SUMÁRIO

1. COMPOSER E AUTOLOAD PSR-4

O Composer é o gerenciador de dependências do PHP, com ele é possível definir uma lista de bibliotecas que sua aplicação necessita para funcionar, além de poder definir requisitos como, versão do PHP, extensions etc. Com algumas poucas linhas de configurações você define todas as bibliotecas de terceiros ou mesmo suas que deseja/precisa utilizar em seu projeto, o composer encarrega-se de baixá-las e criar um autoloader deixando-as prontas para uso.

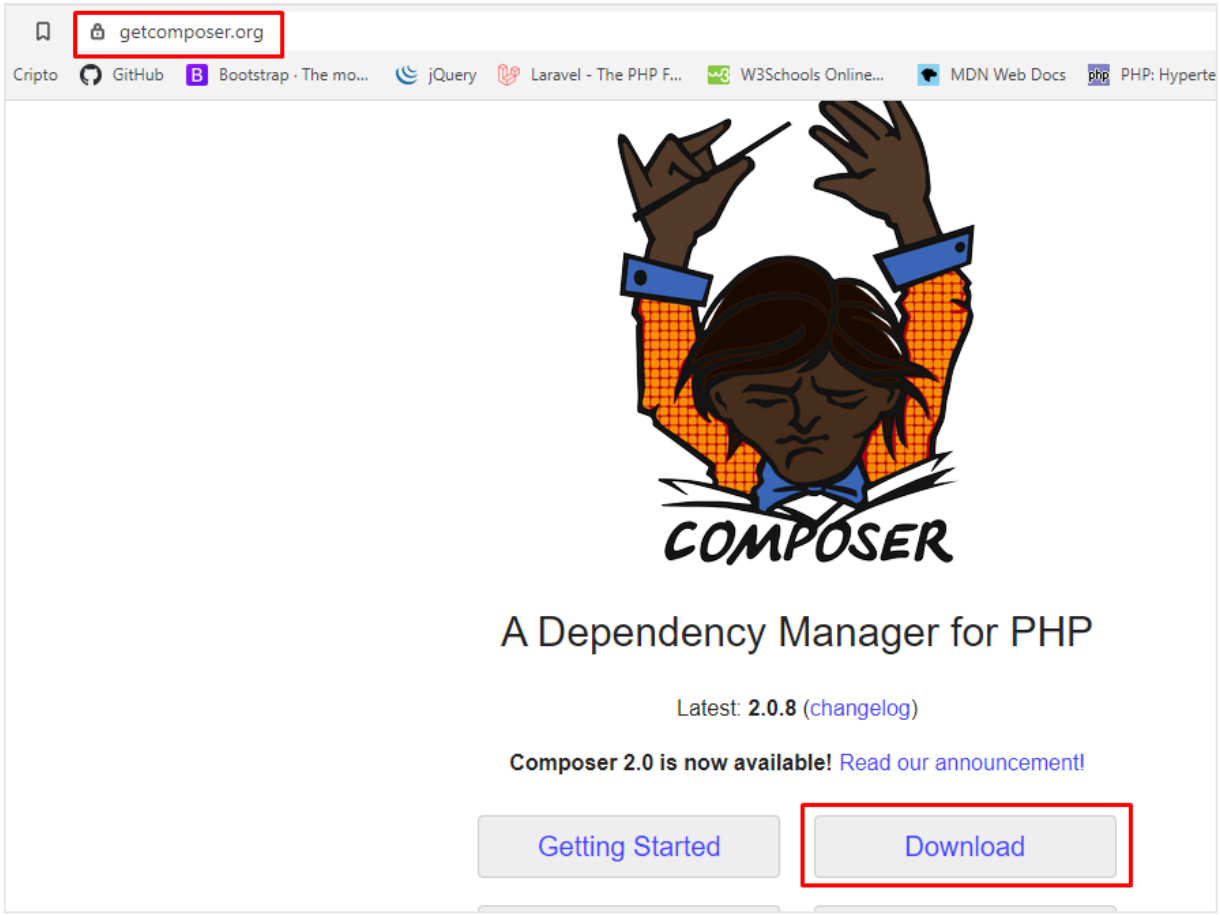
Vamos entender melhor considerando o escopo de uma aplicação MVC. Como bem sabemos, existem camadas da aplicação, possuindo cada uma seus recursos, que devem, por serem segmentos do programa, conversar entre si em situações determinadas. Normalmente, para fazer isto, utiliza-se recursos como o require, para transportar funcionalidades entre as estruturas das aplicações, certo? **Pois o Composer serve para este fim**, tendo consigo um sistema de carregamento automático, que opera segundo as especificações passadas, cujo nome é PSR-4.

