

Sumario

Modificaciones realizadas en este documento.....	1
Creación 26/10/23.....	1
Modificación 20/11/23.....	2
Comentarios.....	2
Calendario.....	3
Normas de obligado cumplimiento.....	3
1. Aplicación de gestión de incidencias/tareas - Básica.....	4
1.1. Ver la lista de tareas. (Adm. y ope.).....	5
1.1.1 Ver lista paginando resultados (medio).....	6
1.1.2 Ver lista de tareas pendientes de completar (avanzado).....	6
1.1.3 Ver información detallada de la tarea (avanzado).....	6
1.2. Añadir un nueva tarea (Adm.).....	6
1.3. Modificar datos de una tarea (Adm.).....	6
1.4. Eliminar una tarea (Adm.).....	7
1.5. Completar una tarea (operario) (avanzado).....	7
1.5.1 Completar tarea. Documentar el trabajo con archivos adjuntos (avanzado).....	7
1.6. Buscar o filtrar tareas utilizando distintos campos. (Avanzado).....	7
Reglas de obligado cumplimiento.....	8
Consejos prácticos.....	8
¿Como comenzar?.....	8
1.7 Diseño de aplicación modular usando Blade.....	9
2. Gestión de tareas mejorada.....	9
2.1 Validación de usuario.....	9
2.2. Validación de múltiples usuarios (avanzado).....	9
2.3. Diferenciando el tipo de usuario (roles).....	10
2.4. Recordando al usuario.....	10
2.4.1 Recordando credenciales (avanzado).....	10
2.5. Configuración de parámetros de la aplicación (avanzado).....	11
2.5. Instalador de la aplicación (avanzado).....	11
3. Adenda.....	12
3.1. Documentación de la aplicación.....	12
3.2. Control de versiones.....	12
Consideraciones finales.....	12
Comentarios y evaluación.....	12
Puntuación del proyecto.....	13
Instrucciones para entregar la práctica.....	15
Anexo I – Programas de generación automática de documentación.....	16
Doxygen.....	16
Apigen.....	16
Instalacion en Windows.....	16
Anexo II – Control de versiones.....	19

Modificaciones realizadas en este documento

Entrega inicial 16/10/23











- Envío del documento- INCOMPLETO-
- Pendiente de revisiones varias

Comentarios

El objetivo de esta práctica/proyecto es trabajar los contenidos que iremos viendo en clase incrustandolos en una aplicación real o algo que se le parezca. Trabajaremos con muchos problemas sencillos, pero que al estar todos juntos se volverán un poco complejos.

Esta proyecto tiene como objetivo crear un espacio de colaboración entre vosotros y con vuestro profesor. En todo momento podréis consultar a vuestro profesor sobre como abordar un determinado problema, el os dará indicaciones y si es preciso os proporcionará una solución.

En un primero momento utilizaremos la siguiente estructura de carpetas para almacenar los ficheros de la aplicación, luego cuando incorporemos el framework laravel utilizaremos la que proporciona el Framework:

Tipo	Elemento	Descripción
	app	Carpeta que contiene los ficheros de la aplicación. Controladores, vistas y modelos cada uno organizado en sus respectivas carpetas
	index.php	Este fichero arrancará la aplicación e implementará el controlador frontal
	config.php	En este fichero se incluirán parámetros de configuración de la aplicación, como: - Usuario y clave para acceder a la base de datos - Nombre de la base de datos Cualquier otra información que consideréis relevante como - ruta url de la aplicación - ruta absoluta dentro del servidor ... Todo lo anterior se almacenará en una variable/array/objeto de PHP que luego se utilizarán en la aplicación.
	 controllers  views  models ...	Carpetas en las que almacenaremos las distintas partes de la aplicación
	install	En esta carpeta se incluirá cualquier información que se considere relevante para instalar la aplicación.
	bd.sql	Fichero que contiene un script sql que crea la base de datos que utilizará el programa. En el script se creará - La base de datos - La estructura de las tablas - Los usuarios si utilizase alguno diferente
	Doc	Documentación de la aplicación.
	Assets  css  img  js	Imágenes, ficheros de estilo (css), fichero de script (js), y otro tipo de ficheros que se utilicen para el diseño de la interfaz.



Los apartados tienen como propósito obligaros a que trabajéis los contenidos vistos así como que profundicéis en vuestros conocimientos sobre la metodología de la programación (como resolver los problemas). Esta práctica está pensada para que la realicéis en clase completando el trabajo en vuestra casa. Ante cualquier duda acerca de cómo afrontar un problema deberíais preguntar al profesor al respecto para que el os oriente.

Los problemas, **obligatoriamente deberéis enseñaroslos al profesor funcionando en clase antes de la fecha tope de entrega.**

Al entregar la práctica deberéis rellenar un cuestionario, y luego en clase explicar y demostrar el funcionamiento de la práctica al profesor.

Calendario

10(M), 11 (X), 12 (J) o 13 (V) de diciembre	Examen evaluación
10 de diciembre	Entrega de la práctica – Realización de un cuestionario sobre las tareas realizadas. Si el examen es el 10 se podrá una fecha anterior de entrega
11 al 18 de diciembre	Defender la práctica y explicar al profesor

Normas de obligado cumplimiento

Para la realización de la aplicación será **obligatorio el uso del patrón MVC** en el desarrollo de los problemas en PHP. Se valorará igualmente cualquier uso de otro tipo de patrones de software disponibles.

El acceso a la base de datos se realizará utilizando **un patrón Singleton** para guardar la conexión a la base de datos.

Los problemas deberán ser documentados con comentarios, indicando en cada fichero con código php el autor, la fecha de creación, la versión del fichero.

Se documentará cada una de las funciones y variables de clase que se creen utilizando el formato algún programa de documentación automática (ApiGen, DoxyGen u otros).

Básicamente lo que debéis hacer es abrir comentarios de tipo bloque `/** . . . */` delante de cada función y variable de clase, y el entorno se encargará de generaros automáticamente las [etiquetas](#) (@ . . .) pertinentes.

