### Composer PHP

Objetivo del proyecto

* Entender que es un gestor de dependencias.
* Entender que es Composer y cuales son sus fundamentos.
* Mejorar tus conocimientos en PHP en entornos de trabajo profesionales

### Análisis

### El primer punto fuente de este proyecto es elaborar un listado de qué dependencias necesita tu proyecto. Para usarás como dependencia una librería denominada Guzzle.

### Instalamos Guzzle mediante composer y otras librerías necesaria.

### Antes de instalar Guzzle necesitamos instalar Composer

### <https://getcomposer.org/doc/00-intro.md#system-requirements>

### Ejecutamos el script importado

### Que es composer

### No es un administrador de paquetes, se trata de paquetes pero lo administra por proyectos instalando un directorio (vendor), no instala a nivel mundial.

### Es un administrador de dependencias, admite proyecto global a través del comando global

### es una herramienta para el manejo de dependencias en PHP. Permite declarar las librerías de las que tu proyecto depende y se encarga de administrarlas (tanto en su instalación como en su actualización).

### Esta inspirado en el paquete NPM

Suponer:

* Tiene un proyecto que depende de varias bibliotecas.
* Algunas de esas bibliotecas dependen de otras bibliotecas.

Compositor:

* Le permite declarar las bibliotecas de las que depende.
* Averigua qué versiones de qué paquetes pueden y deben instalarse, y los instala (lo que significa que los descarga en su proyecto).
* Puede actualizar todas sus dependencias en un solo comando.

Que forma podemos actualizar los paquetes en Composer

composer update []

Este comando realiza los siguientes pasos:

1. Lee **SIEMPRE** el fichero composer.json.
2. Busca en **Packagist** los paquetes especificados en ese fichero.
3. Resuelve la versión a instalar de cada paquete a partir de las versiones indicadas y la configuración de estabilidad.
4. Resuelve todas las dependencias para esas versiones.
5. Para aquellos paquetes que exista una nueva versión disponible, la descarga y la instala sustituyendo a la versión actual.
6. Una vez instalados los paquetes, si no existe composer.lock, lo crea para dejar ‘una foto fija’ del entorno de ejecución de la aplicación. En caso de que exista, lo actualiza. También crea los ficheros de autocarga de clases de la aplicación.

# Puedes actualizar todos los paquetes que se encuentran en composer.json

composer update

# Puedes actualizar únicamente uno o varios paquetes separándolos por espacios

composer update doctrine/dbal laravel/framework

# Puedes actualizar todos los paquetes de un proveedor (‘vendor’) usando un asterisco

composer update doctrine/\*

Podemos eliminar dependencia

El comando remove sirve para eliminar alguna dependencia que ya no utilicemos, de la siguiente manera:

$ php composer remove vendor/package

Para que sirve el fichero composer.json

Es un archivo json que nos sirve para gestionar las dependencias instaladas a nuestro proyecto.

Requisitos del proyecto

* Debes de realizar todos los pasos haciendo uso únicamente de la línea de comandos.
* Debes de configurar tu repositorio para que ignore los siguientes ficheros y directorios
  + Directorio donde se instalen las dependencias de composer
* Debes de poder ejecutar la librería Guzzle y hacer un pequeño ejemplo mediante la metodología que proporciona composer.
* Debes de tener claro el uso de composer y de su herramienta por línea de comandos.
* Debes de tener claro qué diferencia existe entre una dependencia de desarrollo y una dependencia de producción
* Crear una estructura de directorios clara y ordenada
* Tanto el código como los comentarios tienen que estar escritos en inglés
* Usa el estilo de código camelCase para la definición de variables y funciones
* En el caso de estar usando HTML nunca uses estilos en línea
* En el caso de estar usando diferentes lenguajes de programación siempre define la implementación en términos separados
* Recuerda que es importante dividir las tareas en varias sub-tareas para que de esta forma puedas asociar cada paso en particular de la construcción con un commit específico
* Debes intentar en la medida de lo posible que los commits y las tareas planificadas sean lo mismo
* Borra los ficheros que no se usen o no sean necesarios para evaluar el proyecto

Implementación del proyecto



|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Prioridad** | **Horas** | **Dificultad** |
| Documentación | Normal | 1,00 | Alta |
| Organización | Alta | 1,00 | Alta |
| Búsqueda Previa de información | Normal | 1,00 | Normal |
| Creación de repositorio | Baja | 0,15 | Baja |
| Documentando Proyecto | Normal | 1,00 | Normal |
| Previa investigación  Composer | Normal | 1,00 | Normal |
| Installar Composer | Alta | 1,00 | Alta |
| Estructura Index.php principal | Normal | 1,00 | Normal |
| Importacion Librería necesarias | Normal | 1:00 | Normal |
| Vistas primarias  base.html.twig | Normal | 1,00 | Normal |
| Bloques Template home | Alta | 2.00 | Alta |
| Bloques formulario de contacto | Alta | 1.00 | Normal |
| Bloques iterar las noticias | Alta | 1.00 | Alta |
| Comprobar que existen noticias | Alta | 1.00 | Normal |
| Creación README | Baja | 1,00 | Baja |
| Testing / Corrección Errores | Alta | 1,00 | Alta |

