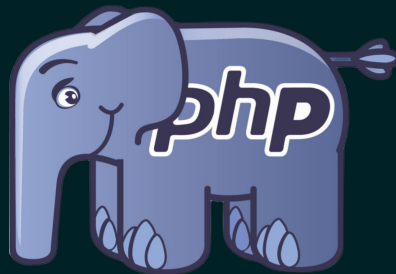


Semestre 3

Back-End

Semaine 14 - PHP



Par Pierre-Alexandre Lacaze - pa.lacaze1@gmail.com



01

Le Back-End

Qu'est ce que c'est ?

Ce que l'on connaît déjà



HTML

Contenu

JAVASCRIPT

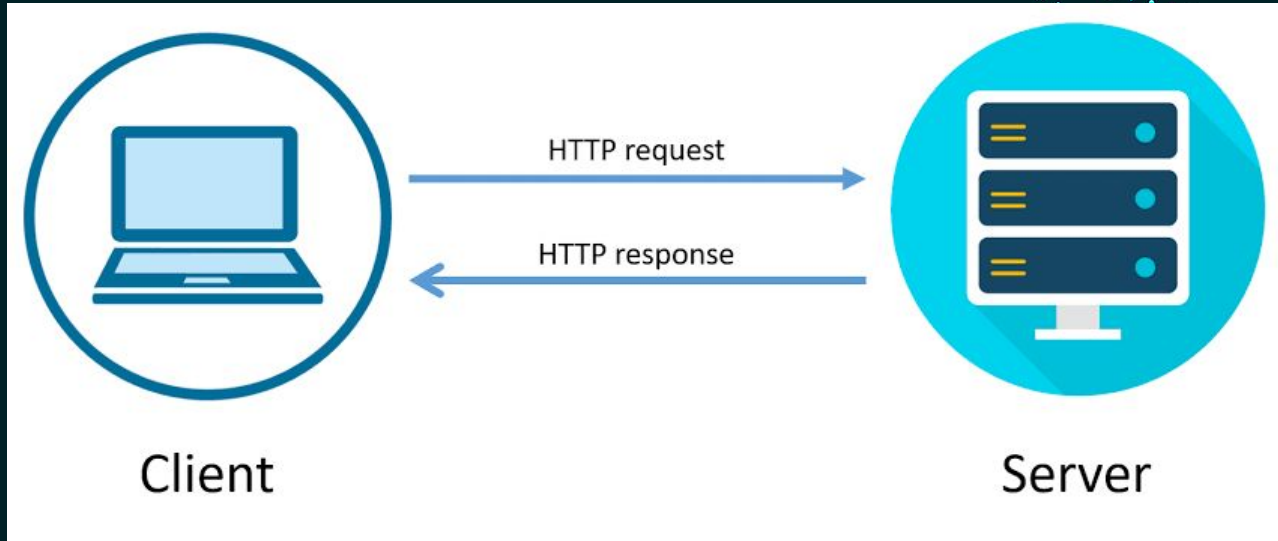
Interactions

CSS

Design

Structure sémantique

Client/Serveur Front-End



- Une page demandée -> Une page renvoyée

Le Back-End c'est quoi ?



