la aplicación.

Figura 7.10. Acceso a prácticas

Figura 7.11. Acceso a exámenes

Empezando por lo básico, el proceso para la colocación del icono ha sido el mismo que en los iconos del menú solo que se le han aplicado algunos elementos extra de diseño como el sombreo del borde o ese formato circular que usa la directiva "*border-radius: 50%*". Ahora se va a explicar algo importante a la hora del diseño. Primero se va a mostrar una de las pestañas desplegadas para que el concepto que se va a comentar esté más claro.

Figura 7.12. Práctica 3 desplegada

Figura 7.13. Práctica 3 desplegada (continuación)

Como se puede ver las pestañas se pueden desplegar de forma independiente para poder acceder únicamente a la que interesa. Sin embargo, lo importante que se quería explicar es que, como se observa en la figura 7.13, hay una pestaña desplegable que se llama “INET/HW”. Esto quiere decir que las pestañas de prácticas hay que programarlas de forma que se mantengan abiertas todo el rato incluso cuando se despliega esta segunda pestaña. Esto se indica porque muchos de los desplegables se realizan con la metodología “*:focus*” de CSS. De hecho, las tablas de direcciones INET y maquina HW se despliegan con este método. Esta característica expone que cuando realizas un clic esta se despliega porque queda en *“focus”*, pero en cuanto se hace clic en otro lugar, la zona de “*focus*” cambia por lo que vuelve a plegarse. Esto es de gran importancia ya que, si se programan los desplegables de prácticas de esta forma, al hacer clic en “INET/HW” el focus cambiará de lugar para poder abrir esas tablas, pero al mismo tiempo dejará de estar en la pestaña “INFORMACIÓN DE LA PRÁCTICA” y esta se cerrará y por lo tanto nunca podrá desplegarse la pestaña que está dentro de esta otra pestaña.

Para solucionar esto se ha realizado lo que se conoce como “*checkbox*”. Este método representa la solución a este problema dado que se genera una “caja” que se mantiene activa hasta que se vuelva a pulsar en la pestaña y, por lo tanto, de esta forma sí que se podrá desplegar la pestaña INET/HW como se muestra justo abajo.

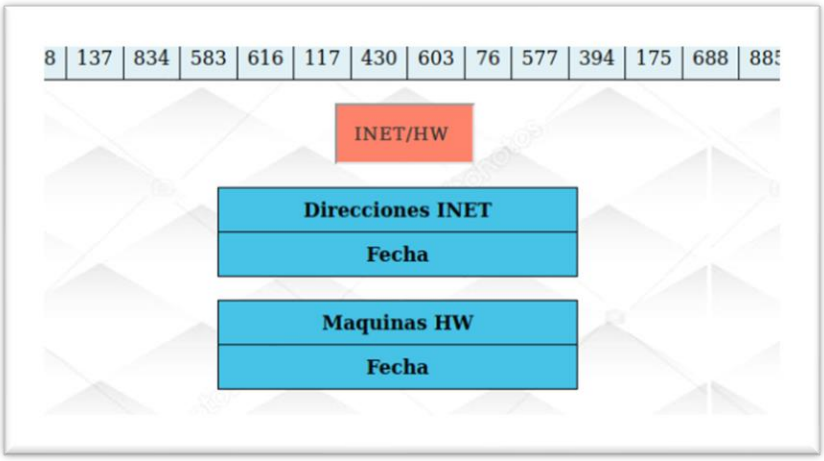


Figura 7.14. Tablas INET/HW desplegadas

Para aplicar esta característica a la hora de realizar los desplegables se debe generar una etiqueta “<details>”. Esta agrupación indica un trozo de contenido oculto. Además de esta etiqueta hay que añadir el elemento “<input>”, que representa un campo de datos al que se le asocia un tipo que normalmente se corresponde con un control que permite a los usuarios editar su valor. En este caso, como se ha comentado, es un tipo “*checkbox*”. Se adjuntan abajo las dos imágenes acerca de la lógica de ambos códigos,

el que incluye "*checkbox*" no requiere de código CSS excepto para el diseño. En cambio, en el que se ha usado para las tablas "INET/HW" el CSS juega un papel importante como se ha mencionado anteriormente.

```
print "<details class='prex'>"
print "<summary>INFORMACION DE LA PRACTICA
print '<p><input type="checkbox" name="pr"
print '<a href="#ocultar" class="ocultar">
printStatUser(user,int(str),"ESTADISTICAS
print "</p>"
print "</details>"
```

Figura 7.15. Código para desplegable checkbox

```
print '<input type="text" value="INET/HW">'
print "<div id='contenido'>"
print "<br>"
printInvertTable(userHier,userHier.Inet)
print "<br>"
printTable(userHier,userHier.HW)
printInvertTable(userHier,userHier.HW)
print "</div>"
```

```
#contenido {
        display: none;
}
input[type="text"]{
        position:relative;
        margin: 0 0 0 47.7em;
        text-shadow: 0 0 0 #000;
        padding: 1em;
        width: 8em;
        cursor: pointer;
        font-family: italic;
        letter-spacing: 1px;
        background: #ff7f67
    }
input[type="text"]:focus{
        outline: none;
}
input[type="text"]:hover{
        background:#b8d2f2;
}

input:focus + div#contenido {
        display: block;
}
```

Figura 7.16. Código HTML (Izq.) - Código CSS (Der.)

