

## Índice de contenido

Modificaciones realizadas en este documento.....	1
Creación 22/10/15.....	1
Modificación 9/11/15.....	2
Normas de obligado cumplimiento.....	2
Comentarios.....	2
Calendario.....	3
1. Gestión de tareas (Ptos 6).....	4
Ver la lista de tareas. (Adm. y ope.).....	5
Añadir un nueva tarea (Adm.).....	6
Modificar datos de una tarea (Adm.).....	6
Eliminar una tarea.....	6
Completar una tarea .....	7
Buscar o filtrar tareas utilizando distintos campos.....	7
Consideraciones finales.....	7
2. Instalador de la aplicación (1 ptos).....	8
3. Gestión de tareas mejorada.....	8
1. Diseño de aplicación modular (1 ptos).....	8
2. Validación de usuario (1 ptos).....	9
3. Validación de multiples usuarios (1,5 ptos).....	9
4. Diferenciando el tipo de usuario (1,5 ptos).....	9
5. Documentación de la aplicación (0,5 pto).....	10
6. Control de versiones (0,5 pto).....	10
7. Configuración de parámetros (0.2 ptos).....	10
Comentarios y evaluación.....	11
Puntuación ejercicios.....	11
Criterios de evaluación.....	11
Instrucciones para entregar la práctica.....	11
Anexo I – Programas de generación automática de documentación.....	12
Apigen.....	12
Instalacion en Windows.....	12
Opciones – apigen.neon.....	14
Modificación de apigen.bat.....	16
Anexo II – Control de versiones.....	17

## Modificaciones realizadas en este documento

### Creación 22/10/15

- Creación del documento. Pendiente de revisión en calendario
- Pendiente revisión de inclusión apartado 3

## Modificación 9/11/15

- Revisado el tercer apartado "3. Gestión de tareas mejorada". Para que incluya versión definitiva. Se han modificado subapartados 2,3 y 4

## Normas de obligado cumplimiento

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

El acceso a la base de datos se realizará utilizando una **capa de abstracción de datos creada por vosotros**, utilizando un enfoque de POO.

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).










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 generarlos automáticamente las [etiquetas](#) (`@ . . .`) pertinentes

## Comentarios

Cada problema de los expuestos a continuación se creará en una carpeta independiente, que llamaréis "Problema 1", "Problema 2", etc.

Cada carpeta que contiene los problemas tendrá la siguiente estructura:

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
	config.php	En este fichero se incluirán parámetros de configuración de la aplicación, como: <ul style="list-style-type: none"><li>- Usuario y clave para acceder a la base de datos</li><li>- Nombre de la base de datos</li></ul> Cualquier otra información que consideréis relevante como <ul style="list-style-type: none"><li>- ruta url de la aplicación</li><li>- ruta absoluta dentro del servidor</li><li>...</li></ul> Todo lo anterior se almacenará en variable 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 <ul style="list-style-type: none"> <li>- La base de datos</li> <li>- La estructura de las tablas</li> <li>- Los usuarios si utilizase alguno diferente</li> </ul>
	Doc	Documentación de la aplicación.
	Assets  css  img  js	Imágenes, ficheros de estilo (css), fichero de script (js), y otro tipo de ficheros.

Los problemas 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 él os oriente.

Cada problema tendrá una fecha de entrega, fecha en la cual deberíais tener completado el ejercicio, esta fecha es orientativa y tiene por objeto evitar que os durmáis en los laureles, tan solo es obligatorio tenerlo todo completo en la fecha de entrega.

Los problemas, **obligatoriamente deberéis enseñárselos 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

13de noviembre	1. Gestión de tareas
20 de noviembre	2. Instalador

1 de diciembre	Fin del bloque 1 y fecha de finalización de práctica
9 de diciembre	Entrega de la práctica
2 o 9 de diciembre	Examen evaluación

## 1. Gestión de tareas (Ptos 6)

La empresa de jardinería "Paco's Garden S.L" se dedica a llevar el mantenimiento de jardines de empresas, comunidades de vecinos, etc. Debido a la mejora económica la empresa ha aumentado enormemente su clientela en los últimos tiempos y su carga de trabajo ha aumentado lo que le ha permitido contratar nuevo personal. Para llevar un control preciso del trabajo a realizar precisa implementar un gestor de tareas que facilite el control y seguimiento de los trabajos que tiene encargados. Para ello nos ha encargado la realización de una aplicación web que permita llevar dicho control y permita a cada operario en cada momento saber las tareas que tiene pendientes y notificar cualquier incidencia o contratiempo que se produzca en la realización de las mismas.

La aplicación tendrá dos tipos de usuarios diferenciados:

- 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 en los que ellos trabajan.

