RAPPORT DE STAGE

du 15 Janvier au 23 Février

Maître de stage: Monsieur Olivier Haro

Année 2024

ANDRIAMANANTENA Tahiniavo Chrissy

SOMMAIRE

SEMAINE 1

CAHIER DE CHARGE HEBDOMADAIRE (Semaine 1):

- S’informer sur Laravel et rédiger une documentation

Initiation au framework Laravel:

Semaine de formation et d’initiation (assez calme):

- Réunion à titre informatif avec le tuteur de stage
- recherches permettant de répondre aux questions suivantes (avec documentation):

Lundi :

- Qu’est-ce que Laravel et pourquoi l’utiliser ?
- Quels en sont ces avantages ?

  - Laravel est un framework open-source de développement web écrit en PHP. Il a été créé par Taylor Otwell et est largement utilisé pour la création d'applications web robustes et évolutives. Lancé en 2011, Laravel est devenu l'un des frameworks PHP les plus populaires grâce à sa syntaxe élégante, sa simplicité d'utilisation et ses fonctionnalités puissantes.

  - Syntaxe Élégante : Laravel offre une syntaxe propre et élégante qui facilite la lecture et l'écriture du code. Cela contribue à accélérer le développement en rendant le code plus expressif.

  - Modèle MVC : Laravel suit le modèle MVC (Modèle-Vue-Contrôleur), une architecture qui sépare la logique métier, la présentation et la

gestion des requêtes. Cela améliore la maintenabilité du code et permet une meilleure organisation de l'application.

- Eloquent ORM : Laravel intègre Eloquent, un ORM (Object-Relational Mapping) qui simplifie l'interaction avec la base de données. Il permet aux développeurs de travailler avec la base de données en utilisant des modèles et des relations orientées objet, plutôt qu'en écrivant des requêtes SQL directes.

- Système de Routage Puissant : Laravel propose un système de routage simple et puissant qui permet de définir facilement les URL de l'application et de diriger les requêtes vers les contrôleurs appropriés.

- Gestion des Dépendances avec Composer : Laravel utilise Composer, un gestionnaire de dépendances PHP, pour simplifier l'installation et la gestion des bibliothèques tierces, ce qui facilite l'intégration de composants externes dans le projet.

- Système de Modulaire : Laravel est conçu de manière modulaire, ce qui signifie que vous pouvez utiliser uniquement les composants dont vous avez besoin. Cela rend l'application plus légère et offre une flexibilité accrue.

- Système d'Authentification Intégré : Laravel propose un système d'authentification prêt à l'emploi qui simplifie la gestion des utilisateurs, des sessions et des rôles. Il offre également des fonctionnalités de sécurité avancées, telles que la protection CSRF (Cross-Site Request Forgery) et la protection XSS (Cross-Site Scripting).

- Support de la Communauté Active : Laravel bénéficie d'une communauté active de développeurs. La documentation est complète, et il existe de nombreux tutoriels, forums et packages qui facilitent l'apprentissage et la résolution de problèmes.

Mardi :

- Approfondissement sur la structure des fichiers et dossiers sur un projet Laravel (+ documentation)

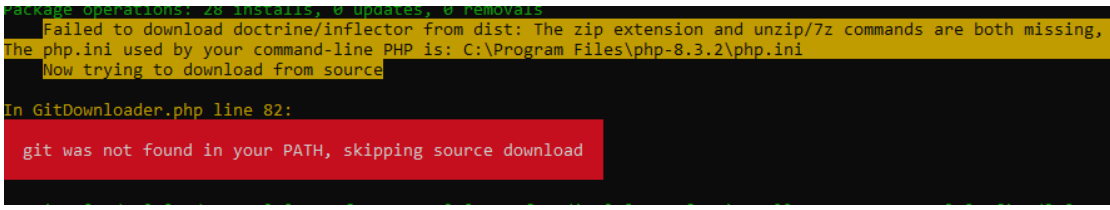
- formation en autonomie (vidéos youtube principalement, documentations officiels de laravel)

Mercredi :

- formation en autonomie de Laravel (apprentissage des bases):
    - découverte des similitudes avec Symfony:
        - Même structure : MVC (modèles vues controllers)
        - même concept de routes etc…

Jeudi et vendredi:

- Après avoir revu la théorie sur le framework, 1ère tentative “d’installation” et de lancement d’ un projet Laravel:
    - Installation de la dernière version de PHP
    - Installation de composer
    - 1 erreur inattendue:



    -

Tentatives de résolution du problème (résultat échéant)

SEMAINE 2

CAHIER DE CHARGE HEBDOMADAIRE (semaine 2):

- Installer Laravel et réussir à créer un premier projet

Lundi:

- tentative de résolution de l’erreur:
    - téléchargement de GIT

    - ajout de GIT dans le PATH => le variable d'environnement de l'ordinateur
    - modification de php.ini

- Recherche sur la cause de l'erreur

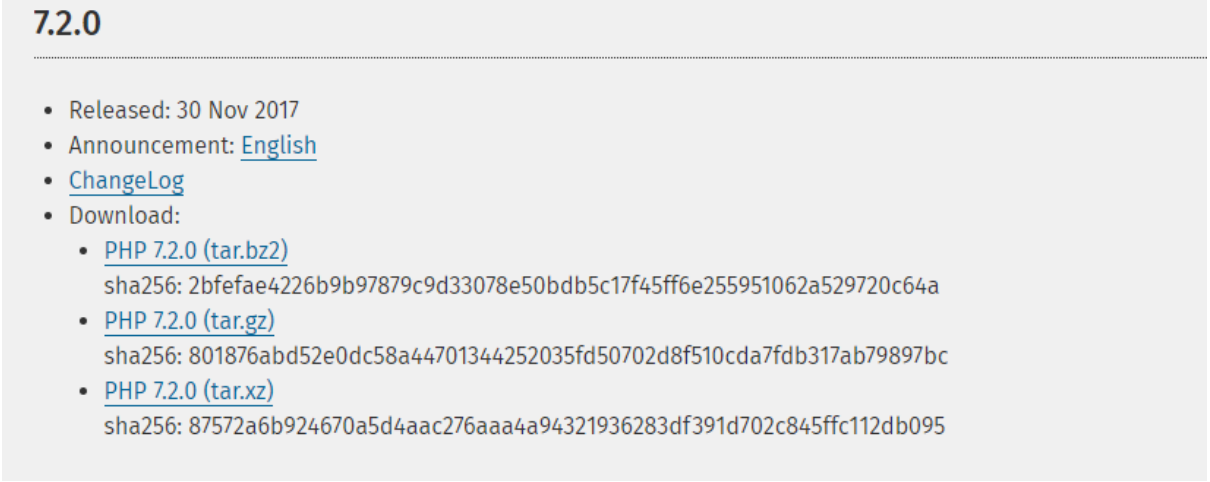
Mardi:

- Réunion avec un développeur de l'entreprise dans le but de résoudre l'erreur:

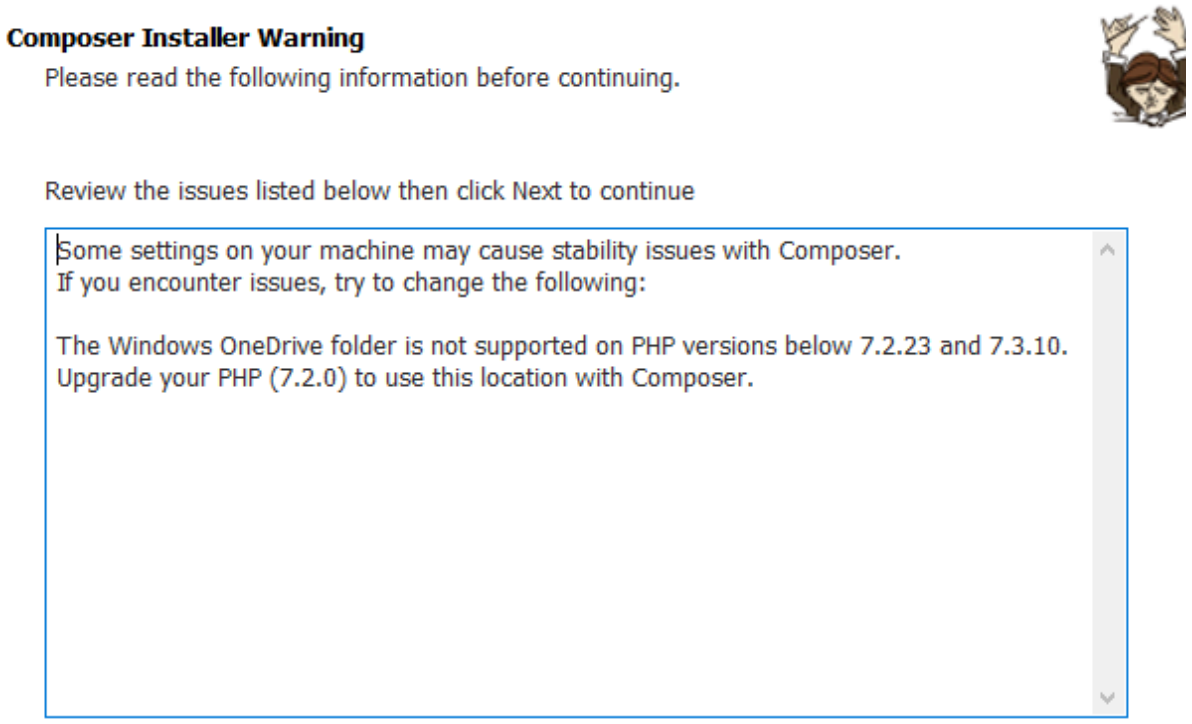
    - Il s'avère que la version de PHP et de Laravel utilisé par l'entreprise est une ancienne version
    - Il faut donc installer les mêmes versions que l'entreprise.
    - Laravel 5.4 et php 7.2 (version de 2017)
    - recherche sur comment installer des versions spécifiques et tous les prérequis pour.