Stockage

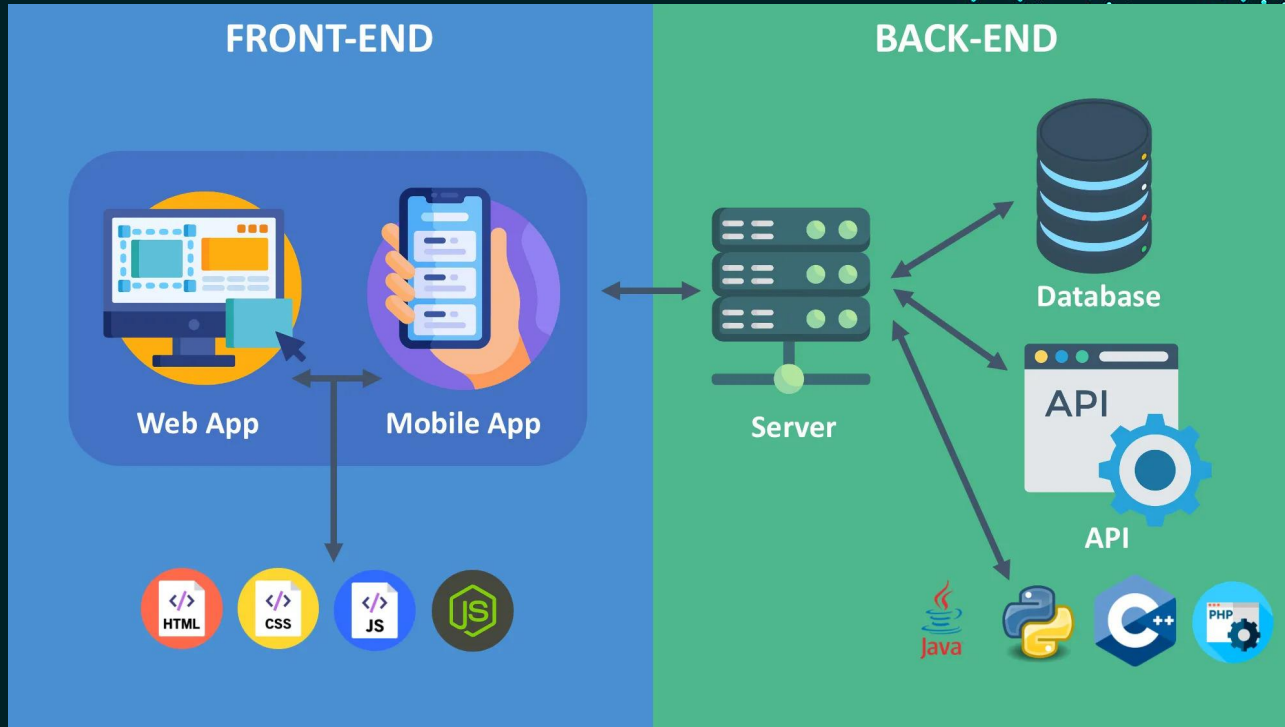


Logique



Sécurité

Client/Serveur Back-End



Quelques langages Back-End

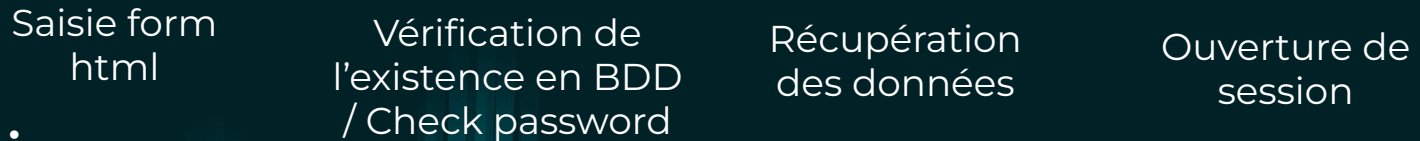


Exemple de processus

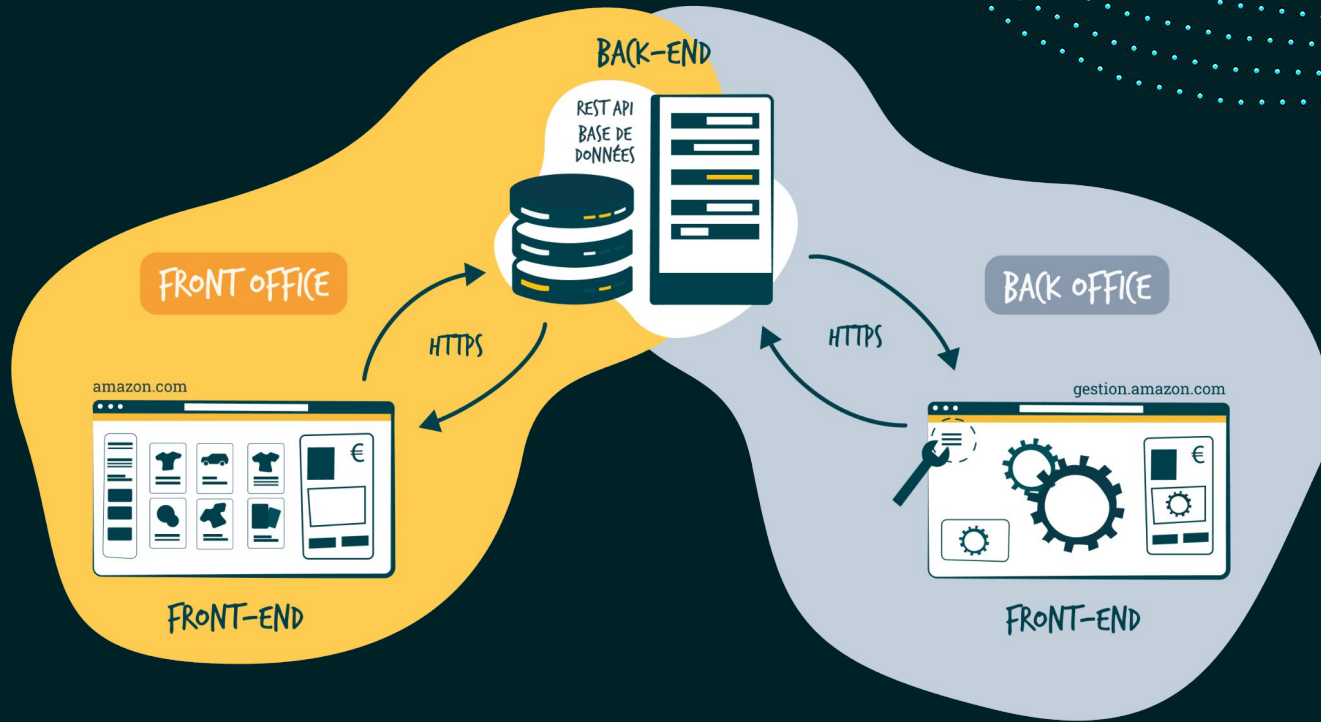
Inscription :