Na prática, o Composer irá mapear toda a aplicação, todas suas segmentações, classes, scripts, e **irá criar namespaces** para o acesso aos recursos mapeados e que são necessários em diferentes escopos do projeto. **Isto dispensa o uso recorrente de require ou include** para a utilização de recursos externos.

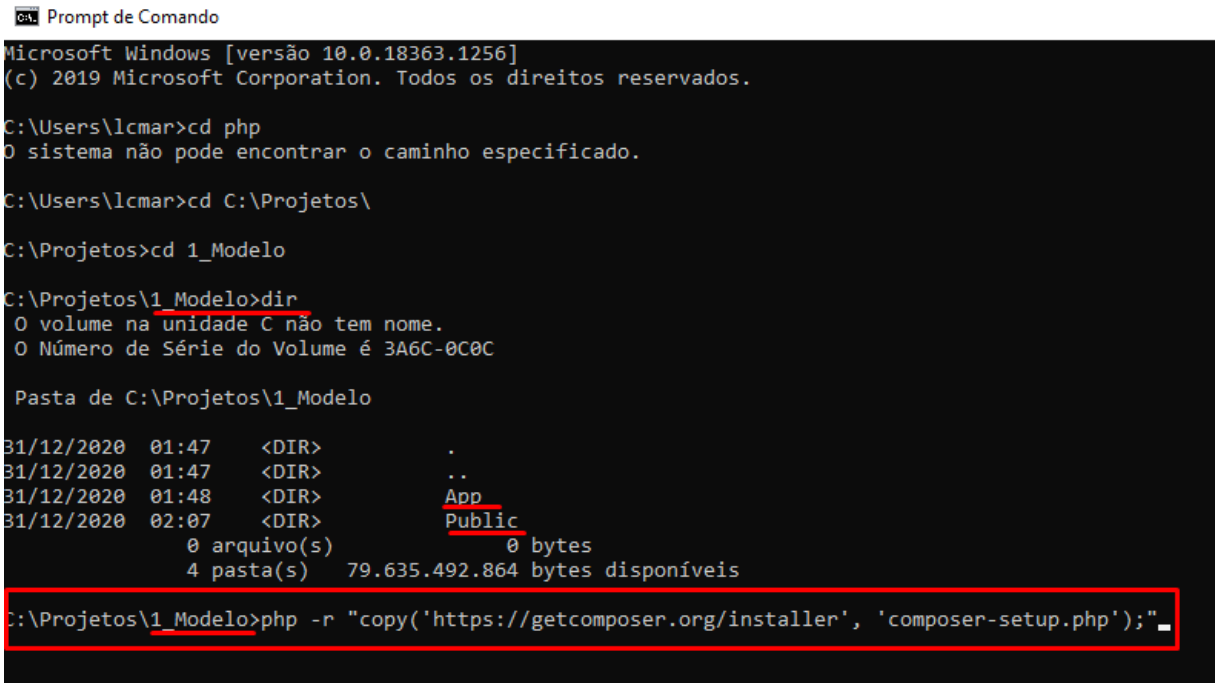
1.1. INSTALANDO O COMPOSER

Existem duas alternativas: baixar para o Windows, como um executável, ou para um projeto especifico, apenas, por meio de linha de comando. A diferença é que, baixando como executável, as funcionalidades do composer, derivadas do composer.phar file, serão acessadas globalmente, e instalando localmente, será necessário o uso do comando php.