Para la codificación del desplegable "INET/HW" (Figura 7.16) se ha creado un campo de datos de tipo texto que representa el botón que muestra las tablas. El contenido se ha designado mediante una clase "contenido". En la hoja de estilos (izquierda) se declara dicho contenido como oculto de forma predeterminada. Saltándose las llaves que representan el diseño se observa que la directiva "*input:focus + div#contenido*" incluye un *display* que, cuando *input* se encuentra en "*focus*", muestra el " " con "id= contenido" que se sitúa justo después.

- **Entrada en la aplicación (profesor)**

Cuando la entrada a la aplicación la realiza un profesor, éste tendrá acceso a la lista de alumnos que cursan la asignatura. Por lo tanto, el diseño será ligeramente distinto en

este caso. Como se puede observar en la imagen que se adjunta más abajo, los profesores tienen acceso a las estadísticas de los alumnos.

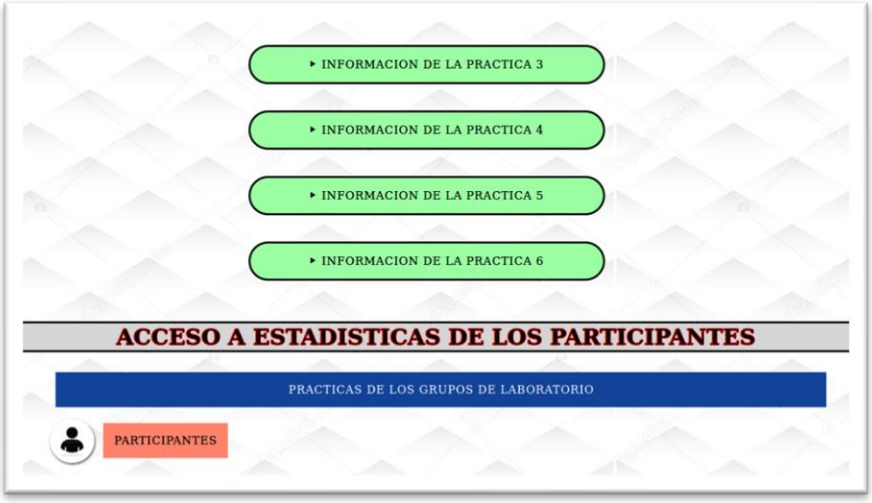


Figura 7.17. Acceso a las estadísticas de los alumnos

La pestaña de prácticas se despliega para que el profesor acceda a la práctica que le interesa. De esta forma puede acceder a los datos que reflejan el estado del trabajo de los alumnos de la asignatura.

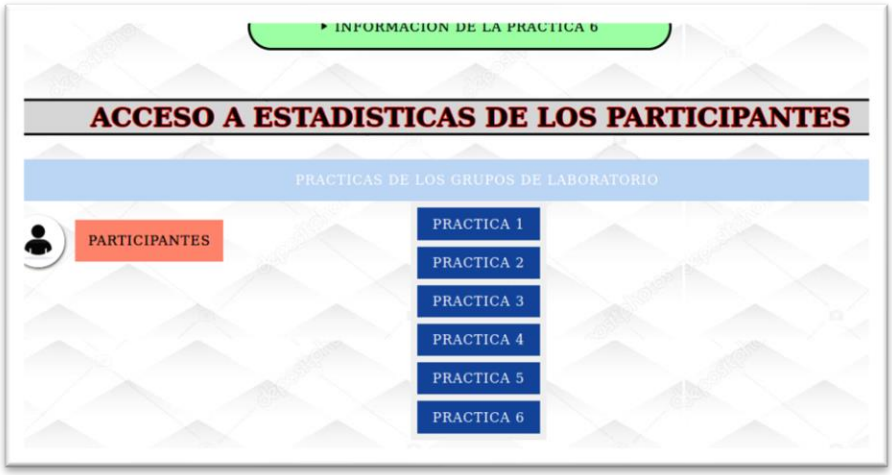


Figura 7.18. Despliegue de prácticas

El sistema que se ha utilizado para programar este menú de prácticas ha sido prácticamente el mismo que el utilizado en el menú inicial. En él se han hecho uso de etiquetas de listas, ítems e hipervínculo. La distribución en este aspecto es la misma, la etiqueta de lista envuelve todos los ítems que son cada una de las prácticas que a su

vez representan un vínculo asociado como referencia. La diferencia con respecto al menú es que, en este caso, al desplegarse un pliego de opciones, habrá que añadir una etiqueta de lista dentro de la propia lista. Por otro lado, habrá que añadir una etiqueta "<nav>" que se utiliza para englobar enlaces de navegación. Esto se hace de la siguiente manera.

```
print '<nav id="menu">'
print "<ul>"
print '<li><a href="#">PRACTICAS DE LOS GRUP(
print "<ul>"
for str in strprac:
        print "<li><a href='../cgi-bin/practi
        print "Practica %s</a></li>" % str
print "</ul>"
print "</li>"
print "</ul>"
print "</nav>"
```