Connexion :



Une nouvelle ère débute





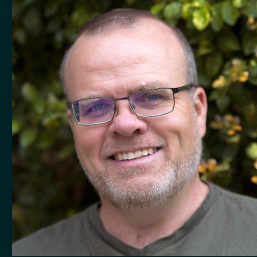
02

Le PHP

La star du Back-End

Historique

- Créé en 1994 par Rasmus Lerdorf
- Nommé Personal Home Page Tools



- Repris et affiné en 1997 par Andi Gutmans et Zeev Suraski



Pourquoi PHP ?

01

LIBRE DE DROIT

Usage libre, gratuit,
open source,
évolution constante

02

RAPIDE

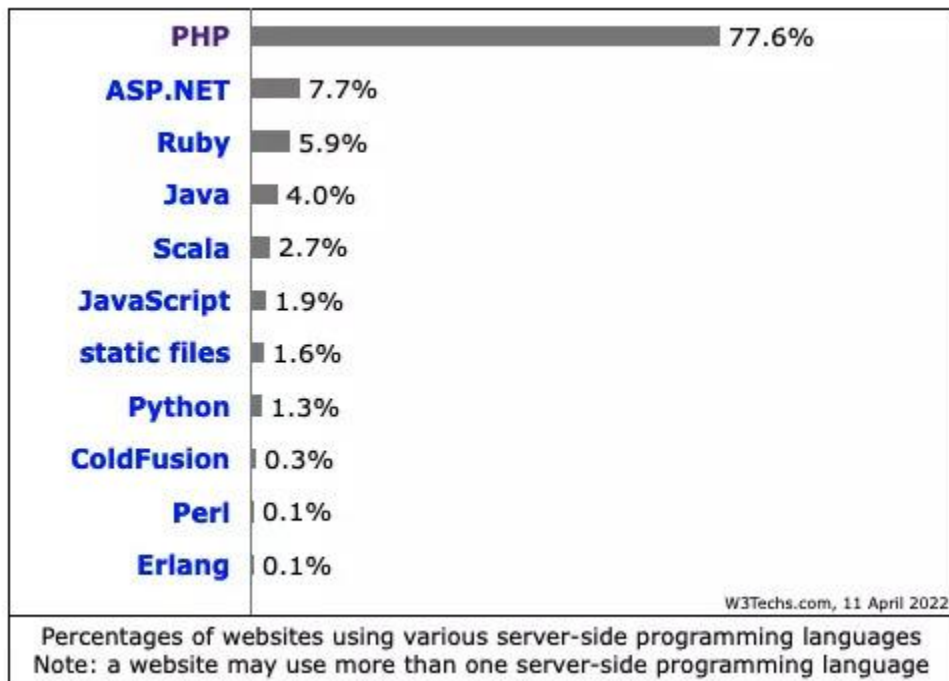
Exécution rapide,
peu/pas de temps de
chargement