Neste caso, será usada a segunda opção. Acesse: https://getcomposer.org/download/



O processo de download do Composer, para projetos em especifico, por linha de comando, deve ser realizado dentro do diretório raiz do projeto, isto é, a pasta que contém o diretório “App” e “Public”. No exemplo abaixo, o diretório raiz chama-se “1_Modelo”:

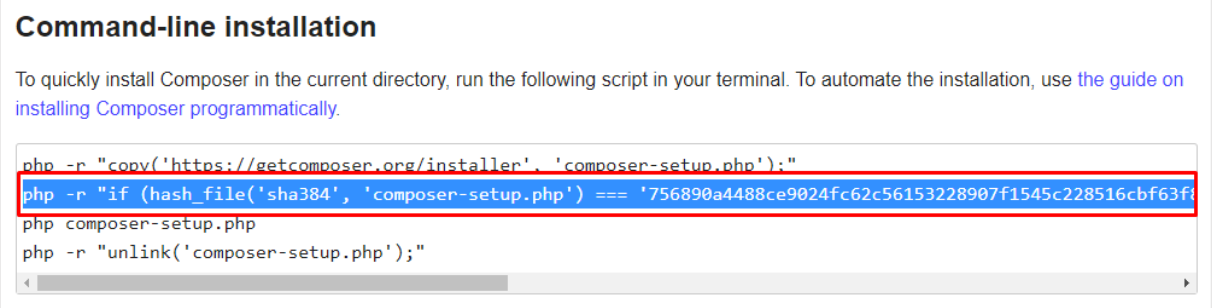


O comando que deve ser executado no diretório raiz é este:

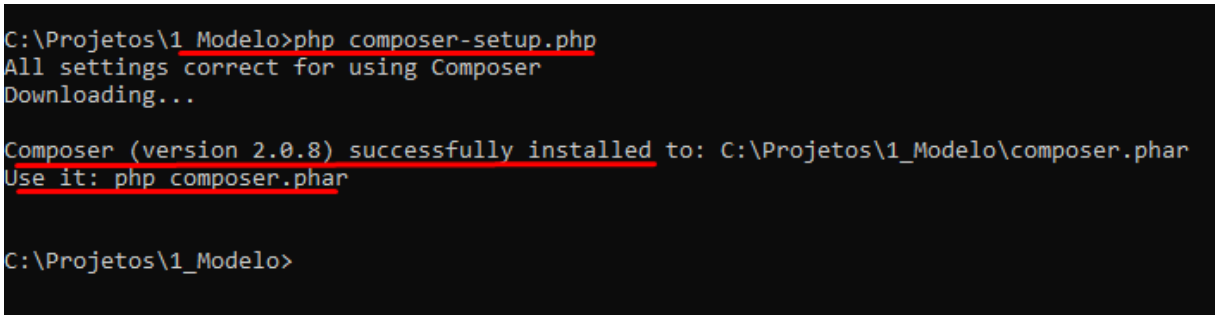
Um erro que pode ocorrer, na tentativa de instalar desta forma, é o "Unable to find the wrapper https". Para resolver isto basta que, no arquivo php.ini, a extensão openssl seja ativada (retirando o ";" a sua esquerda).

Se bem sucedido este comando, no diretório raiz do projeto irá aparecer um arquivo com extensão PHP, chamado "composer-setup".

Em seguida, podemos conferir se o arquivo está de acordo, e se não está, de alguma forma, modificado ou corrompido. Para isso, deve-se executar, inteiramente, o segundo comando, existente no guia de instalação do composer.



O terceiro passo consiste em copiar o terceiro comando existente no guia de instalação, para que efetivamente o script de instalação do composer seja executado.



Após o terceiro passo, existirá, no diretório raiz do projeto, um arquivo com extensão "phar" chamado "composed". Assim, o arquivo de extensão PHP "composer-setup" pode ser removido, ou por um "delete" manual, ou por meio da execução do quarto comando existente no guia de instalação.

Em seguida, para instalar o composer dentro do projeto, deve-se criar, no diretório raiz, um novo arquivo chamado "composer.json", que conterá algumas instruções padrões, mas cujos valores, claramente, variam de projeto para projeto.

1.1.1. ARQUIVO COMPOSER.JSON E FIM DO PROCESSO DE INSTALAÇÃO

Nestas configurações é definido um nome para o pacote do projeto, que deve ser escrito no formato vendor/nome_projeto, a versão mínima do PHP para rodar a aplicação, dados de autoria e para contato, e, mais importante, as configurações para o "autoload". Neste último campo, definimos sua versão, que é PSR-4, e os namespaces, que são diretórios mapeados.