Figura 7.19. Menú desplegable

En lo que se refiere a la programación de la hoja de estilos del menú, habrá que distribuirla en dos secciones, la que detalla el menú principal, que es fijo, y la parte del menú que se despliega del primero. En el caso del primer menú, el cual no está referenciado para mantenerse fijo, las indicaciones son todas de diseño. En el segundo, la mayoría de ellas son también de estilo. Aunque otras son importantes. Por ejemplo, la de la imagen que se adjunta a continuación.

```
257
258 #menu ul li:hover > ul {
259
260         display:block;
261
262 }
263
```

Figura 7.20. Display del menú desplegable

Estos comandos indican que mientras se aplique "*:hover*" al menú principal, el menú secundario se mantiene desplegado. Además, el menú principal se mantiene en ese estado de "*:hover*" mientras navegas por el menú desplegable. De esta forma, el menú solo se pliega en el momento que coloques el cursor fuera del área del menú, y sea el principal o el secundario.

Justo debajo de este menú, se encuentra la pestaña que despliega la lista de DNI de los alumnos que en ese momento cursan la asignatura. Como se observa en la imagen de abajo, se ha aplicado esta misma sentencia "*:hover*" a la lista para que mientras se está navegando sobre ella se iluminan los recuadros que encierran los nombres. Es algo

muy sencillo, solo hay que aplicar ese comando a los elementos fila que contienen los nombres.

Figura 7.21. Tabla de DNI

Para este caso de los DNI, la estructura seguida para desplegar la tabla ha sido distinta a la de las practicas. Aun así, el desafío era el mismo, conseguir que el desplegable se mantuviera activo para poder acceder a los enlaces de su interior. El uso del "*checkbox*" se mantiene, sin embargo, en este caso se ha añadido "<label>" que permite asignar una etiqueta a un elemento de entrada y que permite el clicado de dicho elemento. En este caso, lo que se ha hecho ha sido relacionarlo con el id "<input>" de tipo "*checkbox*" como se ha mencionado antes. El contenido se ha concentrado con la etiqueta correspondiente y todo el conjunto (<input><label>) a su vez se ha incluido en una etiqueta " " a la que se le ha asignado una clase. Todo esto se puede comprobar a continuación.

```
print "<div class='boton'>"
print "<div id='imagen'>"
print "<img src='../../images/icono-de-la-persona-en-el-fondo-blai
print "<input id='idBoton' type='checkbox'>"
print "<label for='idBoton' class='clicker'>PARTICIPANTES</label>'
print "<div class='contenido'>"
print "<br>"
print "<Table border='1'>"
print "<TR><TH><letra>DNI</letra></TH><TH><letra>Nombre alumno</l
for student in userList.usersList:
        print "<TR><TD><letra>%s</letra></TD><TD class='datos'><a
DNI=%s&nprac=1&nprac=2&nprac=3&nprac=4&nprac=5&nexam=1&nexam=2&ne
 webpath, student.dni,student.name)
print "</Table>"
print "</div>"
print "</div>"
```

Figura 7.22. Etiquetado de la tabla de DNI

Como se puede ver se le asigna el etiquetado de “<label>” al “*checkbox*”, de esta forma relacionamos ambos. El funcionamiento se define en la hoja de estilos. El objetivo es mantener el contenido oculto de forma predeterminada como en el resto de los desplegable y se indica que cuando el “<input>” está en estado “*:checked*” el contenido se muestra en la siguiente línea.

```
.contenido {
  display: none;
}

.boton input:checked ~ .contenido {
  display: block;
}
```

Figura 7.23. CSS tabla de DNI

Por último, solo queda mostrar el acceso a una practica concreta para mostrar las tablas de los distintos grupos del laboratorio. La lógica para programar estas tablas ocultas es la misma que se ha usado para los desplegables de las prácticas y los exámenes por lo que no se va a mostrar el código. Lo único que varía es la hoja de estilos que edita el diseño.

Figura 7.24. Grupos de Laboratorio

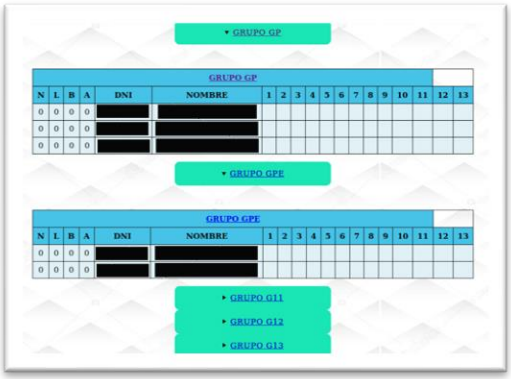


Figura 7.25. Grupos De Laboratorio (desplegado)

7.3 Adaptación a dispositivos

A la hora de utilizar una aplicación web en dispositivos móviles se puede perder la funcionalidad de ciertos elementos. Para ello es necesario adaptar la web a los diferentes dispositivos.