Mercredi:

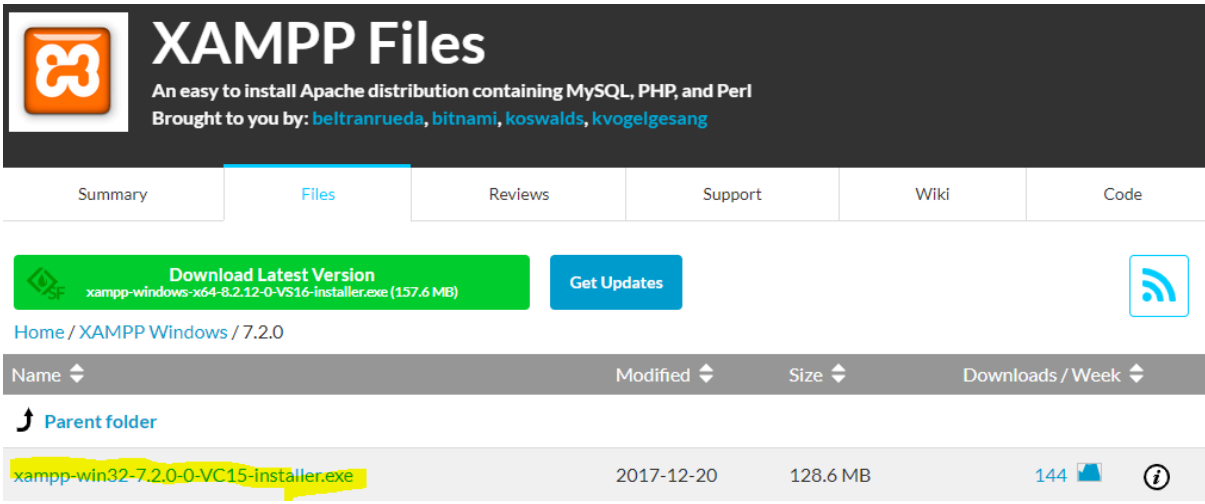
- Désinstallation de Laravel et de PHP

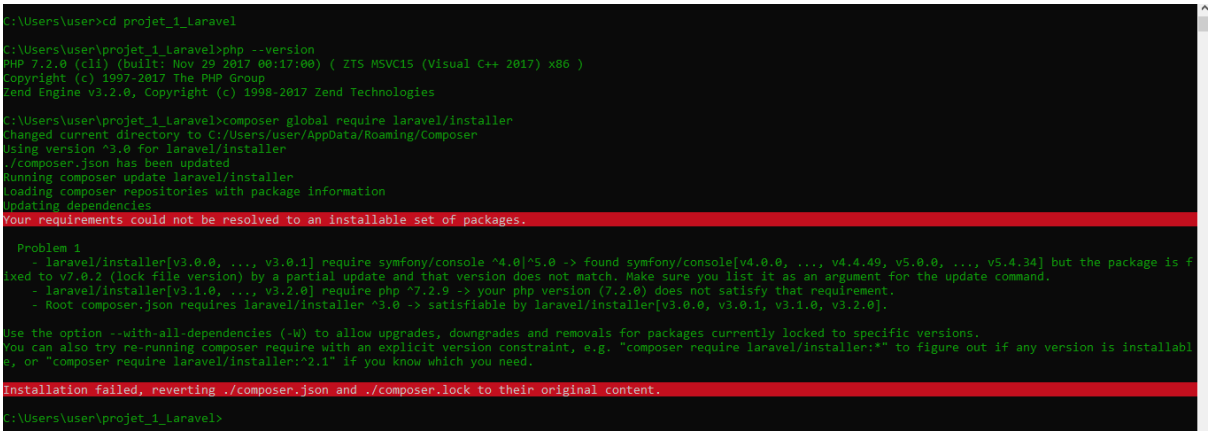


-



-
- Solution trouver : Installer PHP grâce à XAMPP:
    - recherche de la version de XAMP qui installera la version 7.2 de PHP:





- Composer semble ne pas vouloir installer les “requierement” en raison d’une version trop ancienne de PHP

JEUDI et VENDREDI

- Recherche d’un compromis pour pour pouvoir utiliser Laravel 5.4:
    - Création d’un nouveau dossier où l’on va tester un projet Laravel avec la version spécifié c’est à dire 5.4
    -
```
C:\Users\user\projet_1_Laravel>composer create-project --prefer-dist laravel/laravel nom-du-projet "5.4.*"
Creating a "laravel/laravel" project at "./nom-du-projet"
Installing laravel/laravel (v5.4.30)
  - Downloading laravel/laravel (v5.4.30)
  - Installing laravel/laravel (v5.4.30): Extracting archive
Created project in C:\Users\user\projet_1_Laravel\nom-du-projet
> php -r "file_exists('.env') || copy('.env.example', '.env');"
```
- Aucun message d’erreur:
    - Test d’Apache sur le localhost port 80 : succès
    - Test de Laravel:
        - “php artisan serve” : succès

```
:\Users\user\projet_1_Laravel\nom-du-projet>php artisan serve
aravel development server started: <http://127.0.0.1:8000>
```

- Après tout les erreurs et tentatives, Laravel semble enfin être installé:

Laravel

DOCUMENTATION LARACASTS NEWS FORGE GITHUB

Semaine 3

Lundi:

Travail sur les fondamentaux et la configuration de l'environnement Laravel:

- La structure d'un projet Laravel:
    - La structure de d'un projet Laravel est conçue pour suivre le modèle MVC:
        - La structure d'un projet Laravel:
            - Dossiers racine :

            - app: Ce dossier contient le code source de l'application Laravel, y compris les modèles, les contrôleurs, les middleware et d'autres classes PHP.
            - bootstrap: Ce dossier contient le fichier app.php qui charge les dépendances et initialise l'application.
            - config: Les fichiers de configuration de l'application, tels que les fichiers de configuration de la base de données, de l'authentification, etc.
            - database: Les migrations de base de données et les seeders sont stockés ici.
            - public: C'est le point d'entrée de votre application. Le fichier index.php dans ce dossier est le point de départ pour toutes les requêtes HTTP.
            - resources: Contient des ressources qui ne sont pas compilées, telles que les fichiers de vue Blade, les fichiers de langues, les fichiers Sass, etc.
            - routes: Les routes de votre application sont définies dans ce dossier, dans les fichiers web.php, api.php, etc.
            - storage: Les fichiers générés par l'application, tels que les logs, les sessions, les fichiers temporaires, etc., sont stockés ici.
            - tests: Contient les tests unitaires et fonctionnels de votre application.
            - Dossiers dans 'app':

            - Console: Contient les commandes de console personnalisées de votre application.
            - Exceptions: Gère les exceptions personnalisées.
            - Http: Contient les contrôleurs, les middlewares et les demandes de votre application.
            - Models: Les modèles Eloquent de votre application sont stockés ici.
            - Providers: Contient les fournisseurs de services de votre application.
            - Traits: Les traits réutilisables sont stockés ici.
            - Dossiers dans 'resources':

- assets: Contient les ressources frontales comme les fichiers JavaScript, les fichiers CSS, les images, etc.
- lang: Les fichiers de localisation sont stockés ici.
- views: Les fichiers de vue Blade, qui définissent la présentation de votre application.
- Fichiers de configuration importants :

- .env: Fichier d'environnement contenant les variables d'environnement spécifiques à l'installation.
- config/app.php: Configuration de base de l'application Laravel.
- config/database.php: Configuration de la base de données.
- Autres fichiers importants :

- artisan: Interface en ligne de commande pour exécuter des commandes Laravel telles que les migrations, la création de contrôleurs, etc.
- composer.json: Fichier de configuration de Composer pour gérer les dépendances PHP.
- package.json: Fichier de configuration de npm pour gérer les dépendances JavaScript.
- webpack.mix.js: Fichier de configuration pour Laravel Mix, qui simplifie la compilation des ressources frontales.

Mardi

Travail sur le modèle vu contrôleur:

- Comme je possède déjà quelques connaissances sur le MVC grâce à PHP, je suis directement passé à la pratique:
    - création des modèles sur Laravel:
        - Sur Laravel, il existe une commande qui permet de créer un modèle:
            - php artisan make:model nom_du_model: celà va créer Cela va créer un nouveau fichier `User.php` dans le dossier `app/Models` par défaut,

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    //
}
```

test de création d’un utilisateur “User”

Celà a créé un nouveau fichier User.php dans le dossier “`app/Models`”

- Les contrôleurs dans Laravel sont des classes PHP qui organisent la logique métier de l' application. Ils traitent les requêtes HTTP entrantes, interagissent avec les modèles pour récupérer ou manipuler des données, puis renvoient les réponses appropriées (par exemple, des vues, des redirections, des réponses JSON, etc.).

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    protected $fillable = ['name', 'email', 'password'];

    // Exemple de relation avec un autre modèle
    public function posts()
    {
        return $this->hasMany(Post::class);
    }
}
```

Mercredi

Travail sur les vues: Sur Laravel, il existe déjà un modèle de template, si sur PHP c'est les vues Twig, sur Laravel, on parlera de vue Blade. Blade offre une syntaxe concise et expressive pour écrire des vues HTML, tout en offrant des fonctionnalités telles que l'héritage de mise en page, l'inclusion de sous-vues, les directives de contrôle et plus encore.