Debido a que estamos comenzando a trabajar, en una primera aproximación realizaremos una interfaz común para 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 tareas. (Adm. y ope.)
- Añadir una nueva tarea. (Adm.)
- Modificar datos de una tarea. (Adm.)
- Eliminar una tarea. Confirmando la operación para evitar errores. (Adm.)
- 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:

- *Descripción*: Texto descriptivo identificativo de la tarea
- *Persona de contacto*: Nombre y apellidos de la persona.
- *Teléfono/s contacto*: Nº de teléfono de contacto de la persona de contacto.
- *Correo electrónico*: Correo electrónico de la persona de contacto.
- *Dirección*: Dirección del jardín, a la que hay que ir a realizar la tarea.
- *Población*: Población en la que está el jardín
- *Código postal*: C.P. del jardín

- *Provincia*: Provincia en la que está el jardín. Para este campo utilizaremos un <select>. Para este campo se almacenará un código numérico.
- *Estado*: Estado en el que se encuentra la tarea (P=Pendiente, R=Realizada, C=Cancelada, ...)
- *Fecha de creación de la tarea*: Fecha en la que se ha creado la envío. Este campo se generará automáticamente. Se deberá usar un disparador de la base de datos.
- *Operario encargado*: Nombre o identificación del operario encargado de la realización de la tarea
- *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.
- *Anotaciones posteriores*: Anotaciones realizadas por los operarios.

La información que contiene estos campos debe cumplir:

- Los campos descripción y persona de contacto debe tener algún valor
- 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.
- La fecha de creación no se podrá modificar.

### **\*\* Obligatorio \*\***

Se deben filtrar todos los datos en el servidor, mostrando el pertinente error en el formulario y guardando el valor enviado para que se pueda editar.

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, aunque esto queda a vuestro criterio de diseño.

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

### ***Ver la lista de tareas. (Adm. y ope.).***

Mostrará la lista de tareas ordenada de forma descendente por fecha de creación. Se deberá paginar la lista de elementos mostrados.

En esta lista se mostrará la información mas relevante de las tareas, o toda si la presentación os lo permite. Si no mostráis toda la información deberéis habilitar una opción que muestre toda la información detallada para dicha tarea.

Sería interesante que desde aquí pudieséis lanzar también las operaciones de borrado y modificación.

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.

### ***Añadir un nueva tarea (Adm.)***

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

A la hora de filtrar los errores se permitirá al usuario ver los valores que ha introducido para permitirle modificarlos.

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.

### ***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 el cual podremos cambiar.

Se deben filtrar los campos antes de proceder a guardar cualquier tipo de dato.

### ***Eliminar una tarea.***

Esta operación permitirá eliminar una tarea de la base de datos. Previo a la eliminación se procederá a confirmar, interactuando con el servidor.

Se mostrará una página de confirmación en la que se muestren los datos más importantes

de la tarea y se pregunte si se desea borrar o no.

### **Completar una tarea .**

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, realianso las anotaciones que se consideren oportunas. 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.

### **Buscar o filtrar tareas utilizando distintos campos.**

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.

### **Consideraciones finales**

Notas:

- 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](#) que os simplificará el trabajo y mejorará sustancialmente la presentación sin mucho esfuerzo.
- 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.

Putunación desglosada por apartados

- 1 pto - Ver la lista de tareas.
- 1 pto – Paginación en lista.
- 1 pto - Añadir un nueva tarea
- 1 pto - Modificar datos de tarea
- 1 pto - Eliminar una tarea. Confirmando la operación para evitar errores.
- 1 pto – Modificar estado de una tarea.
- 1,5 ptos - Buscar o filtrar tareas utilizando distintos campos.

- 2,5 pts – Valoración global de la aplicación teniendo en cuenta: completitud, presentación, organización, documentación, calidad del código, calidad del HTML generado, mejoras realizadas, etc.

## 2. Instalador de la aplicación (1 pts)

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. Gestión de tareas mejorada

Se desea ampliar el ejercicio anterior incluyendo las siguientes funcionalidades:

### 1. Diseño de aplicación modular (1 pts).

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	

Podéis utilizar un esquema como el 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.

## 2. Validación de usuario (1 ptos).

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, el cual almacenaréis en el código.

## 3. Validación de multiples usuarios (1,5 ptos).

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.]

## 4. Diferenciando el tipo de usuario (1,5 ptos).

Ampliaremos el apartado anterior de forma que ahora diferenciaremos el tipo de usuario que se ha validado, restringiendo las operaciones que puede realizar. O sea, los operarios solo podrán realizar las operaciones que tienen asignadas, y tendrán ocultos o deshabilitados los enlaces a las operaciones que están asignadas al administrador.

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.

### **5. Documentación de la aplicación (0,5 pto).**

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.

### **6. Control de versiones (0,5 pto).**

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.

### **7. Configuración de parámetros (0..2 ptos).**

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 al mostrar la agenda.

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.

## Comentarios y evaluación

- La nota de la práctica supondrá el 50% de la calificación sobre el bloque de contenido evaluado, por lo que su entrega será obligatoria.
- 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 ejercicios

---

La puntuación de cada ejercicio es la que se indica en su enunciado.

La práctica puntúa sobre 10, el exceso de puntos que pudierais tener se considerará igualmente para el cálculo de la media.