En esta primera práctica utilizaremos el framework Laravel SOLAMENTE para facilitarnos el uso del gestor de plantillas Blade, y utilizar su controlador frontal para gestionar las peticiones. Aunque el framework dispone de mecanismos para abordar lo siguiente no se autorizará, ni computará para la calificación

- El uso del mecanismo de sesiones de Laravel
- El uso del ORM de laravel para acceder a la base de datos

Si se usan estos mecanismos se entenderá como no realizado el apartado

1. Aplicación de gestión de incidencias/tareas - Básica

La empresa de albañilería Bunglebuild S.L. requiere nuestros servicios para crear una aplicación que le permita mejorar la gestión de las obras y tareas con las que trabaja la empresa. La empresa ha aumentado enormemente su clientela y la carga de trabajo en los últimos tiempos por lo que precisa implementar un gestor de incidencias/tareas que facilite el control y seguimiento de los trabajos que se tienen que realizar. Para ello precisa de una aplicación web que permita llevar dicho control, permitiendo a cada operario en cada momento saber las tareas que tiene pendientes y notificar sobre cualquier incidencia o contratiempo que se produzca en la realización de las mismas.

La aplicación tendrá los siguientes tipos (roles) de usuarios:

- Los **administrativos**: serán los encargados de crear nuevas tareas y supervisar las mismas.
- Los **operarios** cambiarán el estado de las tareas y realizarán anotaciones en las mismas, limitándose solamente a cambiar los campos relativos a su trabajo.

Debido a que estamos comenzando a trabajar, en una primera aproximación a la solución realizaremos una interfaz común para clientes, administrativos y operarios, pero teniendo presente que las operaciones a realizar son diferentes y puede que en un futuro estén separadas.

Nuestra aplicación web nos permitirá realizar las siguientes operaciones:

- Ver la lista de incidencias/tareas. (*Adm. y ope.*)
- Añadir una nueva incidencia/tarea. (*Adm.*)
- Modificar datos de una incidencia/tarea. (*Adm.*)
- Eliminar una tarea. Confirmando la operación para evitar errores. (*Adm.*)
- Cambiar el estado de una incidencia/tarea
- Completar una tarea incluyendo anotaciones si se precisan (*ope.*)
- Buscar o filtrar tareas utilizando distintos campos. (*Adm. y ope.*)

La información que almacenaremos sobre las tareas será la siguiente:

- *NIF o CIF* de la persona sobre la que realizaremos la factura.
- *Persona de contacto*: Nombre y apellidos de la persona.
- *Teléfono/s contacto*: Nº de teléfono de contacto de la persona de contacto.
- *Descripción*: Texto descriptivo identificativo de la tarea
- *Correo electrónico*: Correo electrónico de la persona de contacto.
- *Dirección*: Dirección donde debemos ir a realizar la tarea.
- *Población*
- *Código postal*
- *Provincia*: Para este campo utilizaremos un <select>. En este campo se almacenará un código numérico acorde a las [indicaciones del INE](#). Dicho código será el mismo que los dos primeros dígitos del código postal.
- *Estado*: Estado en el que se encuentra la tarea (B=Esperando ser aprobada, P=Pendiente, R=Realizada, C=Cancelada, ...)
- *Fecha de creación de la tarea*: Fecha en la que se ha creado la tarea. Este campo se generará automáticamente. Una opción sencilla sería usar un disparador de la base de datos.
- *Operario encargado*: Nombre o identificación del operario encargado de la realización de la tarea. La lista de operarios se mostrará con un campo select, y será alguno de los que tengamos registrados en nuestra empresa (no es preciso que estén almacenados en la BBDD)
- *Fecha de realización*: Fecha en la que se realizará la tarea.
- *Anotaciones anteriores*: Cualquier texto que se desee incluir para explicar el trabajo a realizar antes de comenzarlo.
- *Anotaciones posteriores*: Anotaciones realizadas por los operarios después de realizar la tarea.

- Fichero resumen de tareas realizadas. Para cada tarea realizada se permitirá que el operario pueda adjuntar un fichero con indicaciones del trabajo realizado. Dicho fichero se almacenará en una carpeta dentro del servidor y no será accesible desde ninguna ruta url en el cliente sin previamente haberse validado.
- *Fotos del trabajo realizado.* Se incluyan fotos que justifiquen el trabajo que se ha realizado.

La información que contiene estos campos debe cumplir las siguientes reglas:

- Los campos descripción y persona de contacto debe tener algún valor
- El campo CIF o NIF deberá tener un valor correcto. Para esto podréis hacer uso de los múltiples algoritmos existentes en la web que nos facilitan el trabajo. [Ver algoritmos](#)
- El teléfono de contacto debe tener un valor y si existe debe tener un formato válido, sólo números, y caracteres de separación (espacio, guión, y otros que estiméis oportuno).
- El código postal, si existe, debe tener un formato válido, 5 números.
- La provincia debe ser alguna de las existentes en España. Se debe permitir seleccionar la provincia de una lista desplegable.
- El correo electrónico es obligatorio y debe tener un formato correcto.
- La fecha de realización debe tener un formato válido y ser posterior a la fecha actual. Se debe filtrar la fecha en el servidor para comprobar que la fecha es válida.
- La fecha de creación no se podrá modificar, aunque si aparecerá en las consultas y modificaciones.

**** Obligatorio ****

Todos los campos se deben filtrar en el servidor, mostrando el pertinente error en el formulario y **guardando el valor enviado** para que se pueda **editar** y corregir.

Nota: Se desaconseja el uso de controles y atributos avanzados de HTML5 (required, max, etc), pues si bien serán los más indicados en una aplicación real, en la nuestra lo único que harán será dificultar la verificación de nuestro filtrado en el servidor. Si los utilizáis incluid alguna opción para permitir desactivarlos.

La aplicación mostrará una página inicial, desde la que se podrán realizar todas las operaciones antes mencionadas. Dicha página inicial puede ser la lista de tareas, un menú de opciones, etc. Esto queda a vuestro criterio de diseño, aunque siempre debéis considerarlo como se va a trabajar con vuestra aplicación y diseñarla de forma que facilite el trabajo a sus usuarios..

Las operaciones a realizar con más detalle consistirán en lo siguiente.