Como se comentó anteriormente en el apartado que trataba el tema del método con *responsive,* es importante realizar el diseño de la pagina web y la adaptación móvil al mismo tiempo dado que una posterior modificación es muy complicada. Para este caso práctico se mostrará un pequeño ejemplo y se comentaran que opciones se disponen para esta adaptación.

7.3.1 Ejemplo práctico

Para la realización de este ejemplo de cómo adaptar una aplicación web sencilla a móviles se va a tomar como punto de partida el menú de la aplicación web. Como puede verse en la imagen, este es el aspecto del inicio de la página web previa adaptación.

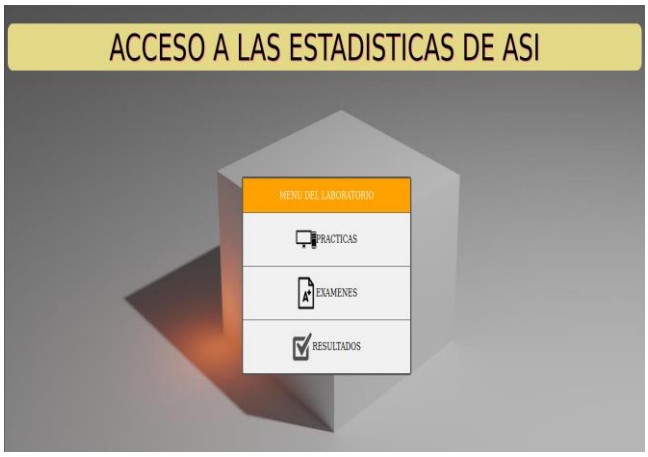


Figura 7.26. Menú de inicio

Para realizar el proceso se ha utilizado como base central la lógica de "*Media Query*". Esta es una técnica introducida por CSS que utiliza las condicionales preguntándose si cierta condición es verdadera. Se puede aplicar esto en favor de la metodología *responsive*. Para ello solo hay que hacer uso una condición que, en este caso, serán los pixeles de la pantalla. De esta forma se define que a partir de un número concreto de pixeles se llevarán a cabo una serie de operaciones. Se va a incluir una imagen del mismo menú, pero adaptado a un dispositivo diferente.

Figura 7.27. Visualización del menú en Galaxy S9

En la imagen se representa una pantalla de móvil Samsung S9 con resolución 360x740px. Para adaptar la resolución del PC a teléfono móvil se ha realizado un pequeño código mediante la sentencia que se ha comentado. Además, en la misma cabecera donde se realiza el link de la hoja de estilos hay que llamar a la siguiente directiva:

<meta name="viewport" content="width=device-width, initial scale=1.0">

Se trata de un método que permite tomar el control de la ventana gráfica. Sirve para dar instrucciones al navegador sobre cómo controlar las dimensiones y escala de la página. El fragmento *width=device-width* define el ancho de la página para seguir el ancho de pantalla del dispositivo. *Initial scale=1.0* indica el nivel de zoom inicial cargar la página por primera vez. En la imagen siguiente se puede apreciar el proceso que se a seguido para adaptar el menú una vez se ha añadido la sentencia anterior.

```
media only screen and (max-width:800px){
        .titulo {
                font-size:6vw;
        }

        ul {
                width: 80%;
                height: 60%;
                margin: 6em 0 0 2.2em;
                text-align: center;
        }
        li a {
                text-align: center;
                height: 25%;
        }
        li a.pr {
                font-size: 6vw;
        }
```

Figura 7.28. Ajuste de Tabla

```
        img.ord {
                width: 3em;
                height: 3em;
                margin: 1em 0 0 1.8em;
                position: absolute;
        }
        img.not {
                width: 2em;
                height: 2em;
                margin: 1.5em 0 0 2.3em;
                position: absolute;
                }
        img.tic {
                width: 2.4em;
                height: 2.4em;
                margin: 1.3em 0 0 2.8em;
                position: absolute;
                }
```

Figura 7.29. Ajuste de imágenes

En la imagen de la izquierda se puede apreciar la sentencia de "*Media Query*" que indica que siempre que se trate de una pantalla cuya resolución de ancho tenga menos de 800px se le aplicarán esos cambios. En este caso, la pantalla es de 360px por lo que sí se le aplicarían los cambios. En los fragmentos del código se puede ver que se han ido reajustando los tamaños de las tablas, márgenes, imágenes… Esto como se ha comentado no afectará al diseño original dado que la resolución de pantalla tiene un máximo superior a 800px por lo que los cambios no le serán aplicados. Aunque no se muestra en las imágenes, el fondo de la aplicación o el título también han sido ajustados con la resolución.

7.3.2 Otras opciones

Aunque se ha presentado un ejemplo básico, las opciones son varias. Por ejemplo, para los menús se pueden realizar desplegables utilizando etiquetas que se incluyen dentro del condicional. De esta forma, lo que en un PC puede ser un contenido directamente visual en las aplicaciones adaptadas podría reducirse a una serie de pestañas desplegables.