## Criterios de evaluación

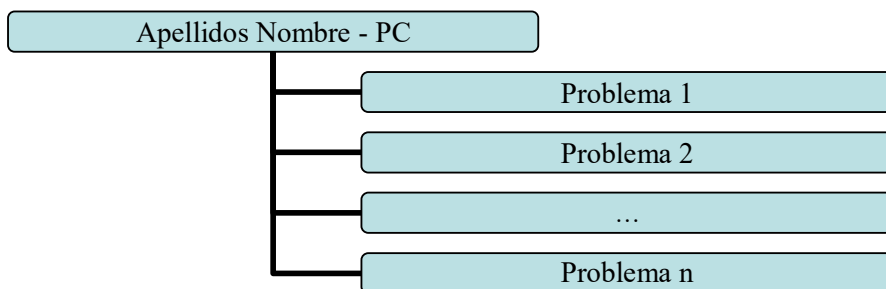
La corrección y puntuación de cada apartado se realizará atendiendo a los siguientes parámetros:

- 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 al servidor web en 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](#). Actualmente ya no se actualiza esta aplicación. Se ha discontinuado su desarrollo.
- Otro interesante programa, que será el que se recomienda utilizar es *ApiGen*. El cual se podrá integrar fácilmente en netbeans como indicamos a continuación.
- **Comparison of documentation generators** [http://en.wikipedia.org/wiki/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>).

### ApiGen

#### Instalacion en Windows

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

Cambia de directorio para situarte en la carpeta que contiene el interprete PHP (php.exe) o incluye en tu PATH este directorio. A continuación ejecuta los siguientes comandos para descargar el programa en php "apigen.phar":

Descargamos el programa y lo compilamos:

```
C:\Users\username>cd C:\bin  
C:\bin>php -r "readfile('http://apigen.org/installer');" | php
```

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":

```
C:\bin>echo @php "%~dp0apigen.phar" %*>apigen.bat
```

Ejecuta el archivo por lotes, deberías obtener información sobre la aplicación:

```
C:\Users\username>apigen  
ApiGen version v4.0.0
```

Una vez ejecutada la aplicación aparecerá el fichero "apigen.neon" en el que describiremos las rutas del proyecto. El fichero `apigen.neon` debería tener el siguiente contenido en la raíz del proyecto:

```
title: My API documentation  
  
source:  
    - src  
  
destination: documentation  
  
todo: true
```

Ejecuta `apigen i` en la carpeta raíz del proyecto y se creará la documentación en la carpeta que se haya indicado para albergar la documentación dentro de la carpeta del proyecto `/documentation`. Se pueden consultar todas las opciones disponibles [aquí](#).

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

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

## Opciones – apigen.neon

```
# list of scanned file extensions (e.g. php5, phpt...)
extensions:
    - php # default

# directories and files matching this file mask will not be parsed
exclude:
    - tests/
    - vendor/
    - *Factory.php

# this files will be included in class tree, but will not create a link to their
documentation
# either files
skipDocPath:
    - * <mask>`` # mask

# or with certain name prefix
skipDocPrefix:
    - Nette

# character set of source files; if you use only one across your files, we
recommend you name it
charset:
    # default
    - auto # will choose from all supported (starting with UTF-8), slow and not
100% reliable
    # e.g.
    - UTF-8
    - Windows-1252

# elements with this name prefix will be considered as the "main project" (the rest
will be considered as libraries)
main: ApiGen
```

```
# title of generated documentation
title: ApiGen API

# base url used for sitemap (useful for public doc)
baseUrl: http://api.apigen.org

# custom search engine id, will be used by search box
googleCseId: 011549293477758430224

# Google Analytics tracking code
googleAnalytics: UA-35236-5

# choose ApiGen own template theme
templateTheme: default # default [other options: bootstrap]

# want to use individual templates, higher priority than option templateTheme
templateConfig: path/to/individual/template-folder/config.neon

# the way elements are grouped in menu
groups: auto # default [other options: namespace, packages, none], auto will detect
namespace first, than packages

# element supported by autocomplete in search input
autocomplete:
  # default
  - classes
  - constants
  - functions
  # other
  - methods
  - properties
  - classconstants
```

```
# access levels of included method and properties
accessLevels:
    # default
    - public
    - protected
    # other
    - private

# include elements marked as @internal/{@internal}
internal: false # default [true]

# generate documentation for PHP internal classes
php: true # default [false]

# generate highlighted source code for elements
sourceCode: true # default [false]

# generate tree view of classes, interfaces, traits and exceptions
tree: true # default [false]

# generate documentation for deprecated elements
deprecated: false # default [false]

# generate list of tasks with @todo annotation
todo: false # default [true]

# add link to ZIP archive of documentation
download: false # default [true]
```

### Modificación de apigen.bat

Para que el script apigen.bat os funcione sin problemas puede que preciséis modificarlo e incluir modificaciones en el PATH



```
set PATH="Ruta en la que está php.exe";%PATH%  
@php "%~dp0apigen.phar" %*
```

Si queréis generar directamente la ayuda

```
set PATH="Ruta en la que está php.exe";%PATH%  
@php "%~dp0apigen.phar" generate
```

## 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](#)
- [Visual SourceSafe. Control de versiones](#)

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.