Nota: Siempre que haya que mostrar fechas se utilizará el formato habitual de nuestra zona/cultura, que hasta hoy es dd/mm/aaaa

1.1. Ver la lista de tareas. (Adm. y ope.).

Mostrará la lista de tareas ordenada de forma **descendente** por fecha de realización.

En esta lista se mostrará la información más relevante de las tareas, descripción, persona de contacto, fecha de realización, etc.

Para cada entrada permitiréis ver la información completa y detallada de la tarea.

Sería interesante que desde aquí pudieséis lanzar también las operaciones de borrado, modificación, ver detalles de la tarea.

1.1.1 Ver lista paginando resultados (medio)

Se deberá **paginar** la lista de elementos mostrados.

Para la **paginación de los resultados** que se muestren en lista. Deberéis incluir:

- Enlace para ir a la página siguiente y anterior.
- Enlace para ir a la primera y última página.
- Debe mostrar en que página nos encontramos.

Opcionalmente mostrará también:

- Indicación del número de páginas.
- Mecanismo que nos permita ir a una página en concreto.

Mejoras:

- Se podrá permitir ordenar por otros campos.

1.1.2 Ver lista de tareas pendientes de completar (avanzado)

Como mejora en esta funcionalidad, se incluirá una opción adicional <<Listar tareas pendientes>>, que nos permitirá filtrar las tareas de forma que solo muestre las que están pendientes, y que tendrían que realizar los operarios.

1.1.3 Ver información detallada de la tarea (avanzado)

En la lista tan solo mostraremos los campos principales de las tareas. Estos serán los campos que a vuestro criterio resulten más útiles para los usuarios que trabajen con ellos.

Para ver la información completa de toda la tarea permitiremos abrir una nueva ventana donde se mostrará esta de forma individual y bien presentada.

1.2. Añadir un nueva tarea (Adm.)

Esa opción nos permitirá crear una nueva incidencia/tarea. Antes de guardar la información debemos comprobar que los datos son correctos, acorde a los requisitos establecidos.

Si hay errores al crear la nueva tarea **se permitirá al usuario ver los valores que ha introducido** para permitirle modificarlos y corregirlos.

No se permitirá guardar datos que no cumplan las restricciones impuestas.

Nota:

- Observad que el filtrado de los campos será similar para la adición y la modificación. Intentad reutilizar código.

1.3. Modificar datos de una tarea (Adm.)

Desde esta opción permitiremos mostrar los campos de una tarea, y modificarlos. Se mostrará un formulario con los datos actuales, los cuales cual podremos cambiar.

Se deben filtrar los campos antes de proceder a guardar cualquier tipo de dato al igual que hemos

hecho con la operación de añadir una nueva tarea.

1.4. Eliminar una tarea (Adm.)

Esta operación permitirá eliminar una tarea de la base de datos. Previo a la eliminación se procederá a confirmar la operación, interactuando con el servidor. (NO sirve en javascript).

Al eliminar una tarea, se mostrará una página de confirmación en la que se muestren los datos más importantes/relevantes de la tarea y se pregunte si se desea borrar o no.

1.5. Completar una tarea (operario) (avanzado).

Esta operación me permitirá cambiar el estado de una tarea y realizar las anotaciones oportunas sobre la misma. Para esta operación tan solo se mostrarán los datos de la tarea y se solicitará que se marque la tarea como completada, cancelada, realizando las anotaciones que se consideren oportunas. El operario **no se deberá poder modificar ningún campo**, salvo las anotaciones y el estado.

El estado se seleccionará preferiblemente con botones de radio, marcando por defecto la opción completada.

Esta acción en un futuro será realizada por los operarios, la pantalla debe ser diferente de la que se utilice en la modificación pues solo se deberán permitir cambiar los campos fecha de realización, estado de la tarea y anotaciones posteriores.

1.5.1 Completar tarea. Documentar el trabajo con archivos adjuntos (avanzado)

La empresa, para evitar reclamaciones o malentendidos desea tener pruebas del trabajo realizado, fotos, videos, documentos, etc. Por este motivo se desea poder adjuntar a la tarea, un fichero donde podremos guardar pruebas gráficas o de otro tipo.

Se requiere que subáis un fichero, aunque podrían ser más si así lo deseais, aunque esto hace más compleja la tarea.

Desde esta operación (formulario) se permitirá subir el fichero adjunto a la tarea para guardarlo en nuestro servidor.

El fichero subido podrá ser consultado/descargado desde la información de la tarea.

Nota: Normalmente los ficheros no se guardan en la base de datos debido al espacio que ocupan y que pueden ser fácilmente indexados por su nombre. Por este motivo, una forma sencilla de guardar el fichero será asignarle como nombre el **id** de la tarea a la que está seleccionada y en un campo de la tabla guardar el nombre original de dicho fichero, si nos interesa.

1.6. Buscar o filtrar tareas utilizando distintos campos. (Avanzado)

Con esta operación permitiremos que el usuario pueda buscar o filtrar la lista de pedidos atendiendo al valor de diferentes campos. Se deberán soportar al menos 3 campos, y se debe considerar que la búsqueda podrá incluir criterios de comparación como igual, contiene, mayor, menor, etc. en los campos que proceda.



El formulario mostrará al menos 3 campos donde podremos seleccionar el criterio de comparación y el valor con el que se compara. Si dejamos en blanco el campo valor no se considerará a la hora de crear la consulta SQL.

Campo	Criterio	Valor
Lista desplegable con campos posibles a comparar o nombre de campo fijo	= > <
Lista desplegable con campos posibles a comparar o nombre de campo fijo	= > <
....

El objetivo de esta apartado es que construyais de forma dinámica una sentencia SQL en función de varios campos.

Reglas de obligado cumplimiento

El trabajo anterior se debe realizar utilizando los siguientes elementos arquitectónicos

- Usar [patrón de diseño MVC](#) – Modelo Vista Controlador –
- Se utilizará el [patrón de diseño Singleton](#) para gestionar la conexión a la base de datos.
- Se utilizará el [patrón de diseño Controlador Frontal](#) para gestionar las peticiones. Para esto haremos uso del framework Laravel
- Deberéis documentar el código utilizando comentarios que luego permitan la generación de la documentación de forma automática.