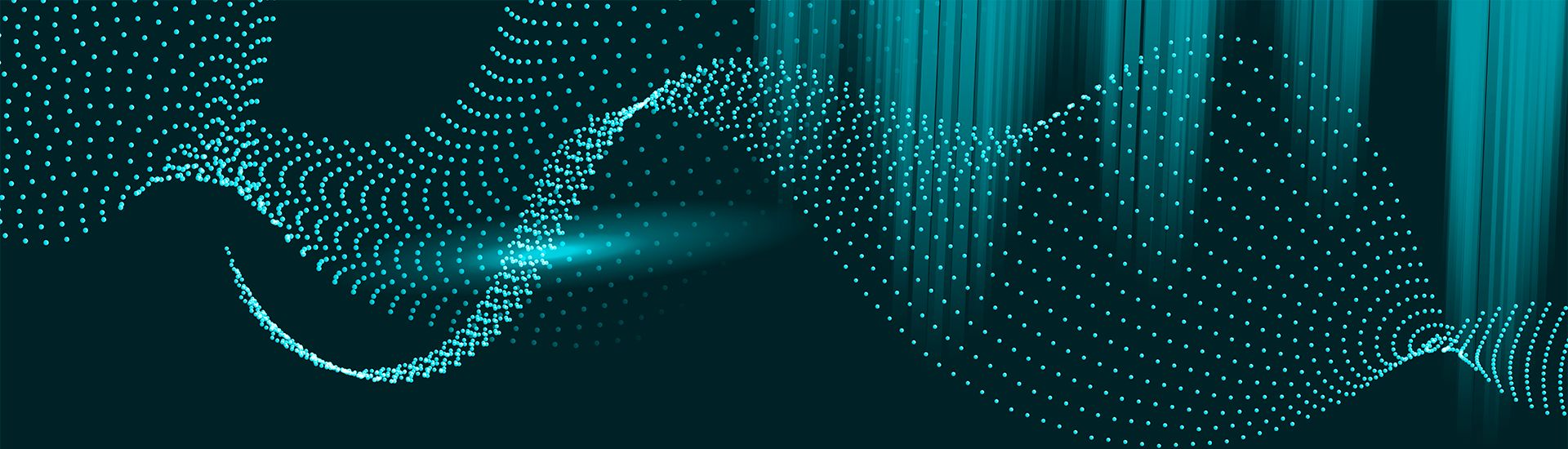
03

SUIVI

Suivi par PHP
Foundation,
Communauté
gigantesque

PHP une valeur sûre





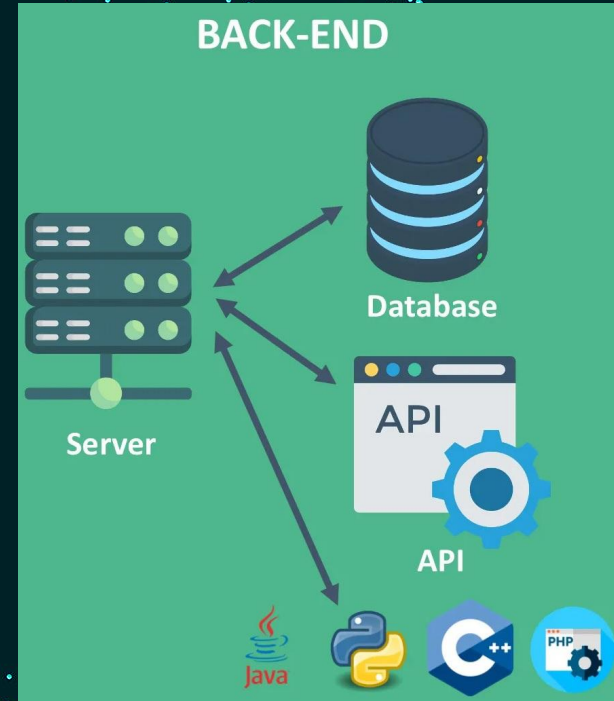
03

Architecture Back-End

Ses différents composants

Les composants principaux

- Le serveur
- La base de données
- Les APIs
- Le langage back-end



Qui s'occupe de quoi ?

Le Développeur



- Installation des langages
- Configuration de la BDD
- Gestion de l'environnement
- Configuration des routes
- Vérification des permissions

Le SysAdmin



- Installation et configuration serveur
- Sécurisation du serveur
- Gestion des permissions
- Mise en place de backlogs
- Sauvegarde et backup
- Optimisation des performances

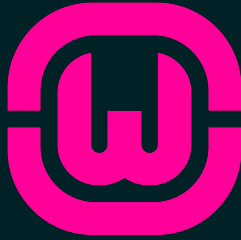
Stack LAMP Définition



Les alternatives en local



XAMPP



WAMP



MAMP



LAMP



EN AVANT !