Otra opción puede ser el ajuste de tablas, las cuales se pueden hacer *responsive* de varias maneras. Por ejemplo, se les puede añadir una barra propia para desplazarla de arriba abajo sin necesidad de navegar con la aplicación en sí. Otra opción es mostrar la tabla de forma horizontal y navegar por ella haciendo uso de la barra para moverse de izquierda a derecha.

Como se puede observar las opciones que se ofrecen son múltiples, incluso pasar de un diseño de página web horizontal a un diseño para dispositivo con orientación vertical. A partir de "*Media Query*" se pueden ir realizando multitud de cambios.

DESARROLLO

1 Plan de proyecto

1.1 Fases del proyecto

1.1.1 Fase 1: Revisión de las tecnologías

Como primer paso, antes de empezar con el desarrollo del trabajo, se realiza una revisión de las distintas tecnologías con las que se cuenta hoy en día en este campo. El objetivo de esta fase es conocer los distintos códigos empleados y sus distintas aplicaciones. Con este fin, se consultan las páginas oficiales de dichas tecnologías, mostradas en la bibliografía aportada.

Se trata de una fase muy relevante, aunque no suponga parte del trabajo escrito, ya que determina el correcto desarrollo de éste. Por esta razón se estima una carga de trabajo necesaria de 120 horas con una duración total de 8 semanas.

1.1.2 Fase 2: Índice

Una vez recopilada la información de la fase 1, se procede a realizar una estructura concreta del trabajo a desarrollar, es decir, el índice. Dicho índice puede ser modificado durante el desarrollo del trabajo, añadiendo o cambiando puntos que sean importantes. Con la estructura clara resulta más sencillo y ágil realizar posteriormente la redacción del proyecto. Es una fase de poca carga de trabajo, de unas 6 horas, para una duración total de 2 días.

1.1.3 Fase 3: Revisión de códigos

Esta fase consiste en profundizar en los códigos concretos que se van a emplear, analizando su lógica y estudiando las distintas modificaciones que se pueden realizar sobre ellos. Para ello se han empleado distintas páginas relativas a cada uno de los códigos presentes en la bibliografía. Para su realización la fase se divide en dos partes, cada una con una duración concreta dependiendo del código.

- **Python.** Carga de trabajo: 30 horas. Duración total: 2 semana
- **CSS.** Carga de trabajo: 30 horas. Duración total: 2 semana

1.1.4 Fase 4: Revisión de servidores

Al igual que en la fase anterior, se estudia el servidor a emplear, que es el responsable de ofrecer la información obtenida de los códigos de la fase 3 al usuario que la solicita. Para ello se emplea la documentación referente a este aspecto recogida en la página web oficial de Apache, que es el servidor que se requiere. Todo esto se estima en una carga de trabajo de 10 horas a lo largo de una semana.

1.1.5 Fase 5: Instalación del servidor

Una vez obtenida toda la información de cada uno de los elementos necesarios en las fases anterior, se procede a aplicarla de forma práctica. Para ello se empieza instalando el servidor (Apache). A continuación, se adapta para que lea la carpeta y se configura para leer CGIs y para que muestre la página en el puerto que se le indique. En esta fase únicamente son necesarios los ficheros obtenidos con la descarga de Apache del repositorio. La carga de trabajo se estima en 12 horas con una duración total de 5 días.

1.1.6 Fase 6: Rediseño de la página web

Esta fase se realiza a partir de una página web como base a partir de la cual se realizan las modificaciones oportunas para conseguir una interfaz más amigable mediante desplegables que faciliten la navegación por la misma. Con este fin, los lenguajes HTML y CSS interactúan juntos. Este paso es de duración variable, dependiendo de los cambios que se quieran aplicar y de su complejidad. Para este proyecto se calcula una carga de trabajo aproximada de 60 horas repartidas a lo largo de 3 semanas, que es la duración total.

1.1.7 Fase 7: Implementación en dispositivo móvil

La última fase para terminar la aplicación práctica de los servidores y códigos estudiados consiste en su adaptación a un dispositivo móvil, sin perder la fluidez de navegación que se consigue implementándolo en el ordenador. Para realizar los ajustes oportunos con este objetivo, se considera una duración total de unos 4 días con una carga de trabajo de 10 horas.

1.1.8 Fase 8: Redacción del proyecto

Para terminar, se procede a elaborar la redacción del trabajo, incluyendo todo el contenido de las fases anteriores de una forma clara y ordenada. Asimismo, se incluyen los posibles cambios en el índice mencionados en la fase 2. Aunque es una tarea muy larga, no necesita recursos especiales ya que únicamente se trata de plasmar lo desarrollado anteriormente. Por esta razón, la duración total es de una semana aproximadamente con una carga de trabajo de 25 horas.

1.2 Duración total

La duración total del proyecto es de 96 días.