Consejos prácticos

- Será conveniente que tengáis un repositorio GIT (asociado a GitHub) que os permita realizar un seguimiento y recuperación del trabajo realizado.
- Aunque aun no sabéis completar todo el trabajo que debéis realizar, ya podéis ir trabajando muchas de las partes para luego crear nuestra aplicación.
- Antes de comenzar deberéis definir con claridad el modelo de datos que estáis utilizando – Esquema de base de datos-. Para esto puede resultaros muy útil la herramienta [MySQL Workbench](#)

¿Como comenzar?

Aunque aun no tenéis los suficientes conocimientos para hacer la aplicación completa, si que podéis comenzar a trabajar realizando operaciones que ya conocéis. Por ejemplo, podéis ir diseñando todos los formularios con los que se trabaja y tratando de ir haciendo operaciones de filtrado.

Aunque luego puede que tengáis que hacer modificaciones a vuestro código, todo este trabajo ya lo tenéis hecho y os facilitará enormemente completar el proyecto.



1.7 Diseño de aplicación modular usando Blade.

Se desea que nuestra aplicación web se muestre utilizando un diseño web modular de forma que en todo momento estemos viendo un encabezado, menú y pie común a toda la aplicación. Como el de la siguiente figura.

ENCABEZADO	
Menú Lateral	CUERPO
PIE	

Debéis utilizar un esquema similar al anterior, aunque no necesariamente tiene que ser igual.

Lo importante es que en todo momento el usuario sepa donde se encuentra, y tenga opción de realizar las opciones principales sin problemas.

Para hacer este tipo de presentación, no necesitaríamos un motor de plantillas, aunque este nos lo simplificará enormemente.

Utilizaréis el motor de plantillas Blade, que explicaremos en clase, y que más adelante utilizaremos igualmente en la 2ª Evaluación cuando trabajemos con Laravel .

2. Gestión de tareas mejorada

2.1 Validación de usuario.

Se desea ampliar la aplicación de forma que solo permita acceder a ella validandonos previamente con un usuario y una clave.

Solo se podrá acceder a la funcionalidad de la aplicación si previamente se ha validado el usuario introduciendo correctamente su usuario y clave.

En caso de intentar acceder a alguna de las funciones, sin haberse validado previamente, se mostrará la pantalla que solicita el usuario y clave.

En el encabezado mostraréis en todo momento en la esquina superior derecha la hora en la que ha iniciado la sesión el usuario, y pondréis un enlace (texto, imagen o ambos) que os permitirá finalizar la sesión.

Para este apartado solo es preciso que validéis con un usuario, no es preciso hacer uso de la base de datos. Los datos usuario y clave los podréis almacenar en el código como constantes o variables.

2.2. Validación de múltiples usuarios (avanzado).

Ampliaremos el apartado anterior de forma que podámos tener múltiples usuarios, los cuales almacenaremos en una tabla. Podrémos realizar con los usuarios las siguientes operaciones:

- Añadir un usuario. [solo Adm.]
- Eliminar un usuario. [solo Adm.]
- Editar un usuario: cambiar usuario o clave.
- Listar usuarios existentes. [solo Adm.]

Este apartado ampliará el anterior almacenando en una tabla los datos de los usuarios. Dicha tabla será manipulada por los administradores.

2.3. Diferenciando el tipo de usuario (roles).

Ampliaremos el apartado anterior (o el 2.1 si no se ha realizado 2.2) de forma que ahora diferenciaremos el tipo de usuario que se ha validado, restringiendo las operaciones que este puede realizar.

Se recuerda que tendremos los siguientes tipos de usuarios:

- Operarios: solo podrán realizar las operaciones que tienen asignadas, y tendrán ocultos o deshabilitados los enlaces a las operaciones que están asignadas al administrador.
- Administrador: no podrá realizar operaciones de los operarios.

En el encabezado de la aplicación se mostrará con claridad si el usuario es de tipo Operario o Administrador, bien con texto o utilizando iconos.

2.4. Recordando al usuario.

Los empleados de la empresa, nos han solicitado que cuando trabajen con el programa desean que cuando les pida el usuario y clave para identificarse, automáticamente nos recuerde y nos escriba el nombre del último usuario que se identifico, de forma que ellos no tengan que escribirlo.

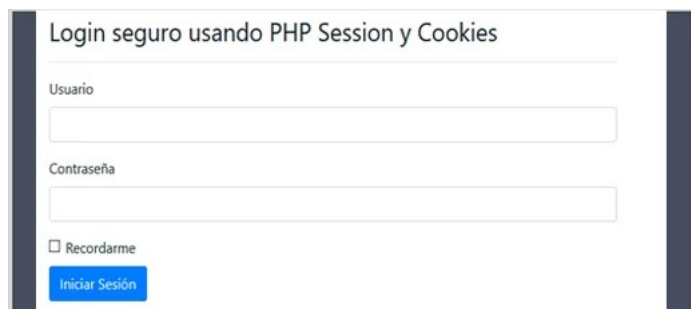
Por lo tanto, en el campo de edición "usuario" aparecerá de forma automática el nombre del último usuario.

2.4.1 Recordando credenciales (avanzado)

Los administrativos de la empresa indican que trabajan siempre con el mismo ordenador el cual está protegido por contraseña, y no desean tener que identificarse todos los días en la aplicación, por lo que solicitan que al igual que gmail, facebook y otros cuando arranquen la aplicación está se valide automáticamente con su usuario.

Por cuestiones de seguridad nos indica el jefe que solamente se recordará 3 días.

Para activar esta opción, se incluirá un checkbox en el formulario el cual si lo activamos hará que



se almacene algo que permita no tener que validarse.