Pour se faire il faut:

Créer un fichier de vue : Créer un nouveau fichier de vue dans le répertoire resources/views. Par exemple, si l'on souhaite créer une vue pour afficher la liste des utilisateurs, vous pouvez créer un fichier users.blade.php.

Écrire du code Blade : À l'intérieur du fichier de vue, on peut utiliser la syntaxe Blade pour écrire du code HTML avec des fonctionnalités Blade. Par exemple, on peut utiliser des directives Blade comme @foreach, @if, @else, @include, etc. pour dynamiquement générer du contenu HTML.

-
```
{{-- resources/views/users.blade.php --}}

@extends('layouts.app')

@section('content')
    <h1>Liste des utilisateurs</h1>

    <ul>
        @foreach ($users as $user)
            <li>{{ $user->name }}</li>
        @endforeach
    </ul>
@endsection
```

JEUDI et VENDREDI

Travail et recherche sur Eloquent ORM,

- Eloquent ORM est un outil de mapping objet-relationnel (ORM) intégré dans le framework Laravel. Il simplifie la gestion des bases de données en utilisant des objets PHP pour représenter les tables de la base de données. Eloquent rend la manipulation des données plus intuitive et expressive en permettant aux développeurs de travailler avec des données de manière orientée objet plutôt que de se concentrer sur des requêtes SQL. J'avais peur de comment m'y prendre mais je me suis rendu compte au fil des temps que c'était l'équivalent de "Doctrine" sur symfony.

- Les modèles Eloquent sont des classes PHP qui représentent des tables de base de données. Chaque modèle Eloquent est associé à une table de base de données, et chaque instance de modèle représente une ligne dans cette table.

- Opérations CRUD (Create, Read, Update, Delete)
  Eloquent simplifie les opérations CRUD en fournissant des méthodes pour créer, récupérer, mettre à jour et supprimer des enregistrements de la base de données.

  Exemple simple de structure de code:

  Create:

```
$user = User::create(['name' => 'John Doe', 'email' => 'john@example.c
```

  Lecture:

```
$users = User::all();
$user = User::find(1);
```

  Mise à jour:

```
$user = User::find(1);
$user->name = 'Jane Doe';
$user->save();
```

  suppression:

```
$user = User::find(1);
$user->delete();
```

- Il existe également plusieurs types de relations entre les classes:

- voici comment ils sont structurés:
  - Les relation OneToOne ou Un-à-Un:

```php
class User extends Model
{
    public function phone()
    {
        return $this->hasOne('App\Phone');
    }
}
```

- Les relation OntoMany ou Un-à-plusieurs:

```php
class Post extends Model
{
    public function comments()
    {
        return $this->hasMany('App\Comment');
    }
}
```

- Les relations ManyToMany ou plusieurs à plusieurs:

```php
class User extends Model
{
    public function roles()
    {
        return $this->belongsToMany('App\Role');
    }
}
```

Semaine 4

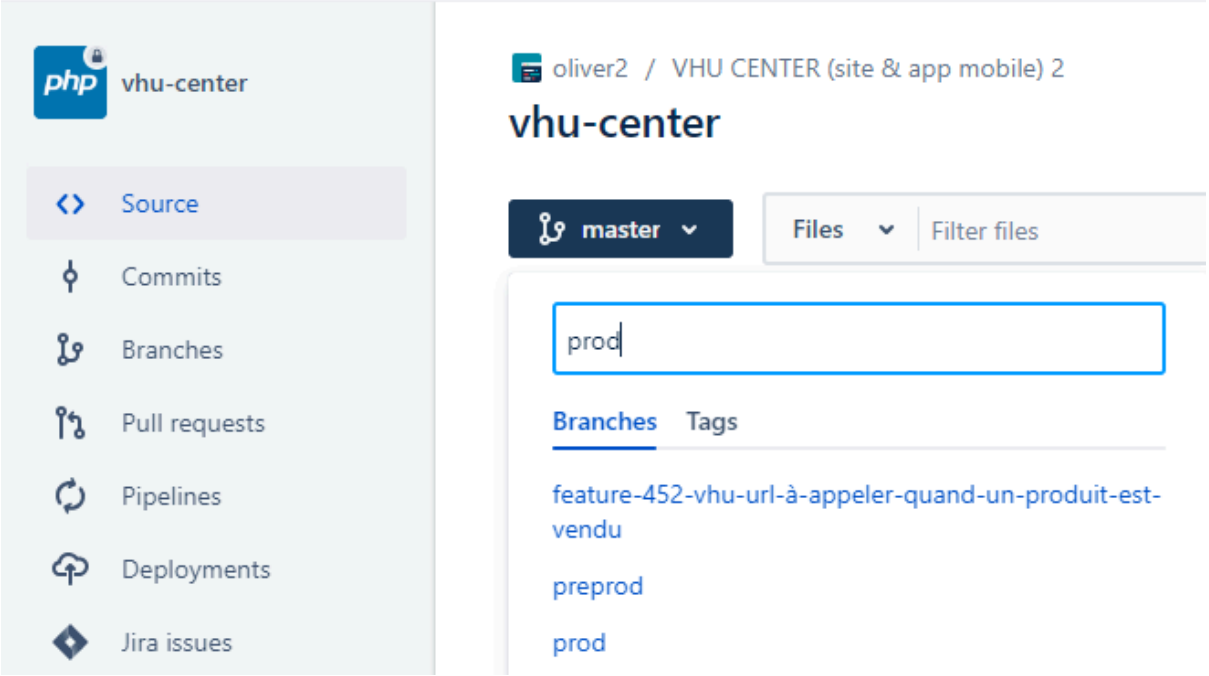
Lundi au Vendredi

Semaine intensive avec beaucoup de session d'erreur

Meeting avec un développeur de la société pour parler du site web et du groupe de projet “vhu”

Début du travail sur le projet “vhu”

L’objectif est de cloner le projet qui se trouve dans la branche Origin => Prod



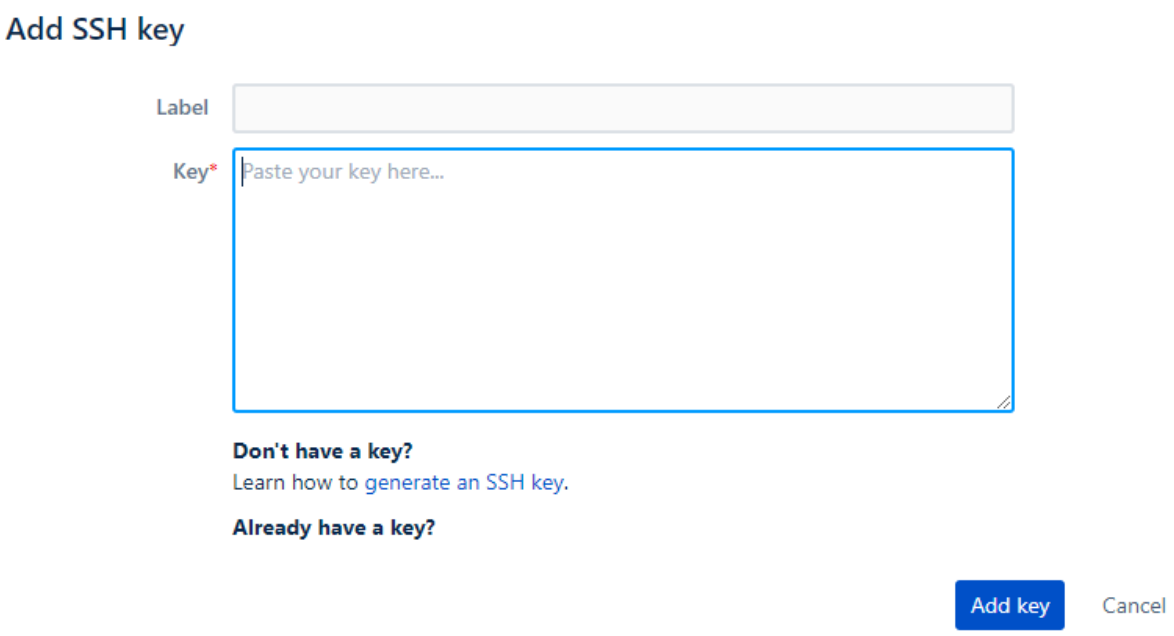
Pour ça il a fallu créer et configurer un compte Bitbucket

Utilisation de Bitbucket : création de compte, clé SSH pour git et PHP storm etc…