O primeiro diretório mapeado é o "App", cujo namespace foi definido como "App\\", e foi informado que sua localização é "App/". Se estivesse em outro nível de diretório, em relação ao composer.json, seria outra localização, uma acima, ou abaixo, naturalmente.

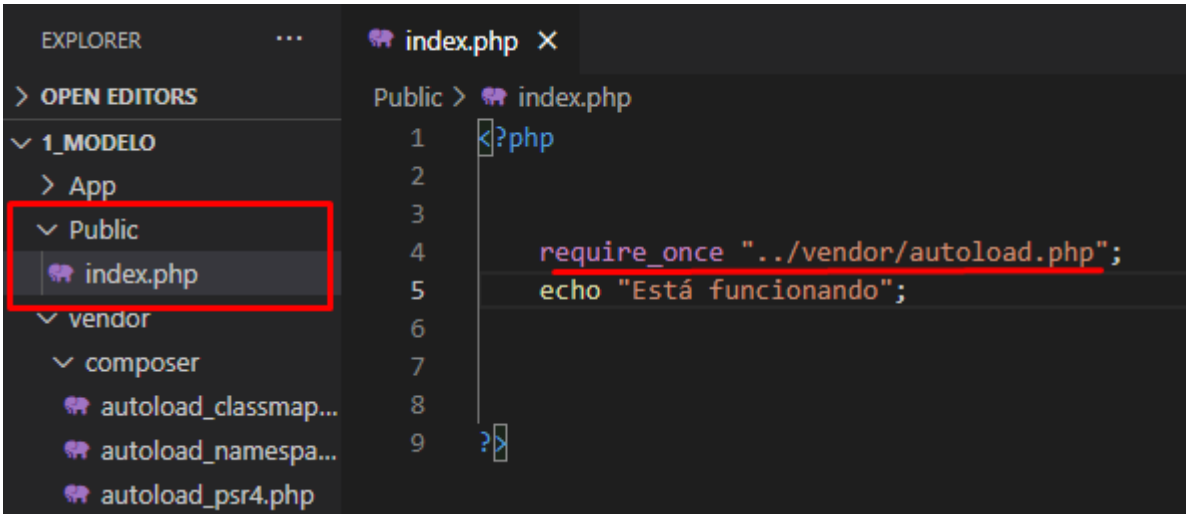
O segundo diretório mapeado é o "Mod", cujo namespace foi definido como "Mod\\", e foi informado que sua localização é "vendor/Mod/". Mas que diretório é este? Sim, não existe, ainda. **Primeiro, o diretório "vendor"** é criado, na raiz, logo após a instalação completa do composer, e sua função é, geralmente, armazenar as dependências do projeto que sejam de terceiros. **Enquanto que, o diretório "Mod"**, que será criado manualmente, irá conter as dependências para abstração de recursos das camadas da aplicação.

Ainda, sobre o diretório vendor, é necessário enfatizar que seu uso é geralmente para armazenar dependências criadas por outros, ou seja, não é uma regra absoluta, e sim um costume de desenvolvimento. Diversos projetos utilizam o diretório vendor para este fim.

Por conseguinte, com os passos anteriores realizados, deve-se **executar o comando**, na linha de comando, ainda no local do diretório raiz do projeto, o comando "**php composer.phar install**".

Com isto, novos diretórios e arquivos serão criados. Dentro todos, que são muitos, estão inclusos a pasta vendor e seus arquivos default, cujos valores correspondem, em parte, as definições do composer.json – algumas configurações podem ser consultadas nestes arquivos. **Além disso, deverá ser criada a pasta Mod**, dentro de vendor, que terá as dependências que irão abstrair elementos das camadas MVC, como citado.

Por fim, na pasta "public" deverá ser criado o arquivo "index.php", e escrito em seu código uma requisição única para o arquivo de autoload do Composer.


EXPLORER
OPEN EDITORS
1_MODELO
App
Public
index.php
vendor
composer
autoload_classmap...
autoload_namespa...
autoload_psr4.php
index.php
Public > index.php
1 <?php
2
3
4 require_once "../vendor/autoload.php";
5 echo "Está funcionando";
6
7
8
9 ?>