Puede resultaros útil el siguiente enlace: [Login con la opción recordarme PHP Session y Cookies](#)

Nota: Aunque una opción que os podéis plantear es almacenar los datos del usuario y la clave en una cookie, esta opción es altamente insegura. Solo tendría sentido si el contenido de la cookie estuviese encriptado Véase:

- [Criptografía en PHP](#)

- [openssl_encrypt](#)

2.5. Configuración de parámetros de la aplicación (avanzado).

Con esta opción se pretende ampliar la aplicación de forma que podamos configurar determinados aspectos de su funcionamiento, como por ejemplo:

- Valor por defecto que se muestra en los campos al crear un nuevo elemento. Por ejemplo en los campos, provincia, población, zona.
- Indicar el nº de elementos que se muestran en la lista.
- Configurar el tiempo que mantenemos la sesión abierta si que se use (<http://blog.controlzeta.net/?p=500>)
- Permitir seleccionar diferentes temas (colores, tipos, etc) al mostrar la aplicación.

Cualquier otro valor que consideréis que pueda ser configurable.

Este apartado se puede realizar de varias maneras:

1. Crear una nueva tabla en la que iremos almacenando el diferente valor utilizado para cada uno de los parámetros de configuración. Dicha tabla se leerá cada vez que se cargue una página y se creará un array/objeto que contendrá los valores que hay seleccionados.

Este sistema tiene el inconveniente de que es más complejo y sobrecarga el sistema accediendo innecesariamente a la base de datos para cada petición.

2. Crear un fichero de configuración en el que almacenaremos todos los valores deseados utilizando un array, objeto o variables sueltas. Dicho fichero se incluirá en todas las páginas que mostremos.

Cuando deseemos modificar la configuración lo que haremos será crear un nuevo fichero desde nuestra aplicación. Nuestra aplicación generará un fichero cuyo contenido será texto que contendrá sentencias de PHP.

3. Serializar los valores de configuración (objeto o array) y almacenarlo en la base de datos o en un fichero.

2.5. Instalador de la aplicación (avanzado)

Un instalador en una aplicación Web es una aplicación que permite crear/configurar los parámetros de una aplicación web. Los parámetros configurables son:

- Solicitará los datos para acceder a la base de datos (usuario, clave) y nombre de esquema. Dichos datos los almacenará en un fichero de configuración. En vuestro caso esto no será preciso pues ya lo habréis creado con el editor en el fichero "app/config.php"
- Creación de la estructura de la base de datos: El instalador creará la estructura de las tablas e inicializará los datos que se precisen. Puede que incluso llegue a crear la base de datos, aunque esto es menos habitual.
- Si es preciso el instalador modificará otros parámetros que tuviese la aplicación.

Se precisa realizar un instalador para la aplicación. El instalador estará situado en la carpeta "install", y arrancará automáticamente "index.php". Cogerá los datos de configuración del fichero "app/config.php" y procederá a la creación e inicialización de la base de datos.

En el fichero config.php tendremos que poder definir:

- Ubicación del servidor de base de datos.
- Usuario que accede a la base de datos.
- Clave del usuario que accede a la base de datos.
- Base de datos con la que se trabajará.

El instalador borrará todas las tablas que existan en la base de datos y luego creará la estructura de las tablas con las que trabajará. Para ello puede que preciséis obtener la lista de tablas existentes en la base de datos y luego ir borrando cada una.

3. Adenda

3.1. Documentación de la aplicación.

Deberéis generar la documentación de la aplicación realizada incluyendo los comentarios pertinentes y luego generando los documentos de forma automatizada como se indica en el Anexo I.

3.2. Control de versiones.

Se valorará que el alumno haya ido publicando las diferentes revisiones de su código utilizando GIT en algún repositorio público (tipo GIT) al que pueda acceder el profesor.

Deberéis proporcionar la dirección de acceso o los datos que sean preciso para comprobarlo.

Consideraciones finales

- En el desarrollo web, cuando diseñamos una tabla para una base de datos siempre es conveniente identificar los registros mediante un campo "id" de tipo numérico que se generará automáticamente, el cual será la clave principal. Esto nos facilitará mas adelante la codificación de la funcionalidad de la aplicación.
- En la web Gnome-look <http://gnome-look.org/> podéis encontrar múltiples paquetes de iconos gratuitos que os permitirán realizar una aplicación con una presentación bonita y coherente.
- Para la presentación os recomiendo que utilicéis algún framework de CSS como [Bootstrap](#), [Materialize](#), etc. que os simplificará el trabajo y mejorará sustancialmente la presentación sin mucho esfuerzo.
- Como iconos también podéis hacer uso de los que se proporcionan en <https://fontawesome.com/icons?d=gallery> y que se incluyen con etiquetas `<i class="fas fa-ad"></i>`

Comentarios y evaluación

- La copia de todo o parte del ejercicio supondrá la inmediata eliminación de la parte copiada. Se dividirá la nota de los implicados entre el número de copias. Igualmente en caso de identificar a la persona que ha copiado se le penalizará en su calificación.
- Cada alumno enseñará individualmente el funcionamiento de la aplicación al profesor. Antes de la fecha de entrega.

Puntuación del proyecto

* en revisión *

Con nuestro proyecto pretendemos garantizar que cumplis los requisitos mínimos exigidos en el currículo, que se establecen a través de Resultados de aprendizaje.