1.3 Hitos

Tabla 1.1. Tabla de hitos

Tarea	Nombre	Comienzo
Hito 1	Aceptación índice	mar 05/05/20
Hito 2	Instalación de la aplicación	mar 16/06/20
Hito 3	Aceptación rediseño	mar 07/07/20
Hito 4	Aceptación implementación a dispositivo móvil	lun 13/07/20
Hito 5	Entrega	lun 20/07/20

1.4 Diagrama de Gantt

Tarea	Nombre	Duración
Fase	**Plan de proyecto**	**96 días**
Fase 1	Revisión de las tecnologías	8 sem.
Fase 2	Índice	2 días
Hito 1	Aceptación índice	0 días
Fase 3	**Revisión de códigos**	**20 días**
Fase 3.1.	Python	2 sem.
Fase 3.2	CSS	2 sem.
Fase 4	**Revisión servidores**	**5 días**
Fase 4.1	Apache	1 sem
Fase 5	Instalación servidor	5 días
Hito 2	Instalación de la aplicación	0 días
Fase 6	Rediseño de la página Web	3 sem.
Hito 3	Aceptación rediseño	0 días
Fase 7	Implementación en dispositivo móvil	4 días
Hito 4	Aceptación implementación a dispositivo móvil	0 días
Fase 8	Redacción del proyecto	1 sem
Hito 5	Entrega	0 días

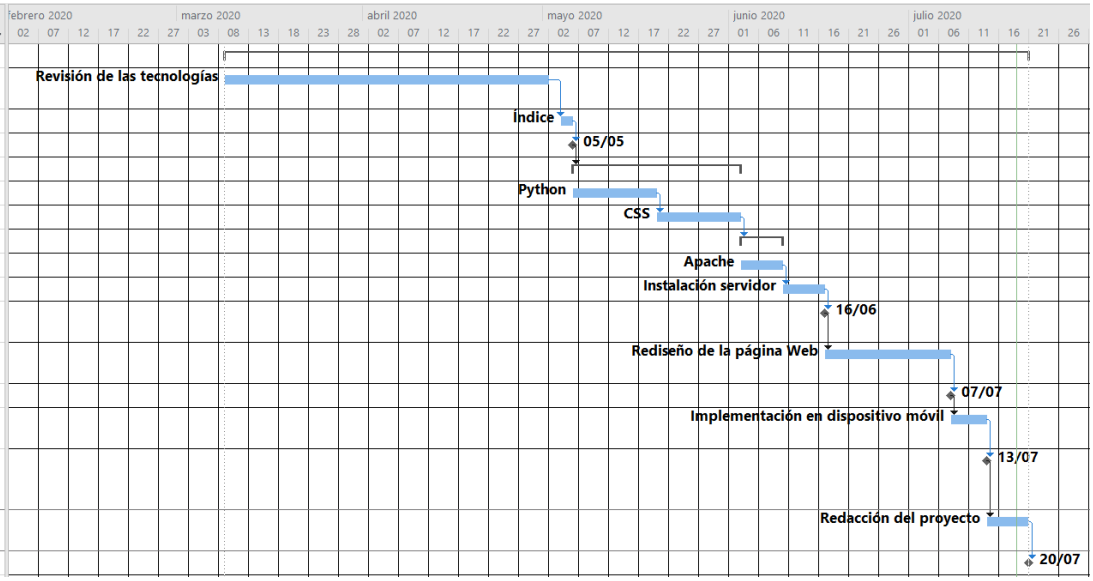


Figura 1.1. Diagrama de Gantt

ASPECTOS ECONÓMICOS

1 Descripción del presupuesto

En la ejecución de este proyecto únicamente es necesario incluir en el presupuesto los recursos siguientes: ingeniero sénior, ingeniero junior y amortización de ordenadores. De esta manera se realiza el siguiente presupuesto:

N.º Ref.	Cant. (Ud.)	Concepto	Precio unitario (€/Ud.)	Precio total (€)
1.		MANO DE OBRA		
1.1	20	Horas de Ingeniero Sénior	50	1000
1.2	300	Horas de Ingeniero Junior	30	9000
		TOTAL MANO DE OBRA		10000
2.		MATERIALES		
2.1	310	Horas de ordenador	0,7	217
		TOTAL MATERIALES		217
		TOTAL MANO DE OBRA + MATERIALES		10217
		Imprevistos (2%)		204,34
		TOTAL		10421,34
		IVA (21%)		2188,48
		PRESUPUESTO TOTAL		12609,82

CONCLUSIONES

En la realización de este trabajo se obtiene una metodología para la mejora y el rediseño de una página web básica. Dicho rediseño se realiza de forma que se simplifican elementos para hacer más sencilla y atractiva la navegación, incluyendo pestañas desplegables. No obstante, las modificaciones que se pueden realizar son múltiples, tanto para optimizar la funcionalidad de la aplicación como para conseguir una interfaz más estética. Por ejemplo, se pueden añadir iconos representativos de cada apartado que se desea visualizar o incluir métodos de seguridad de autenticación. Para ello, el diseño debe tener capacidad de añadir dichas mejoras para poder implementarlo.