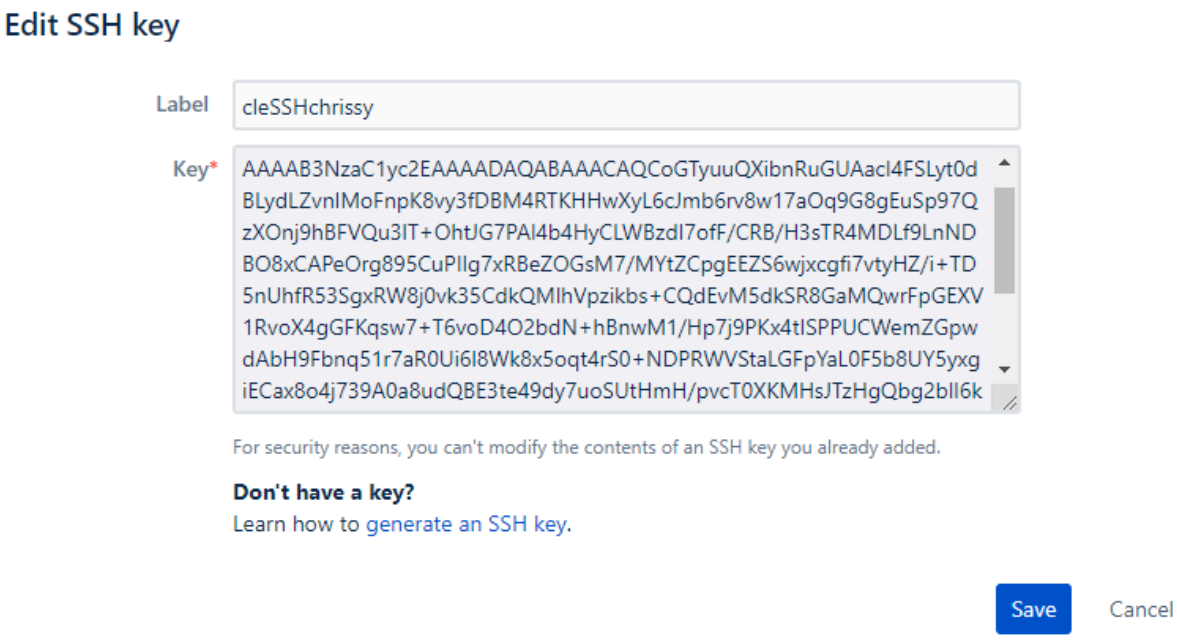
Sans celà, il est impossible de s’authentifier

COMMENT CRÉER UNE SIGNATURE pour git ?

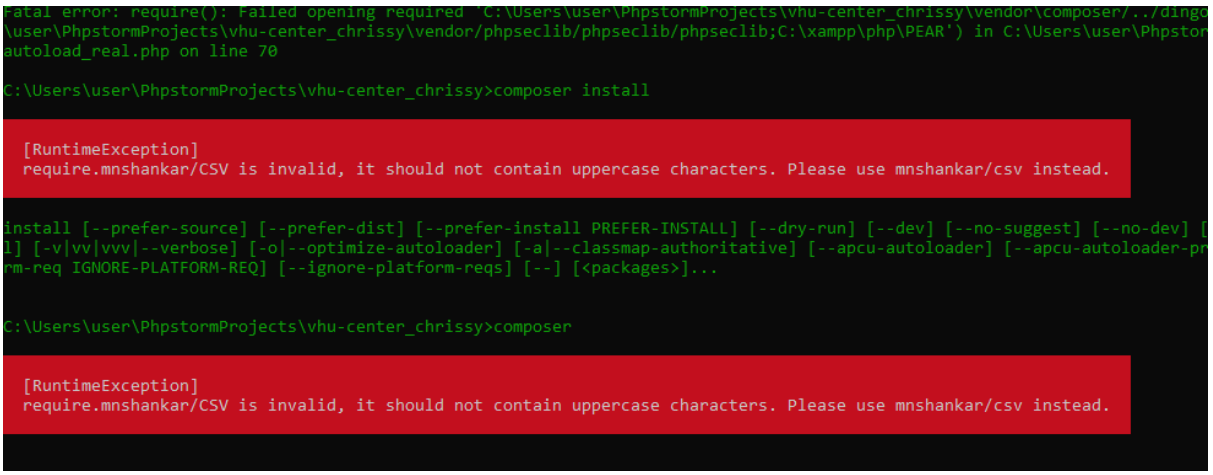
- Générer une clé avec la commande: php artisan make:command GenerateAppKey
- ssh-keygen -t rsa -b 4096 -C "chrissyandri.com"
- copie de la clé



Pour celà, il a fallu coller la clé SSH:



création de compte wetransfert pour envoi d'un vendor potable en raison d'une erreur qui est survenu lors de l'essai de lancement.

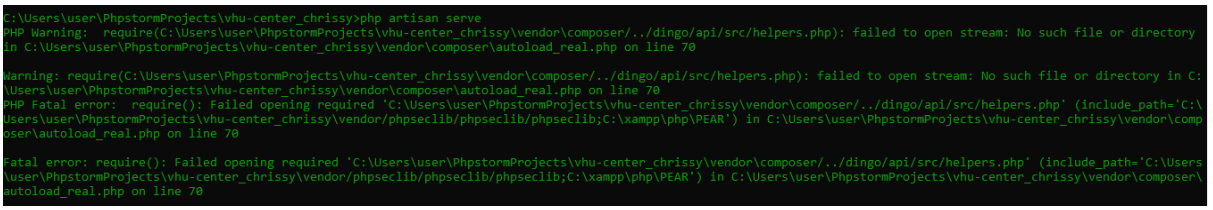


Clonage du projet à utiliser qui se trouve dans la branche “prod”
erreur de composer:

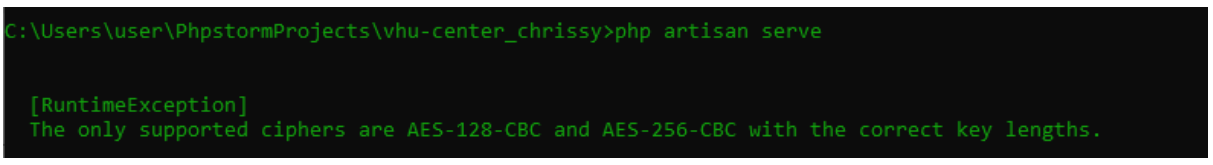
exécution du script SQL permettant d'accéder à la base de donnée

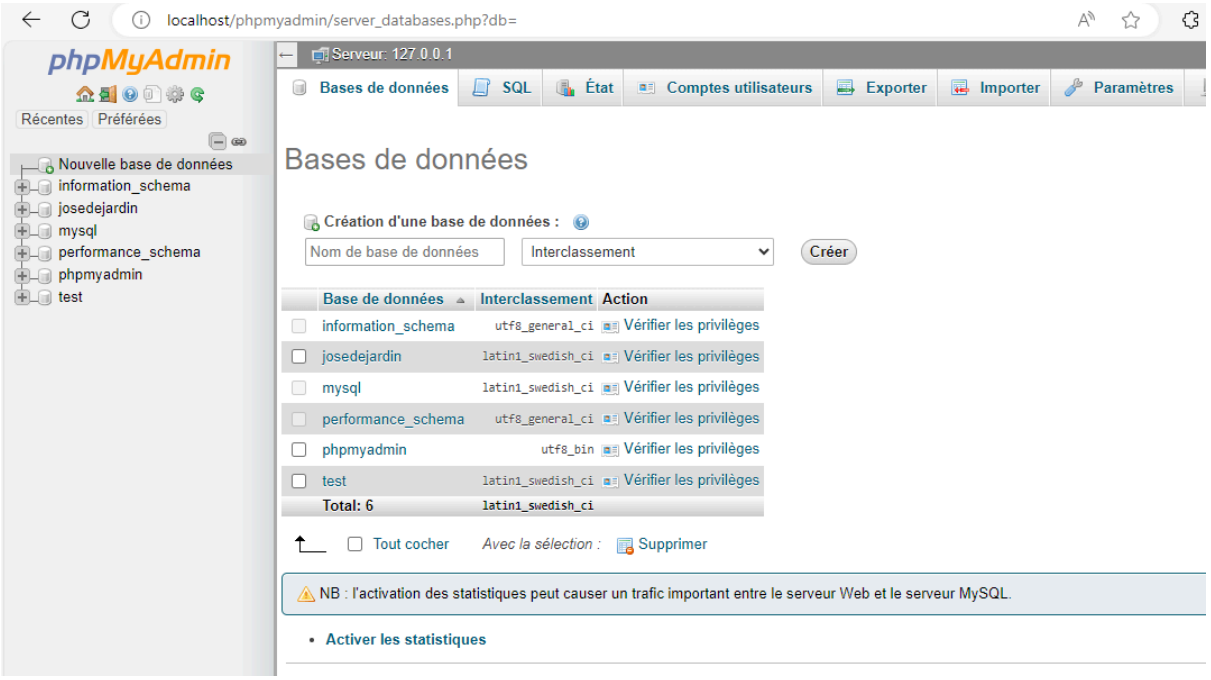
il fallait mettre en minuscule le “CSV”

erreur sur le lancement du projet:

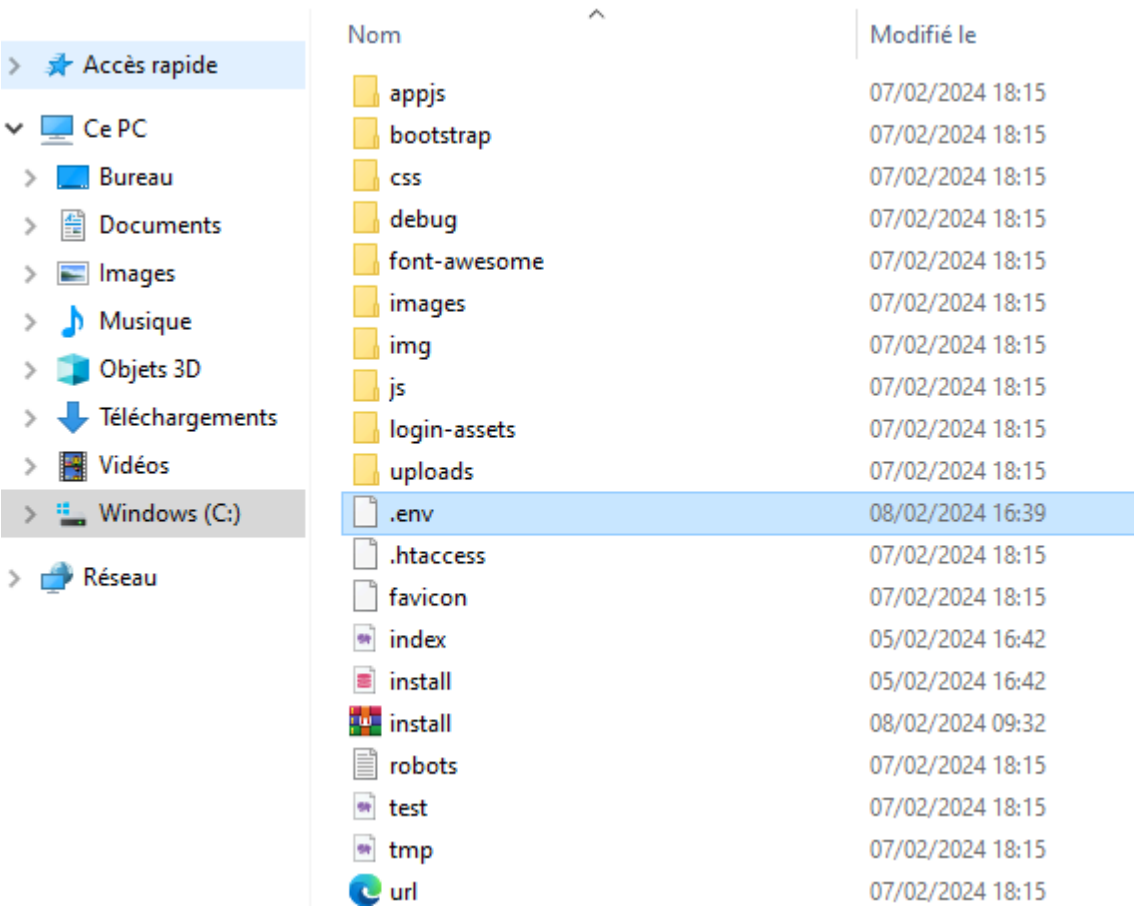


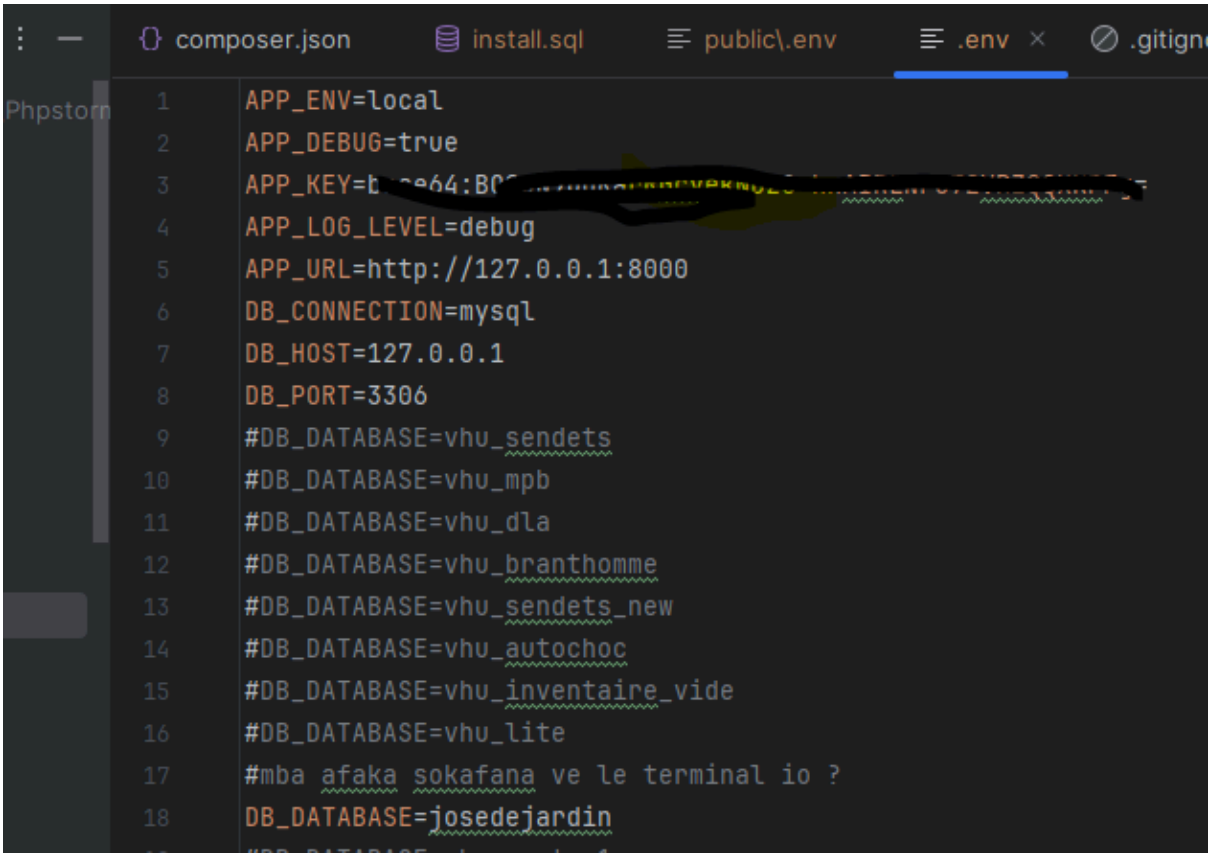
php artisan auto-reload comme solution trouvé mais erreur différent





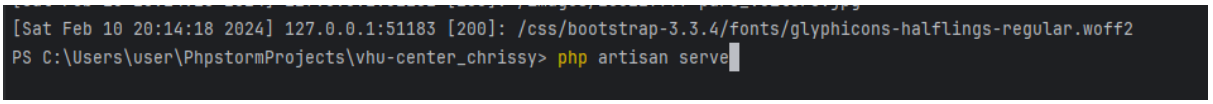
Configuration du fichier .env pour permettre la connexion et la relation avec la base de donnée:





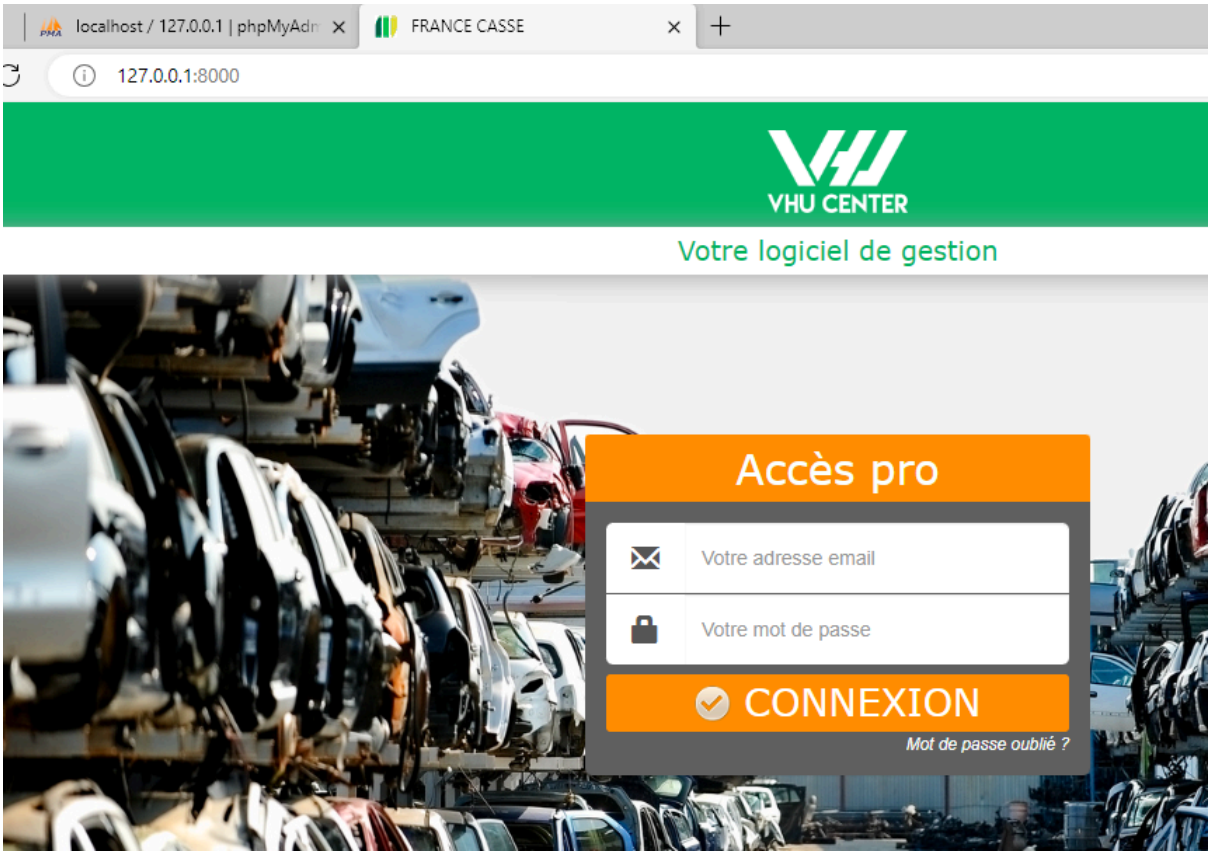
La database fut fonctionnelle et les tests sur la database fut un succès mais lancer le projet Laravel fut un échec.