UT	Resultados de aprendizaje y C.E.	Apartados asociados
UT 1	<p>1. Selecciona las arquitecturas y tecnologías de programación Web en entorno servidor, analizando sus capacidades y características propias.</p> <p><i>Se han caracterizado y diferenciado los modelos de ejecución de código en el servidor y en el cliente Web.</i></p> <p><i>Se han reconocido las ventajas que proporciona la generación dinámica de páginas Web y sus diferencias con la inclusión de sentencias de guiones en el interior de las páginas Web.</i></p> <p><i>Se han identificado los mecanismos de ejecución de código en los servidores Web.</i></p> <p><i>Se han reconocido las funcionalidades que aportan los servidores de aplicaciones y su integración con los servidores Web.</i></p> <p><i>Se han identificado y caracterizado los principales lenguajes y tecnologías relacionados con la programación Web en entorno servidor.</i></p> <p><i>Se han verificado los mecanismos de integración de los lenguajes de marcas con los lenguajes de programación en entorno servidor.</i></p> <p><i>Se han reconocido y evaluado las herramientas de programación en entorno servidor.</i></p>	<p>Mostrar aplicación en:</p> <ul style="list-style-type: none"> - Servidor local con XAMP o similar - una máquina virtual del equipo - En un entorno con Docker
UT2	<p>2. Escribe sentencias ejecutables por un servidor Web reconociendo y aplicando procedimientos de integración del código en lenguajes de marcas.</p> <p><i>Se han reconocido los mecanismos de generación de páginas Web a partir de lenguajes de marcas con código embebido.</i></p> <p><i>Se han identificado las principales tecnologías asociadas.</i></p> <p><i>Se han utilizado etiquetas para la inclusión de código en el lenguaje de marcas.</i></p> <p><i>Se ha reconocido la sintaxis del lenguaje de programación que se ha de utilizar.</i></p> <p><i>Se han escrito sentencias simples y se han comprobado sus efectos en el documento resultante.</i></p> <p><i>Se han utilizado directivas para modificar el comportamiento predeterminado.</i></p> <p><i>Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.</i></p> <p><i>Se han identificado los ámbitos de utilización de las variables.</i></p>	Toda la aplicación
UT3	<p>3. Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.</p> <p><i>Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.</i></p> <p><i>Se han utilizado bucles y se ha verificado su funcionamiento.</i></p> <p><i>Se han utilizado «arrays» para almacenar y recuperar conjuntos de datos.</i></p> <p><i>Se han creado y utilizado funciones.</i></p> <p><i>Se han utilizado formularios Web para interactuar con el</i></p>	<p>Toda la aplicación</p> <p>1.2. Añadir una nueva tarea (Adm.)</p> <p>1.3. Modificar datos de una tarea (Adm.)</p> <p>1.4. Eliminar una tarea (Adm.)</p> <p>1.5. Completar una tarea (operario) (avanzado).</p>

UT	Resultados de aprendizaje y C.E.	Apartados asociados
	<p>usuario del navegador Web. Se han empleado métodos para recuperar la información introducida en el formulario. Se han añadido comentarios al código.</p>	<p>1.5.1 Completar tarea. Documentar el trabajo con archivos adjuntos (avanzado) 1.6. Buscar o filtrar tareas utilizando distintos campos. (Avanzado) 3.1. Documentación de la aplicación.</p>
UT4	<p>4. Desarrolla aplicaciones Web embebidas en lenguajes de marcas analizando e incorporando funcionalidades según especificaciones.</p> <p>Se han identificado los mecanismos disponibles para el mantenimiento de la información que concierne a un cliente Web concreto y se han señalado sus ventajas. Se han utilizado sesiones para mantener el estado de las aplicaciones Web. Se han utilizado «cookies» para almacenar información en el cliente Web y para recuperar su contenido. Se han identificado y caracterizado los mecanismos disponibles para la autenticación de usuarios. Se han escrito aplicaciones que integren mecanismos de autenticación de usuarios. Se han realizado adaptaciones a aplicaciones Web existentes como gestores de contenidos u otras. Se han utilizado herramientas y entornos para facilitar la programación, prueba y depuración del código.</p>	<p>2.1 Validación de usuario. 2.2. Validación de múltiples usuarios (avanzado). 2.3. Diferenciando el tipo de usuario (roles). 2.4. Recordando al usuario. 2.4.1 Recordando credenciales (avanzado) 2.5. Instalador de la aplicación (avanzado)</p>
UT5	<p>5. Desarrolla aplicaciones Web identificando y aplicando mecanismos para separar el código de presentación de la lógica de negocio.</p> <p>Se han identificado las ventajas de separar la lógica de negocio de los aspectos de presentación de la aplicación. Se han analizado tecnologías y mecanismos que permiten realizar esta separación y sus características principales. Se han utilizado objetos y controles en el servidor para generar el aspecto visual de la aplicación Web en el cliente. Se han utilizado formularios generados de forma dinámica para responder a los eventos de la aplicación Web. Se han identificado y aplicado los parámetros relativos a la configuración de la aplicación Web. Se han escrito aplicaciones Web con mantenimiento de estado y separación de la lógica de negocio. Se han aplicado los principios de la programación orientada a objetos. Se ha probado y documentado el código.</p>	<p>Toda la aplicación 1.7 Diseño de aplicación modular usando Blade. Usa patrón singleton Usa patrón MVC y separa lógica de negocio Usa patrón controlador frontal (Slim o similar) 2.5. Configuración de parámetros de la aplicación (avanzado). 3.2. Control de versiones</p>
UT6	<p>6. Desarrolla aplicaciones de acceso a almacenes de datos, aplicando medidas para mantener la seguridad y la integridad de la información.</p> <p>Se han analizado las tecnologías que permiten el acceso mediante programación a la información disponible en almacenes de datos. Se han creado aplicaciones que establezcan conexiones con bases de datos. Se ha recuperado información almacenada en bases de datos. Se ha publicado en aplicaciones Web la información</p>	<p>Usa patrón singleton Clase de acceso a datos Operaciones de 1.1. Ver la lista de tareas. (Adm. y ope.). 1.1.1 Ver lista paginando resultados (medio) 1.1.2 Ver lista de tareas pendientes de completar (avanzado) 1.1.3 Ver información detallada de la</p>

UT	Resultados de aprendizaje y C.E.	Apartados asociados
	<p>recuperada.</p> <p>Se han utilizado conjuntos de datos para almacenar la información.</p> <p>Se han creado aplicaciones Web que permitan la actualización y la eliminación de información disponible en una base de datos.</p> <p>Se han utilizado transacciones para mantener la consistencia de la información.</p> <p>Se han probado y documentado las aplicaciones.</p>	<p>tarea (avanzado)</p> <p>1.2. Añadir un nueva tarea (Adm.)</p> <p>1.3. Modificar datos de una tarea (Adm.)</p> <p>1.4. Eliminar una tarea (Adm.)</p> <p>1.5. Completar una tarea (operario) (avanzado).</p> <p>1.5.1 Completar tarea. Documentar el trabajo con archivos adjuntos (avanzado)</p> <p>1.6. Buscar o filtrar tareas utilizando distintos campos. (Avanzado)</p>

Para la a corrección y puntuación de cada apartado se tendrán en cuenta también los siguientes parámetros:

- Funcionalidad: Que el programa realice lo que se pide
- Estilo de programación: que el código del programa sea fácilmente entendible y modificable por otras personas. Para ello deberá regirse por las directrices de la programación estructurada, la programación orientada a objetos y utilizar patrones de diseño de software.
- Interfaz: Que el programa que utilicen recursos gráficos que faciliten la interacción con el usuario y se usen los recursos que proporciona el entorno de desarrollo.

