Composer

TOLEDO, Lucas[1]

SILVA, Rubens[2]

RESUMO

O seguinte trabalho apresenta os elementos que constituem a estrutura da ferramenta *Compose*r, bem como apresenta de forma geral as regras de uso e particularidades do mesmo. As orientações aqui apresentadas baseiam-se na leitura de artigos e trabalhos previamente estudados.

Palavras-chave: *composer, php, gerenciador, dependências, libraries*

1 INTRODUÇÃO

Em um processo de desenvolvimento de software é comum durante a maior parte do tempo a equipe de implementação usar dependências na aplicação, mesmo que criadas pelos próprios membros da equipe, o que até este ponto não é necessário agregar preocupação, porém, a partir do momento que se depende de códigos que não foram criados pelo próprio desenvolvedor, ou seja pertencem a terceiros, é que surge a necessidade de se ter um controle mais rigoroso, eficaz e simplificado.

Em muitos casos o código implementado nas dependências da aplicação é quase que incompreensível e nesta tratativa é que discorre sobre o uso do *Composer* como ferramenta a fim de auxiliar o desenvolvimento das aplicações em voltadas a linguagem de programação *Personal Home Page,* conhecida popularmente como PHP.

2 DESENVOLVIMENTO

Primeiramente far-se-á necessário entender e responder ao discorrer do texto o entendimento da ferramenta e necessário se faz a resposta de questões sobre como e onde surgiu, junto ao porquê do uso dessa ferramenta para gerência de configuração?

[1] Estudante – Graduando em eng. De Computação – Centro universitário de Anápolis - UNIEvangélica. E-mail:lucastoledooficial@gmail.com

[2] Estudante – Graduando em eng. De Computação – Centro universitário de Anápolis - UNIEvangélica. E-mail: Rubens.silva08@hotmail.com

2.1 A Criação

Composer é uma ferramenta para o gerenciamento de dependências em PHP. Desenvolvido por Nils Adermann e Jordi Boggiano. Eles começaram o desenvolvimento em abril de 2011 e lançaram pela primeira vez em 1º de março de 2012.

Ela permite a declaração de bibliotecas que o projeto necessita, e a gerência de tais bibliotecas para o programador. Para melhor entendimento pode-se tomar como exemplo de comparação, para quem já está familiarizado com gerentes de dependências *PIP* para *Python,* ou *NPM* para *Node.js*, o *Bundler* e o *Mavenno*, *Ruby*. O gerente de dependências *Composer* foi fortemente inspirado pelos *NPM* do *Node.js* e o *Bundler* do *Ruby*.

2.1.1 Metodologia

O *Composer* é executado a partir de linhas de comandos, onde após executadas é realizada as instalações das *libraries* (bibliotecas) para um aplicativo. Ele também permite os usuários instalarem aplicativos PHP que estão disponíveis no "*Packagist*" que é seu repositório principal contendo pacotes disponíveis. O gerente de dependências fornece recursos de carregamento automático para *libraries* que especificam informações de carregamento automático para facilitar o uso de códigos de terceiros.

2.1.2 Comandos

O Composer oferece vários parâmetros, dentre eles podemos referenciar o *require* responsável por adicionar a *librarie* no parâmetro ao arquivo *composer.json*; o comando *install* após executado realiza a instalação de todas as *libraries* do *composer.json* que é o comando usado para baixar todas as dependências do repositório do *PHP*. O *update* após executado atualiza todas as bibliotecas do *composer.json*, de acordo com as versões permitidas mencionadas nele, e por último encontramos o *remove* incumbido de desinstalar uma biblioteca e removê-la do *composer.json*.

2.2 Gerenciamento de Dependências

O *Composer* não é um gerenciador de pacotes no mesmo sentido que o *YUM* ou o *APT* são. Sim, ele trabalha com *package* ou *libraries*, mas os gerencia de outra forma em uma base por projeto, instalando-os em um diretório dentro de seu projeto. Globalmente por padrão ele não instala nada. Dessa forma, ele se torna um gerenciador de dependências. Porém, ele suporta um projeto "global" por conveniência através do comando global. Essa ideia não é nova e o *Composer* foi fortemente inspirado pelo *NPM* e pelo *Bundler* do *Node.js*.

Suponha-se que o programador tenha um projeto que depende de várias bibliotecas, e que algumas dessas bibliotecas dependem de outras bibliotecas. O *Composer* permite a declaração das bibliotecas das quais o usuário da ferramenta

necessita, além de descobrir quais versões dos pacotes precisam ser instaladas e as realiza o que significa que elas serão transferidas para o seu projeto.

2.3 Instalação

Uma característica interessante do *Composer* e que o separa de outras soluções de gerenciamento de dependência como o *PEAR*, é a capacidade de resolver dependências em uma base por projeto. O *Composer* gerencia separadamente dependências para cada um de seus projetos. Isso significa que não é preciso ter as bibliotecas em seus projetos que não estão sendo trabalhados - um bônus em termos de manter o tamanho do projeto em cheque. Dito isto, pode-se instalar pacotes em todo o sistema usando o *Composer*.

A instalação no sistema operacional *Windows* é realizada de forma prática onde nenhuma instrução de linha de comando é necessária para fazer o download e instalar o *Composer*, é disponibilizado um *setup* de fácil instalação no *site* oficial da ferramenta. Após concluir a instalação, abra o *prompt de comando* (para abri-lo, pressione a tecla *Windows*, digite *cmd* e pressione a tecla *Enter*). Já no *prompt de comando* digite: *composer. Será gerada uma imagem dentro da janela do prompt de comando com a seguinte mensagem: "*Parabéns! Você já instalou o *Composer* em seu computador com Windows". O instalador automaticamente se encarregará de adicionar *Composer* a sua variável *PATH*. Você pode abrir o *prompt de comando* e executar o *Composer* a partir de qualquer lugar.

REFERÊNCIAS BIBLIOGRÁFICAS

Documentação Oficial do *Composer*. Disponível em <https://getcomposer.org/doc/>. Acesso em 11 de junho de 2019.

Pataki, Daniel. A Beginner's Guide To *Composer*. Disponível em <https://scotch.io/tutorials/a-beginners-guide-to-composer>. Acesso em 10 de junho de 2019.

Van De Vreken, Hannes. Tilde and caret version constraints in *Composer*. Disponível em <https://blog.madewithlove.be/post/tilde-and-caret-constraints/>. Acesso em 11 de junho de 2019.