PHP Composer: Dependências, Autoload e Publicação

- Esse curso capacita a:
  - Saber como gerenciar dependências;
  - Entender o Autoload de classes e funções;
  - Integrar ferramentas como PHPUnit;
  - Automatizar tarefas rotineiras com scripts;
  - Publicar e versionar o seu pacote;

Aulas:

1. **Instalando o Composer:**
   - Composer é um gerenciador de dependências PHP, que instala as dependências, para cada projeto, de forma especificada;
   - Para usar dependência global -> **global command**;
   - Um pacote sempre segue a nomenclatura: <vendor>/<name>;
   - **composer init** inicializa o composer.json;

2. **Gerenciando dependências:**
   - Composer possui um repositório central de pacotes: packagist.org;
   - guzzlehttp/guzzle, pacote que serve para executar requisições HTTP de alto nível;
   - Para instalar alguma dependência: **composer require <nome do pacote>**;
   - Composer já gera um arquivo de Autoload.php;
     - O comando **composer install** automaticamente baixa todas as dependências no composer.json;
     - O arquivo composer.lock define todas as versões exatas instaladas;

3. **Entendendo Autoload:**
   - A PSR-4 define um padrão para o carregamento automático de classes;
   - O vendor namespace (namespace padrão) fica mapeado para uma pasta do projeto dentro do arquivo composer.json;
   - Para atualizar o Autoload: **composer dumpautoload**;
     - Para classes que não seguem o PSR-4, podemos definir um classmap dentro do composer.json;
     - Para carregar uma biblioteca de funções automaticamente, podemos adicionar uma entrada files no composer.json;

4. **Ferramentas de qualidade de código:**
   - Através da flag "—dev" definimos que uma dependência não faz parte do ambiente de produção;
   - Para baixar apenas as dependências de produção: **composer install no-dev**;
   - Componentes encontram-se na pasta vendor/bin;
   - **phpunit** para rodar testes // **phpcs** para verificar padrões de código // **phan** para executar uma análise estática da sintaxe do nosso código;

5. **Automatizando processos com Scripts:**
    - Scripts são definidos dentro do .json;
    - Scripts podem definir comandos que chamam ferramentas instaladas pelo Composer, comandos do sistema operacional, códigos PHP e podem ser associados a eventos, como **composer install** ou **update**;

6. **Publicando um pacote:**
    - Composer entende as tags de versão de um repositório Git, e segue o conceito do versionamento semântico (MAJOR.MINOR.PATCH);
    - No composer.json é possível definir constraints;
        - Para distribuir e disponibilizar o projeto deve-se:
            - Criar um repositório no Github;
            - Usar o packgist e associar com o repositório no Github;