Instrucciones para entregar la práctica

La práctica la subiréis a la tarea de la moodle como un fichero comprimido, en el que incluiréis todos los ficheros que componen la práctica.

Anexo I – Programas de generación automática de documentación

Existen una serie de programas que nos generarán automáticamente documentación a partir de nuestro código fuente, como pueden ser:

- **PHPDoc** (<http://es.wikipedia.org/wiki/PHPDoc>) del cual podréis obtener más información en [este artículo](#).
Podéis obtener una información más detallada en la [web de PHPDocumentator](#).
- Otro interesante programa, que será el que se recomienda utilizar es *ApiGen*. El cual se podrá integrar fácilmente en netbeans como indicamos a continuación.
- Por su sencillez **DoxyGen** es la opción recomendada
- Comparison of documentation generators
http://en.wikipedia.org/wiki/Comparison_of_documentation_generators

Estos utilizan una [serie de etiquetas](#), las cuales podréis consultar aquí (<http://www.phpdoc.org/docs/latest/for-users/list-of-tags.html>).

Doxygen

[Documenting PHP with Doxygen: The Pros and Cons](#)

ApiGen

Instalacion en Windows

Fuente: <http://www.apigen.org/>

ApiGen es un programa hecho en PHP que viene en formato "phar"

¿Qué es phar? ([más información ...](#))

Un archivo phar proporciona una forma para distribuir una aplicación PHP completa en un único fichero y ejecutarla desde ese mismo fichero sin necesidad de extraerlo en el disco. Además, los archivos phar pueden ser ejecutados por PHP fácilmente al igual que cualquier otro fichero, tanto desde la línea de comandos como desde un servidor web.

[Decargamos el programa](#) desde su sitio web <http://www.apigen.org>:

Nota: se recomienda que incluyáis la ruta de PHP en el PATH y creéis este comando en el directorio raíz de vuestro proyecto.

Crea un fichero de proceso por lotes "apigen.bat" que te permitirá manejar la aplicación más fácilmente en lugar de utilizar directamente el fichero en php "apigen.phar":

```
@rem *** Instrucciones ****  
@rem Descarga "apigen.phar" (http://www.apigen.org/) y copialo en una carpeta, la que
```



```
quieras.  
@rem Inicializa las variables PHP_PATH y APIGEN_PATH  
@rem - PHP_PATH = ruta del fichero "php.exe" que estará en la carpeta php de XAMMP  
@rem - APIGEN_PATH = ruta del fichero "apigen.phar"  
  
@set PHP_PATH=D:\programas\xampp-win32-5.5.27-1-VC11\php  
@set APIGEN_PATH=D:\programas\xampp-win32-5.5.27-1-VC11\php  
  
%PHP_PATH%\php.exe %APIGEN_PATH%\apigen.phar %1 %2 %3 %4 %5 %6 %7 %8 %9
```

Ahora tan solo tenemos que ejecutar el archivo por lotes para generar la documentación utilizando la línea de comandos como se nos indica en la web del programa.

```
#genera documentación de la carpeta "src" , en la carpeta "doc"  
  
# Formato largo:  
  
apigen generate --source src --destination doc  
  
# Formato corto:  
  
apigen generate -s src -d doc
```

Donde "src" Carpeta fuente, la que contiene el código de vuestra aplicación, y "doc" la carpeta en la que se generará la documentación.

Otra forma de ejecutar el programa es crear un fichero "apigen.neon" o "apigen.yaml" en el que se guardaran las opciones para generación de la documentación, y que estará en la misma carpeta que ejecutemos el script "apigen"

El fichero `apigen.neon` tendrá una estructura similar a :



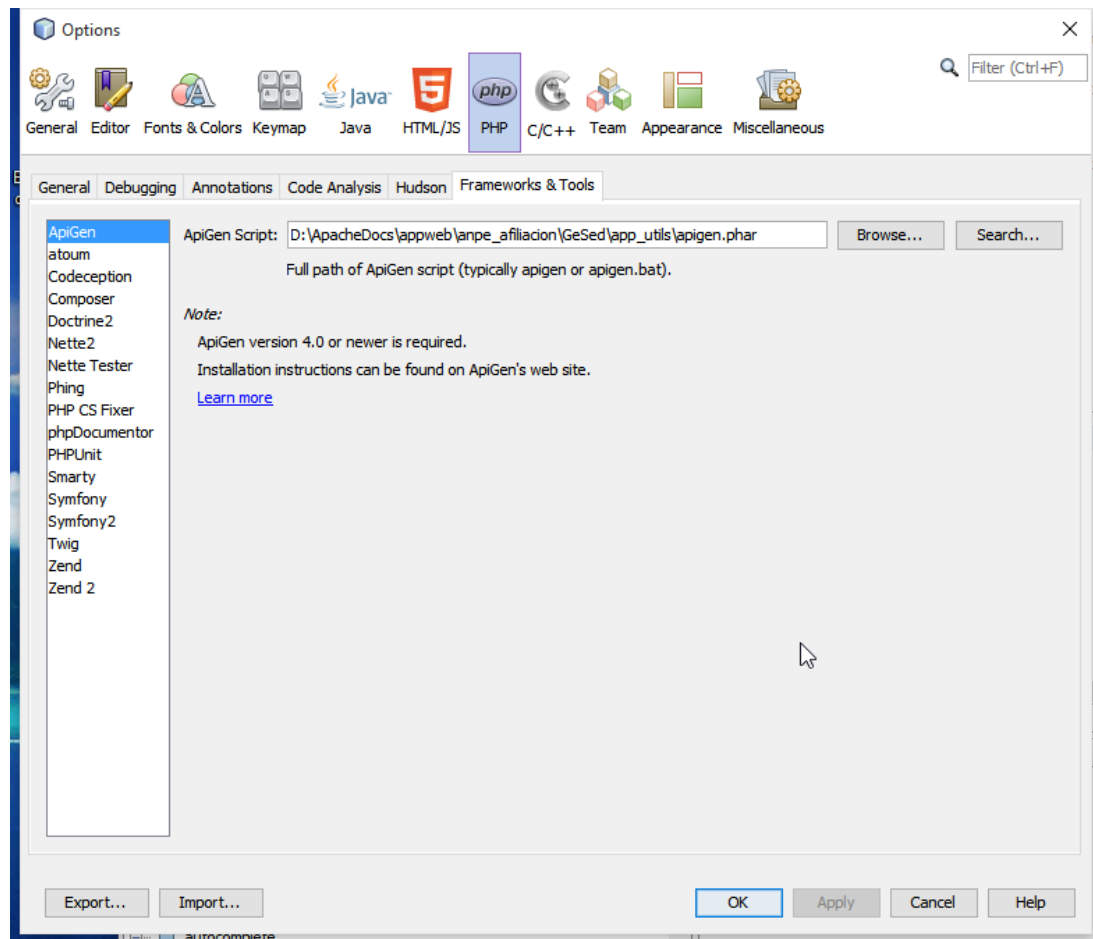
```
title: Título de la aplicación - documentacion
source:
  - src
destination: doc
todo: true
```