Un meeting avec un développeur fut alors engagé dans le but de régler l'erreur manquante:



Un test final fut exécuté, et le résultat fut un succès

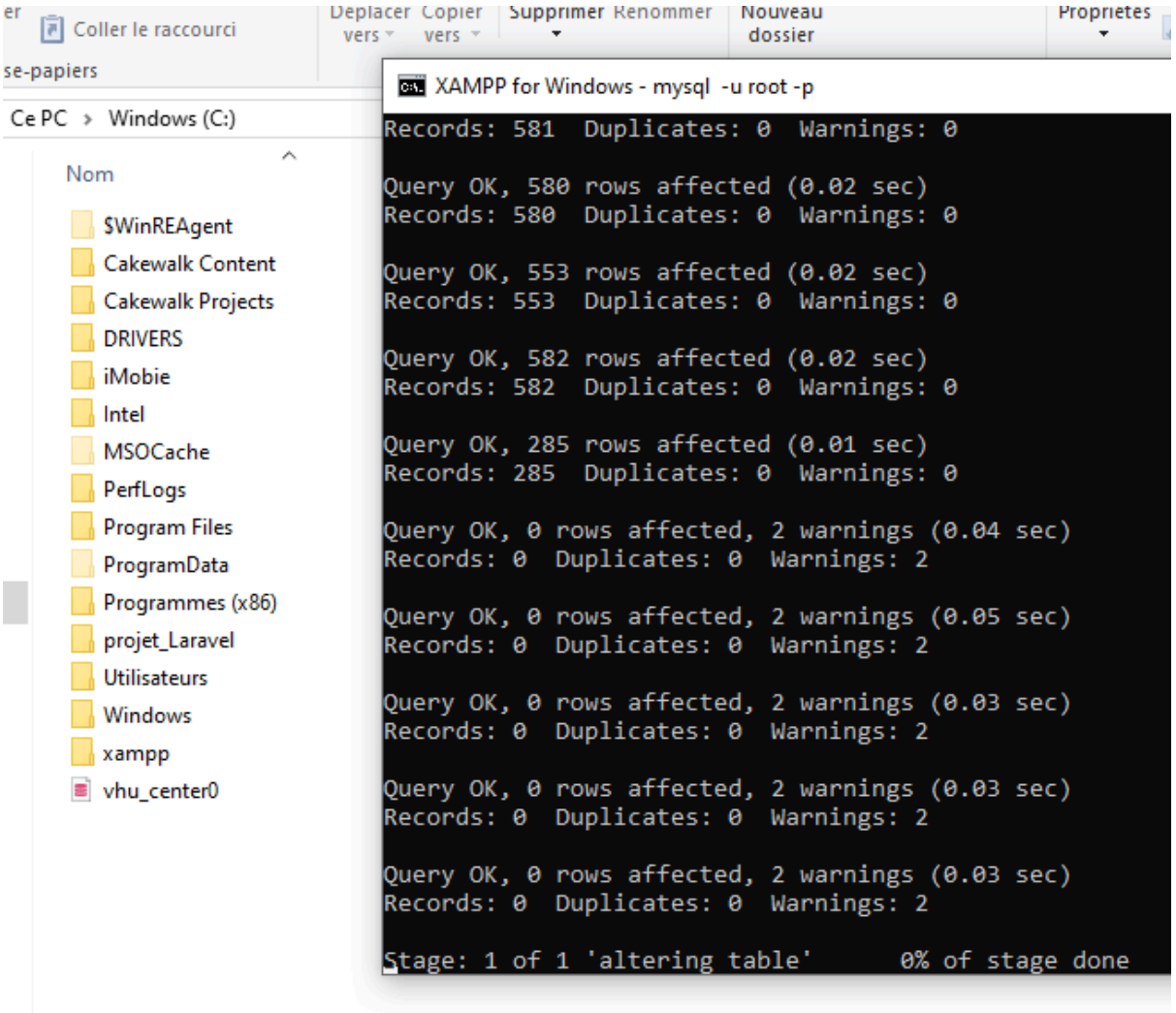
Voici ce que donne l'application:



Semaine 5

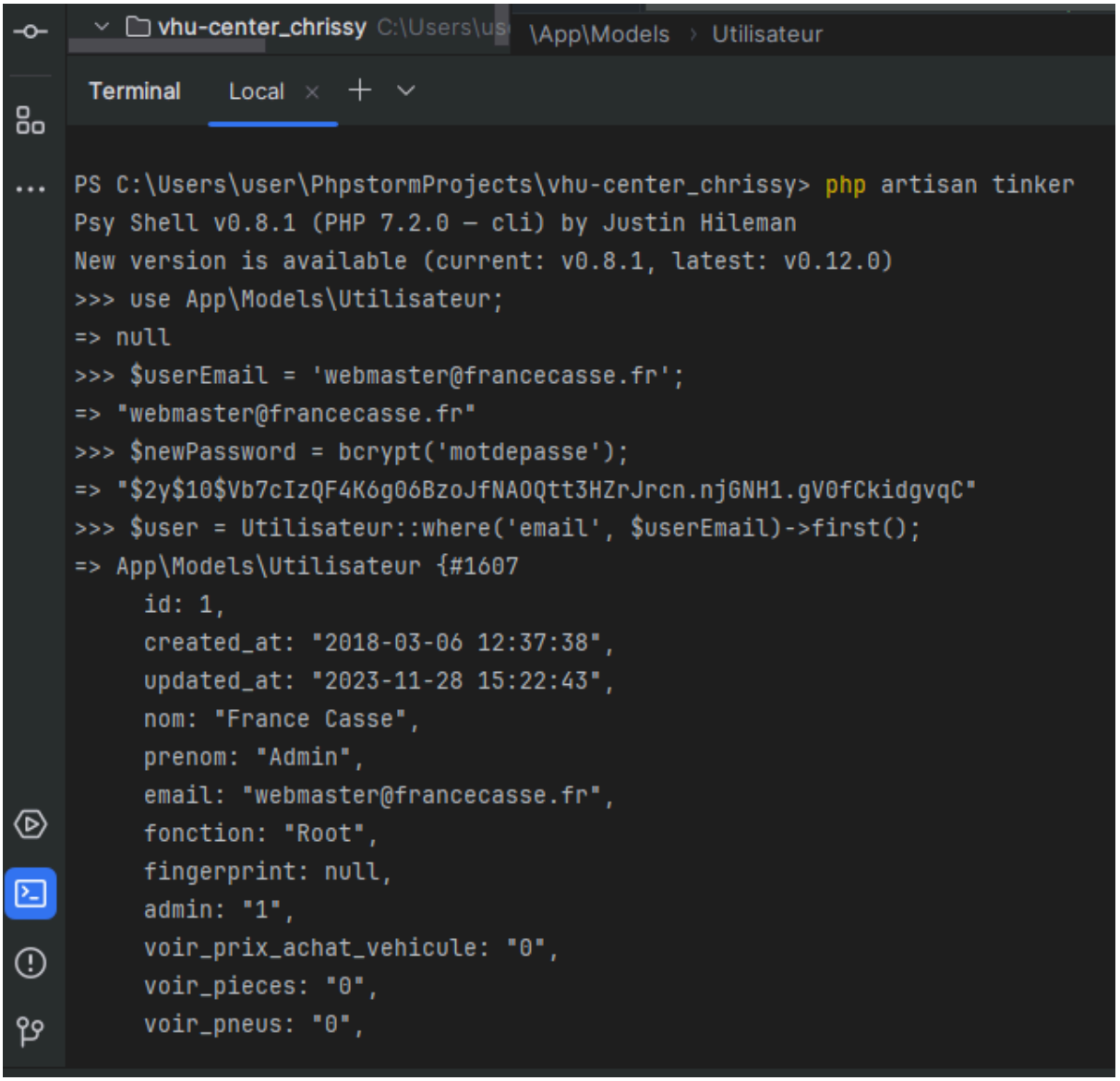
Cette semaine porte sur l'authentification Laravel:

- Une table "utilisateur" est présente dans la base de donnée:
    - Cette table contient tous les utilisateurs qui sont enregistrés dans la base de donnée
    - Dans le script fourni, il existe déjà des utilisateurs avec les droits nécessaires à utiliser, cependant il faut régénérer le mot de passe de cet utilisateur pour pouvoir s'authentifier sur l'application.