Semaine 1 :

- Préparation serveur local
- PHP Procédural
- Ecosystème PHP

Semaine 2 :

- MySQL
- Sécurité des BDD
- Cryptographie

Semaine 3 :

- PHP Orienté Objet

Préparation du serveur local

Etape 1 :

- En fonction de votre poste de travail, installez un des soft de serveur local (wamp préféré pour windows, mamp pour mac)
- Vérifiez le bon fonctionnement en accédant à une page via l'url "localhost" dans votre navigateur puis à l'interface "PhpMyAdmin"

Etape 2 :

- Tentez d'installer un vrai stack LAMP sur votre ordinateur
- Installez Ubuntu via WSL sur windows ou homebrew sur MacOS
- Installez Apache2
- Installez PHP et updatez le en 8.2 au moins
- Installez MySQL
- Testez le fonctionnement en accedant à la page localhost

Les commandes Linux à exécuter

- Mettre à jour les dépôts (à exécuter régulièrement)

```
sudo apt update
```

- Installer Apache serveur (localhost devrait déjà fonctionner après cette étape)

```
sudo apt install apache2
```

- Installer MySQL et premier réglage de sécurité

```
sudo apt install mysql-server  
sudo mysql_secure_installation
```

Les commandes Linux à exécuter 2

- Installer PHP et l'update en 8.2+

```
sudo apt install php libapache2-mod-php php-mysql
sudo apt install software-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt update
sudo apt install php8.2
sudo apt install php8.2-cli php8.2-fpm php8.2-mysql php8.2-xml php8.2-mbstring php8.2-curl
sudo a2dismod php8.1
sudo a2enmod php8.2
sudo systemctl restart apache2
php -v
```

- Créer un fichier info.php dans le répertoire web d'apache (www/html)
- Y écrire "<?php phpinfo();?>" puis accéder à ce fichier via localhost/info.php

Les commandes Linux à exécuter 3

- Installer PHPMyAdmin

```
sudo apt install phpmyadmin
```

- Création d'un lien pour accéder à PHPMyAdmin

```
sudo ln -s /usr/share/phpmyadmin /var/www/html/phpmyadmin
```

Vérification de l'installation serveur

- Localhost accessible via "localhost" dans le navigateur
- Test du php en accédant à "localhost/info.php"
- Test du PHPMyAdmin et MySQL en accédant à "localhost/phpmyadmin"

Tout fonctionne ?
BRAVO !



Autres manipulation utiles

- Aidez vous de la documentation officielle :
<https://guide.ubuntu-fr.org/server/web-servers.html>
- Installer sur VSCode l'extension WSL
- Configurer d'autres Virtual Hosts sur Apache
- Changer le port d'écoute par défaut de Apache
- Afficher les erreurs PHP
- Définir une limite mémoire accessible par PHP
- Configuration et sécurisation de PHP-FPM (FastCGIProcessManager)
- Sécurisation de MySQL
- Optimisation de MySQL
- Sécurisation de PHPMysqlAdmin
- Tests de performances avec Apache Bench (ab)
- Configurer le HTTPS, le SSL, un firewall, etc.