Una vez hayamos creado y configurado el fichero "apigen.neon" ejecutaremos el comando

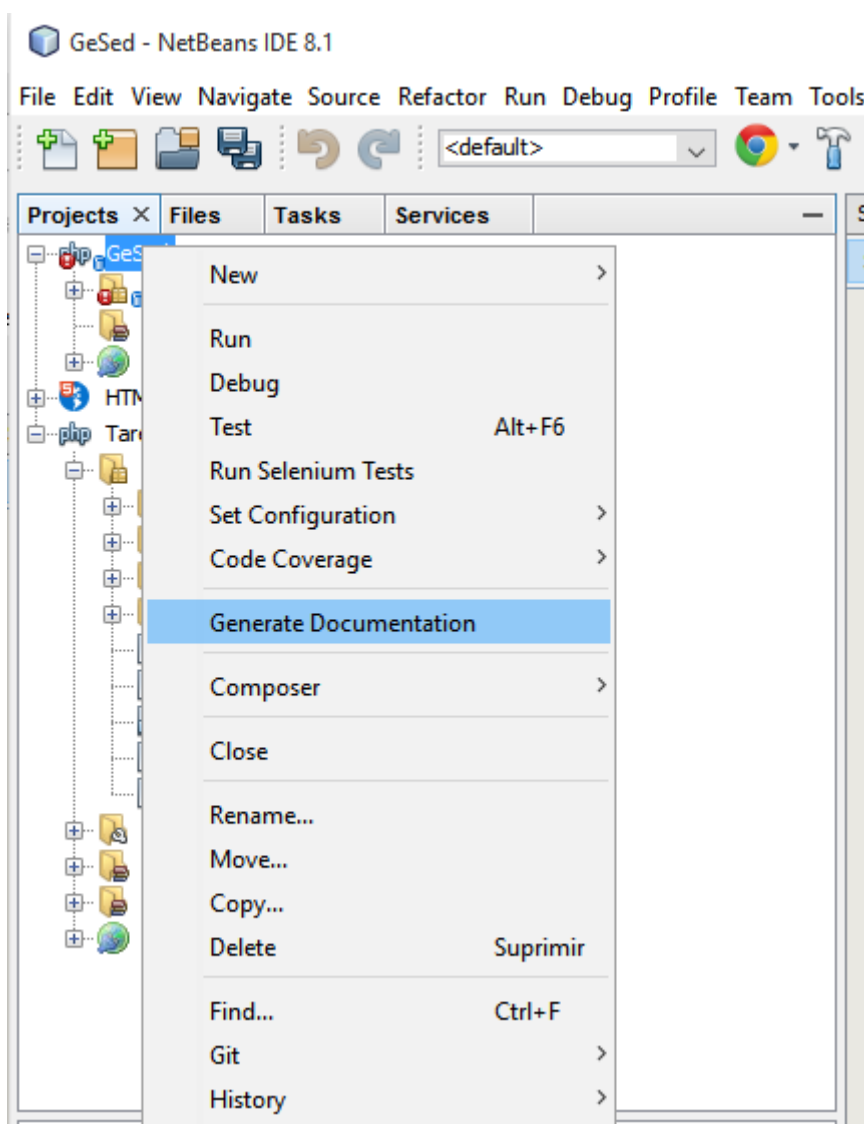
```
C:\Users\username>apigen generate
```

ApiGen se integra de forma sencilla en Netbeans y genera la documentación de un proyecto sin necesidad de utilizar comandos. Tan solo tendremos:

- Configurar NetBeans indicando donde se encuentra la aplicación "apigen.phar". Ver Menú Tools-Options => Ficha "PHP" => Ficha "Frameworks & Tools"



- Seleccionar la opción "Generate Documentation" que se despliega del proyecto



Más información información sobre la utilidad <https://github.com/ApiGen/ApiGen>

Si queréis formatear el texto de ayuda, no dejéis de mirar el [formato Markdown](#)

Anexo II – Control de versiones

Fuente: [Wikipedia - Control de versiones](#)

Una versión, revisión o edición de un producto, es el estado en el se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico).

El control de versiones se realiza principalmente en la industria informática para controlar las distintas versiones del código fuente. Sin embargo, los mismos conceptos son aplicables a otros

ámbitos como documentos, imágenes, sitios web, etcétera.

Más información:

- [Introducción a los sistemas de control de versiones](#)
- [Introducción a los sistemas de control de versiones](#)
- Youtube - [Curso: GIT - GITHUB \[Desde Cero\]](#)

La información anterior está centrada en el trabajo en grupo y en enfoques centralizados, que es la que se suele utilizar en el desarrollo de software. Debido a que nosotros trabajamos individualmente y lo que queremos solamente es mantener una copia de las diferentes versiones de nuestros programas a medida que los vamos desarrollando no precisamos tanta complejidad.

Para uso que nosotros vamos a hacer es suficiente con un sistema de control de versiones distribuido, en nuestro caso el elegido es GIT.