En lo que al servidor web se refiere, se ha empleado Apache, ya que es el que más ventajas presenta frente a otros competidores. Éste es el más a nivel mundial dado que permite emplear distintos lenguajes en el lado del servidor (Python, Perl y PHP principalmente). Además, tiene la capacidad de incorporar conexiones seguras y utilizar URLs amigables. Otro competidor, como es Microsoft IIS, no presenta tantas ventajas y su principal problema es que únicamente funciona en Windows. Es posible que con este último servidor hubiera surgido una serie de problemas que con Apache no suceden.

Por otro lado, cabe destacar la relevancia que han adquirido en el tema del diseño los CSS. Anteriormente el código debía ser modificado tantas veces como el número de cambios en un mismo elemento reiterativo que se quisieran realizar. Por ejemplo, si se desea cambiar el color de un título que aparece en todas las páginas, habría que aplicar la mejora una vez en cada página. Esto resulta una tarea muy compleja y, por esta razón, se crean los CSS, que permiten realizar un solo cambio en la hoja de estilos que se implanta en todo el código. Es a partir de la creación de este lenguaje que comienzan los avances en cuanto al diseño de las webs. Es importante tener en cuenta que, aunque con una implementación básica ya se pueden cumplir los requisitos de diseño, es deseable optimizar lo máximo posible CSS. De esta manera no solo se aumenta la velocidad de navegación del usuario, sino que también permite que el CSS empleado sea reutilizable para futuras aplicaciones.

Por último, quedan claramente reflejadas las diferencias existentes en la implementación dependiendo del dispositivo a emplear (ordenador o móvil). Un código óptimo aplicable al ordenador es posible que provoque muchos problemas de visualización o funcionalidad en su versión móvil si no se realizan los cambios oportunos. Este es un factor muy importante y en el que se puede profundizar dado que, hoy en día, aunque los móviles son los soportes más empleados, existen todavía numerosas aplicaciones web que no cumplen los requisitos esperados por el usuario.

BIBLIOGRAFÍA

- D. Group, "Welcome! - The Apache HTTP Server Project", *Httpd.apache.org*, 2020. [en línea]. Disponible en: https://httpd.apache.org/. [Consulta: mayo 2020].

- "W3Schools Online Web Tutorials", *W3schools.com*, 2020. [en línea]. Disponible en: https://www.w3schools.com/. [Consulta: julio 2020].

- "Welcome to Python.org", *Python.org*, 2020. [en línea]. Disponible en: https://www.python.org/. [Consulta: julio 2020].

- "CSS", *Desarrolloweb.com*, 2020. [en línea]. Disponible en: https://desarrolloweb.com/home/css. [Consulta : abril 2020].

- "Server Fault", *Server Fault*, 2020. [en línea]. Disponible en: https://serverfault.com/. [Consulta: julio 2020].

- "Servidor Web: ¿Qué es? ¿Para qué sirve? - Infranetworking", *Infranetworking*, 2020. [en línea]. Disponible en: https://blog.infranetworking.com/servidor-web/. [Consulta: abril 2020].

- "Concepto y funcionamiento de CGI", *Diego.com.es*, 2020. [en línea]. Disponible en: https://diego.com.es/concepto-y-funcionamiento-de-cgi. [Consulta: mayo 2020].

- "¿Qué es el CGI?", *Maestros del Web*, 2020. [en línea]. Disponible en: http://www.maestrosdelweb.com/cgiintro/. [Consulta: junio 2020].

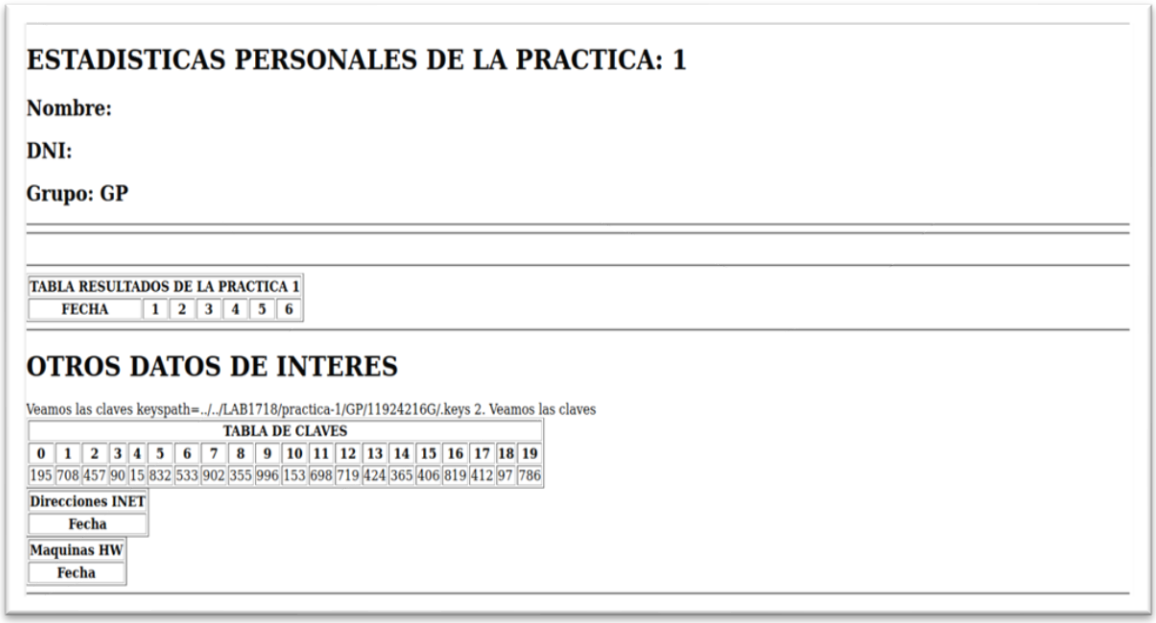
- "Guía Responsive: adapta 100% tu web a dispositivos móviles", *Aulacm.com*, 2020. [en línea]. Disponible en: https://aulacm.com/responsive-adapta-web-a-dispositivos-moviles/. [Consulta: julio 2020].