Tenemos previsto terminar el proyecto en 12 Horas

En las cual damos 8 horas más para prever incidencias en el proyecto

## Especificaciones del proyecto

1. Debes de usar GIT desde el inicio del proyecto
2. Debes desarrollar el proyecto con **PHP**y **WampServer**
3. **Utilizamos Composer para instalar Twig**
4. Deberás usar en nuestro Script **Twig**para importar los componentes necesario
5. Componentes
6. Todo el código debe de estar debidamente documentado
7. Todos los comentarios incluidos en el código tienen que estar escritos en inglés
8. Usa el estilo de código camelCase
9. Es recomendado dividir las tareas en varias subtareas para que de esta forma puedas asociar cada paso en particular de la construcción con un commit específico
10. Para la documentación del proyecto es requerida una versión en PDF dentro del repositorio
11. Debes intentar en la medida de lo posible que los commits y las tareas planificadas sean lo mismo
12. Borra los ficheros que no se use

Registro De Incidencia que se ha detectado durante el proyecto

* Ninguno

Seguimiento Calendario del proyecto

Métricas de Calidad

A pesar de que el proyecto debe adherirse a todos los [*requisitos*](https://docs.google.com/document/d/1dtnHybW7qhBAuuuNEZi1Pb0neJEqbME-hXvGSw2XHUI/edit#heading=h.rf0k2roeckt1) y [*especificaciones*](https://docs.google.com/document/d/1dtnHybW7qhBAuuuNEZi1Pb0neJEqbME-hXvGSw2XHUI/edit#heading=h.xr3uz2kyop9d)del proyecto, hay algunas condiciones que, si se cumplen adecuadamente, añaden una sensación de calidad y robustez al propio proyecto. Estas condiciones son:

1. Wampserver utilizamos index.php
2. Instalacion de Composer
3. El código twig utilizamos Visual Studio Code
4. El código twig debe estar libre de pelusas.
5. La aplicación web debe responder.
6. La aplicación web debe ser compatible con los principales navegadores del mercado:
   * Internet Explorer 11 o superior.
   * Safari en una de sus últimas versiones.
   * Firefox en una de sus últimas versiones.
   * Chrome en una de sus últimas versiones.

Documentación de Riesgos

* Pérdida o daños de material de trabajo.

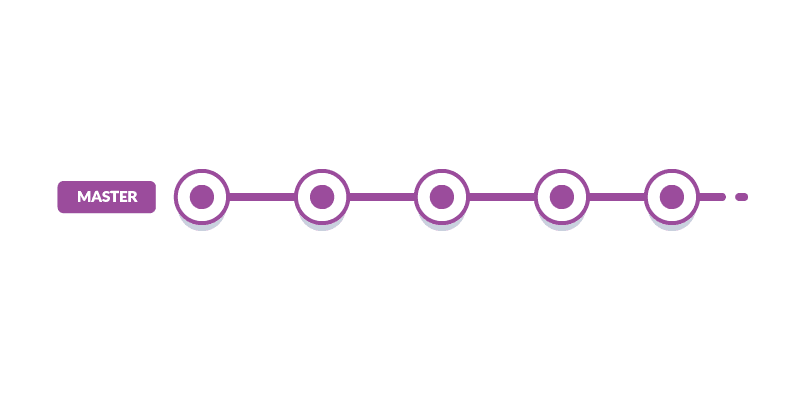
Documentación WORKFLOW DE GIT

* Creación Git Hub https://github.com/robertfox11/twigPills.git
* Hacemos commits de la estructura de la página principal.
* Probabilidad de que ocurra 80%
* Impacto en el proyecto 60%
* Posible alternativa (mitigación) Pedir ayuda a compañeros
* Probabilidad de que ocurra 30%
* Impacto en el proyecto 60%
* Posible alternativa (mitigación) Pedir ayuda a compañeros
* No encontrar con facilidad la información relacionada con el proyecto
* Probabilidad de que ocurra 30%
* Impacto en el proyecto 60%
* Posible alternativa (mitigación)
* Pedir ayuda a compañeros

A partir de la realización de la estructura se continuó trabajando solamente en la

rama “master”, a través del Workflow “Gitflow”.

Mas información --> <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow>



Herramiento del proyecto

Se utilizaron diferentes herramientas en el desarrollo del proyecto. Son los siguientes:

1. ***git: un potente sistema de control de*** versiones que ayuda a realizar un seguimiento de los cambios en el árbol de trabajo.
2. ***Visual Studio Code: un editor de*** código optimizado para crear y depurar aplicaciones web modernas.
3. ***Wamps Server, para realizar la el proyecto***
4. ***Composer para hacer las peticiones de las librería***
5. ***Instalacion de TWIG y dependencias necesaria***
6. ***Herramientas para desarrolladores de Google Chrome:*** se utiliza para depurar el código JavaScript y para probar los ajustes de diseño.
7. ***Documentos de Google:*** se utiliza para escribir la documentación del proyecto.
8. [***Validador W3C***](https://validator.w3.org/)***:*** utilizado para validar el código HTML y CSS.

## Flujo de trabajo de Git

Todas las confirmaciones se van a insertar en la rama ***maestra,*** siguiendo un criterio personal de cargar solo instantáneas que son funcionales y que funcionan correctamente, sin contar errores menores. No hay otras ramas, ya que ralentizaría el proceso de desarrollo.

En otro lado, los mensajes de confirmación finalizan con su objetivo principal indicado entre corchetes: por ejemplo.

* Utilizaremos un template base.html.twig que las instalaremos con composer
* Plugin Twing

## Estructura del archivo

Los archivos del proyecto se organizarán de esta manera:



Registro de lecciones aprendidas.

* Instalaccion de Composer
* Renderizacion con twig
* Plantillas que se utilizan con twig