PHP

- Voir fichiers de code



Ecosystème PHP

- Les Frameworks PHP



Symfony



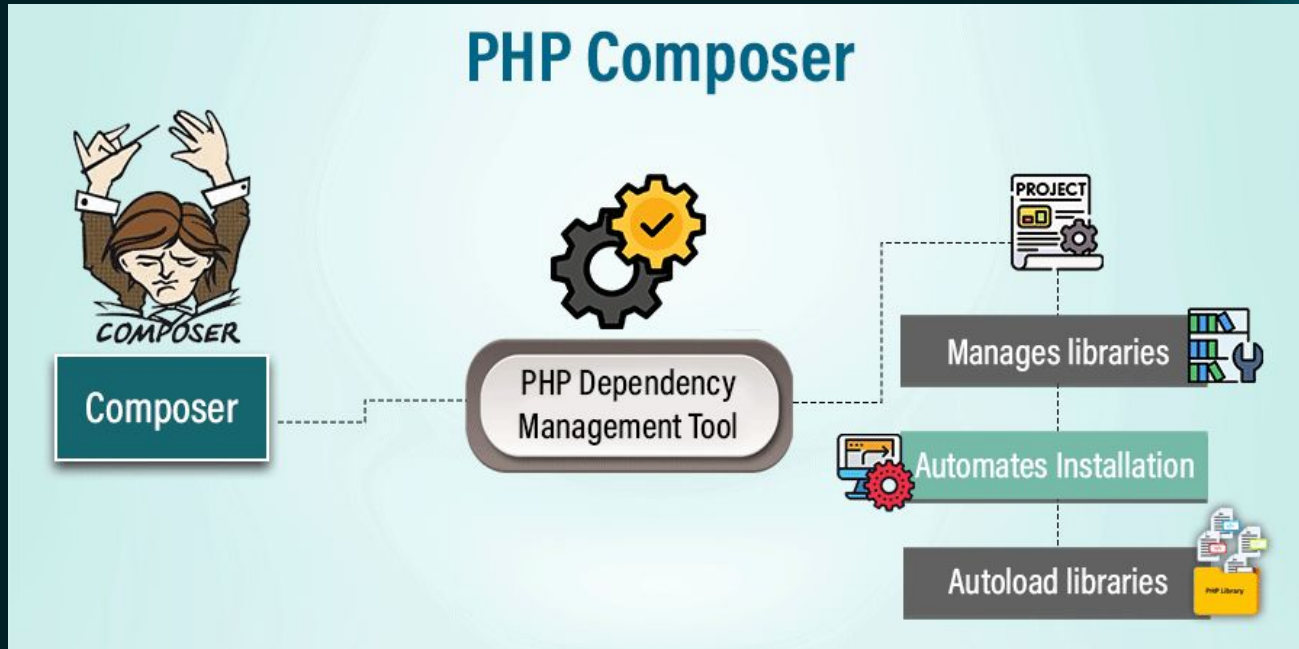
Laravel

Composer

- Gestionnaire de dépendances PHP
 - Dépendance = bibliothèque externe
 - Automatisation du processus d'installation
 - Récupération de projets plus aisée
 - Mises à jour simplifiées



Composer



Installation de Composer

<https://getcomposer.org/>

- Windows
 - Installation via setup globalement sur le pc
- Mac & Linux
 - Installation via command line dans le dossier du projet
- Pour vérifier :

```
composer -v
```

Installation d'une librairie

<https://packagist.org/>

- Exemple de librairie générant des unique id avec ramsey/uuid

```
composer require ramsey/uuid
```

- Création automatique de composer.json, composer.lock, /vendor/

Fichiers/Dossiers auto-générés

- `composer.json`
 - Contient la liste des dépendances utilisées dans le projet + leur version
- `composer.lock`
 - Contient les packages entier et versions exactes installées sur ce projet
- `/vendor/`
 - Dossier d'installation des dépendances

Les commandes importantes

- **composer require** “nom du package/librairie”
 - Télécharge le paquet spécifique
 - Installe le paquet dans /vendor/
 - Inscrit les détails dans le composer.json et .lock
- **composer install**
 - Installe toutes les librairies dans un projet en se basant sur un composer.lock récupéré (en cas de récupération de code)
- **composer update**
 - Met à jour les dépendances, le fichier composer et .lock

Utilisation d'une dépendance

- **Vérifier la documentation**

- <https://uuid.ramsey.dev/en/stable/>

- **Toujours ajouter l'autoload de composer**

- C'est ce qui permet de include/require les bons fichiers automatiquement

```
// Autoload de composer  
require __DIR__ . '/../vendor/autoload.php';
```

- **Importer la dépendance sur le fichier**

- C'est ce qui nous permet de l'utiliser directement

```
// Importation de la dépendance sur ce fichier  
use Ramsey\Uuid\Uuid;
```

Suivre la documentation pour comprendre l'utilisation

Using ramsey/uuid

After installing ramsey/uuid, the quickest way to get up-and-running is to use the static generation methods.

```
use Ramsey\Uuid\Uuid;

$uuid = Uuid::uuid4();

printf(
    "UUID: %s\nVersion: %d\n",
    $uuid->toString(),
    $uuid->getFields()->getVersion()
);
```

Comment ferions nous sans composer ?

- **Avec des archives .phar**

- PHp ARchive
- Comprime plusieurs fichiers PHP (mais pas seulement) en un seul.
- Permet de transporter facilement nombreux fichiers PHP
- Peut aider à la mise en prod en transportant un seul fichier
- Les fichiers PHP sont lisibles à l'intérieur du .phar

Quelques autres dépendances pour s'entraîner

- **Guzzle** : pour une meilleure gestion des requêtes API
- **Carbon** : pour la gestion de la date et heure
- **ramsey/Collection** : pour la gestion de collection typées d'objets
- **monolog** : pour la gestion de logs d'erreurs