- "Nginx: ¿qué ventajas e inconvenientes tiene?", *Raiola Networks*, 2020. [en línea]. Disponible en: https://raiolanetworks.es/blog/nginx/. [Consulta: abril 2020].

- "Tipos de Servidores Web - Características, Ventajas y Desventajas", *Infranetworking*, 2020. [en línea]. Disponible en: https://blog.infranetworking.com/tipos-de-servidores-web/#Apache. [Consulta: abril 2020].

ANEXO

En este anexo se muestra el punto de partida del proyecto de implementación de la aplicación web usada.



ESTADISTICAS PERSONALES DE LA PRACTICA: 1

Nombre:

DNI:

Grupo: GP

TABLA RESULTADOS DE LA PRACTICA 1						
FECHA	1	2	3	4	5	6

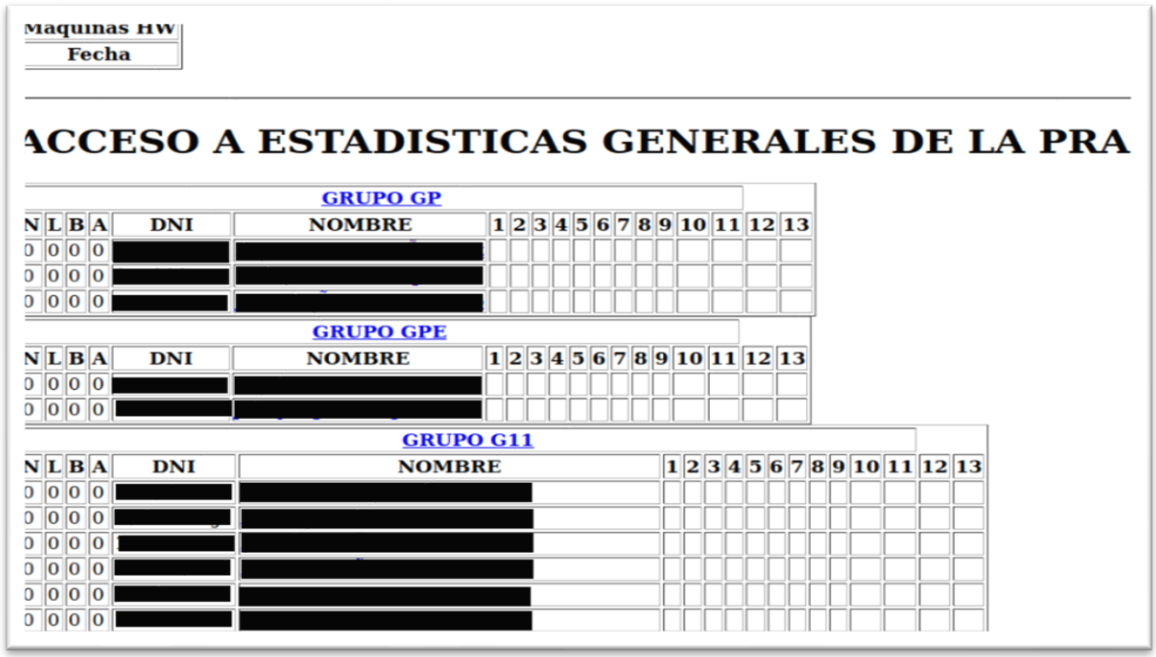
OTROS DATOS DE INTERES

Veamos las claves keyspath=../../LAB1718/practica-1/GP/11924216G/.keys 2. Veamos las claves

TABLA DE CLAVES																			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
195	708	457	90	15	832	533	902	355	996	153	698	719	424	365	406	819	412	97	786

Direcciones INET
Fecha

Maquinas HW
Fecha



Maquinas HW
Fecha

ACCESO A ESTADISTICAS GENERALES DE LA PRA

GRUPO GP

N	L	B	A	DNI	NOMBRE	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													

GRUPO GPE

N	L	B	A	DNI	NOMBRE	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													

GRUPO G11

N	L	B	A	DNI	NOMBRE	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													
0	0	0	0	[redacted]	[redacted]													