- Réimportation de la base de données car la table “Utilisateur” manquait:
    - Pour celà il faut:
        - déplacer le script dans un dossier accessible
        - connexion à la base de donnée Mysql
        - charger le script à l’aide de la commande:
            - source C:/le_chemin_du_script

- Génération d’un nouveau mot de passe pour l’utilisateur webmaster@francecasse.fr à l’aide de “ php artisan tinker”, qui permet de coder et d'interagir avec l’application Laravel en temps réel



- Après avoir consulter les documentations et tutoriels, le code final qui fut exécuté dans le Tinker:

```
use App\Models\Utilisateur;
$user = Utilisateur::where('email', 'webmaster@francecasse.fr')->first();
$user->password = bcrypt('motdepasse');
$user->save();
```

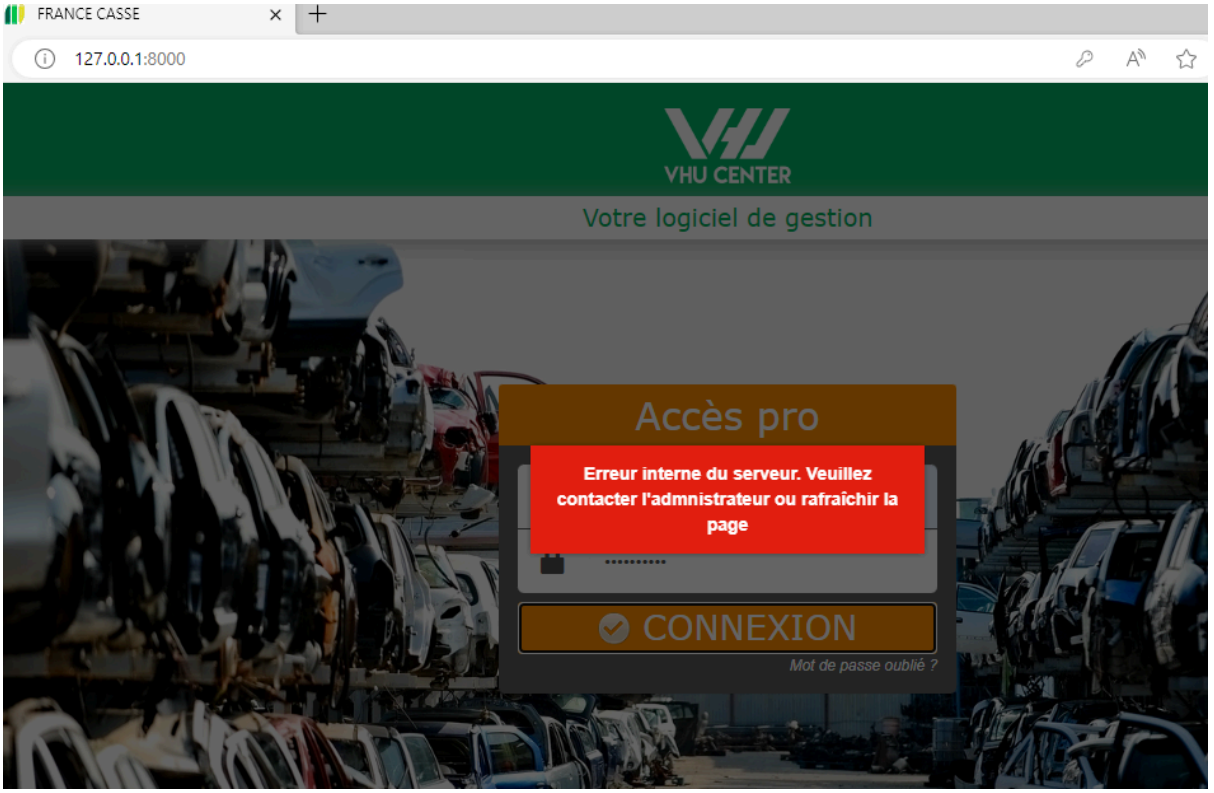
Ce code permet de mettre à jour le mot de passe de l'utilisateur "webmaster@francecasse.fr" par l'utilisation de la fonction 'bcrypt' et en sauvegardant les modifications dans la base de données. Grâce à celà, le mot de passe sera crypté dans la base de données.
Ici, le nouveau mot de passe local fut défini par "motdepasse".

```
>>>  $user = Utilisateur::where('email', 'webmaster@francecasse.fr')->first();
=> App\Models\Utilisateur {#1606
     id: 1,
     created_at: "2018-03-06 12:37:38",
     updated_at: "2024-02-13 11:32:27",
     nom: "France Casse",
     prenom: "Admin",
     email: "webmaster@francecasse.fr",
     fonction: "Root",
     fingerprint: null,
     admin: "1",
     voir_prix_achat_vehicule: "0",
     voir_pieces: "0",
     voir_pneus: "0",
     voir_pieces_neuves: "0",
```

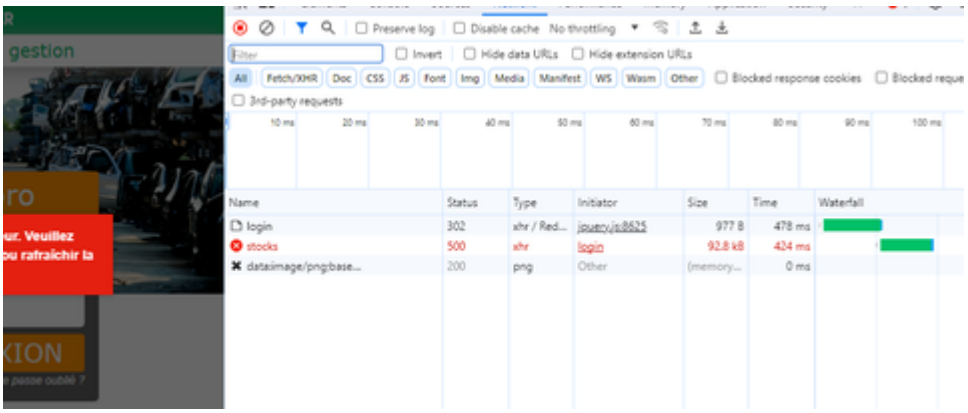
Les utilisateurs ont bien étés mis à jour

- Après avoir réalisé le processus d'authentification sur l'application, il faut le tester en temps réel sur l'application:

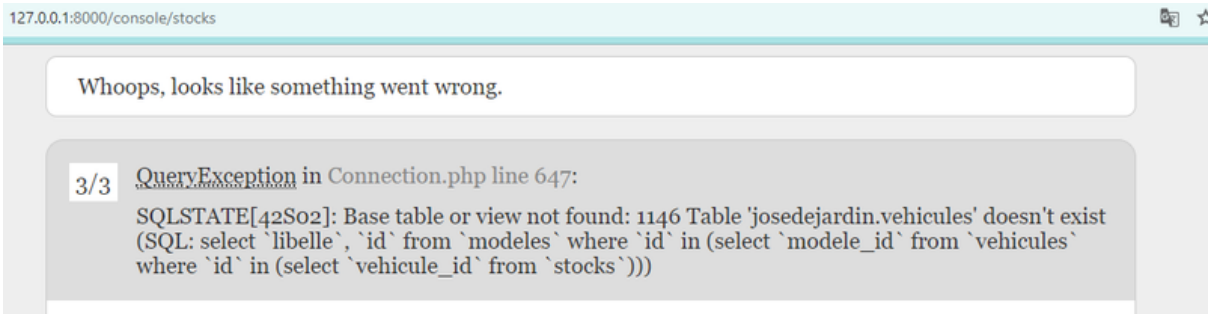
- Tentatives de connexion avec l'utilisateur "webmaster@gmail.com":



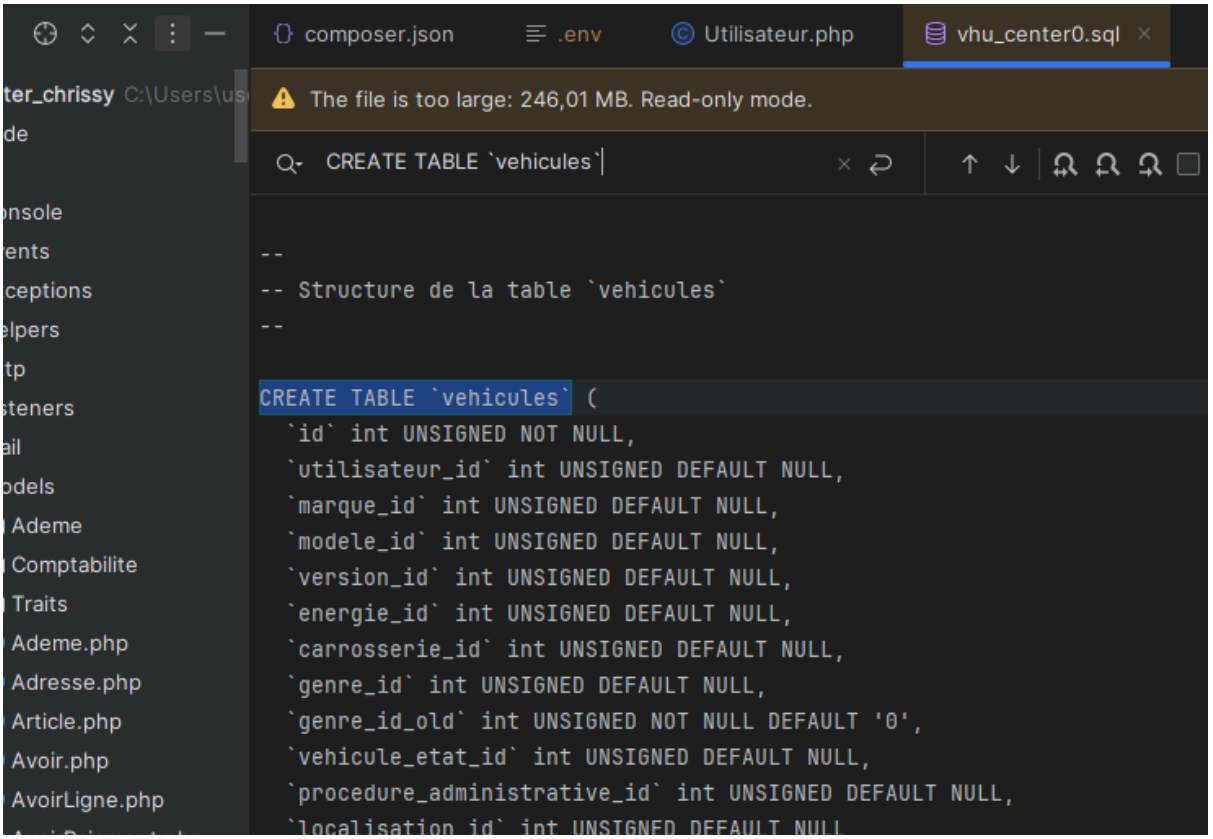
  - après avoir fais des recherches sur les potentiels erreurs possibles:



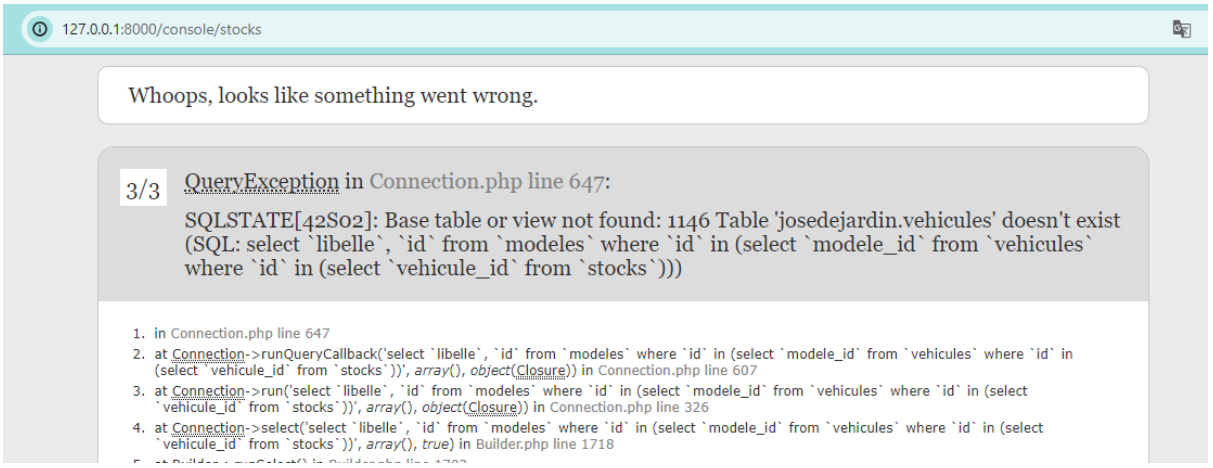
- Consultation des logs pour essayer de trouver d'où pourrait venir le problème



- Il semblerait que la table "vehicules" n'existe pas, ce qui causerait l'erreur.
- Un meeting avec le développeur fut alors organisé dans le but d'échanger sur le sujet



- Après consultation du script, la conclusion fut donc que l'erreur ne vient pas du script SQL mais de l'import de la base de données car elle est trop volumineuse.

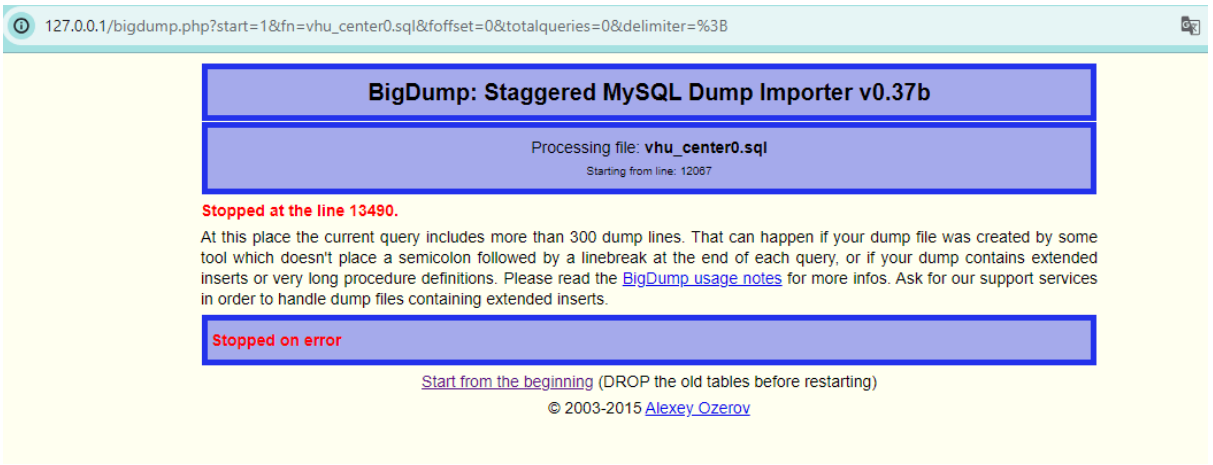


- La solution qui vient fut alors l'utilisation de "Bigdump.php":

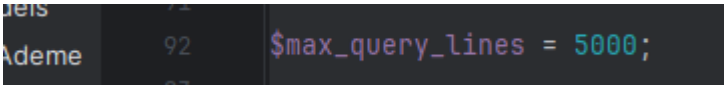
bigdump.php est un script PHP open-source conçu pour importer des fichiers de sauvegarde MySQL de grande taille, souvent appelés "dump" MySQL, dans une base de données. Les fichiers de sauvegarde MySQL peuvent être volumineux en raison du nombre de données qu'ils contiennent, et les importer directement via un client MySQL standard peut parfois être inefficace ou entraîner des problèmes liés à la taille du fichier.

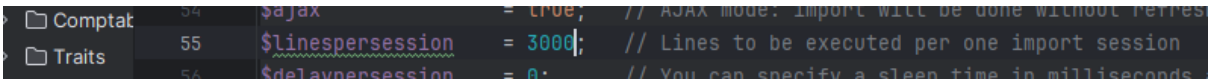
Le script BigDump a été conçu pour résoudre ce problème en permettant d'importer des fichiers de sauvegarde MySQL de manière plus efficace, notamment en fractionnant l'importation en plusieurs parties (d'où le terme "Staggered MySQL Dump Importer") afin de contourner les limitations de taille et de temps d'exécution qui pourraient être rencontrées lors d'une importation en une seule fois.

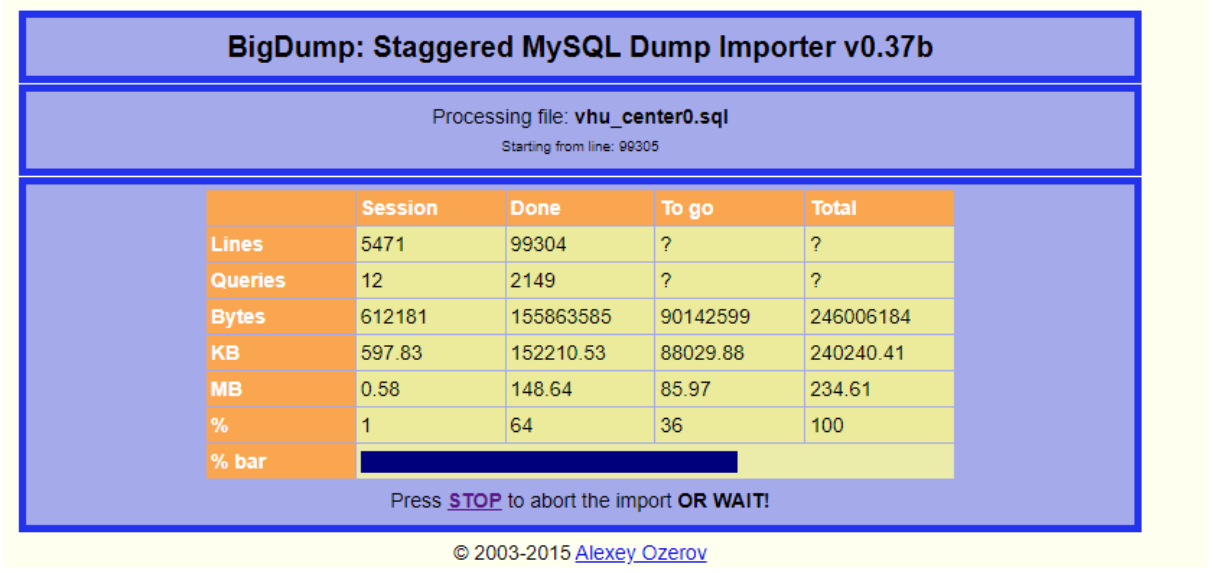
- une erreur est survenue lors de l'import via bigdump:



- Après les recherches pour résoudre le problème, changer le code du script php fut une solution possible en augmentant les limites:

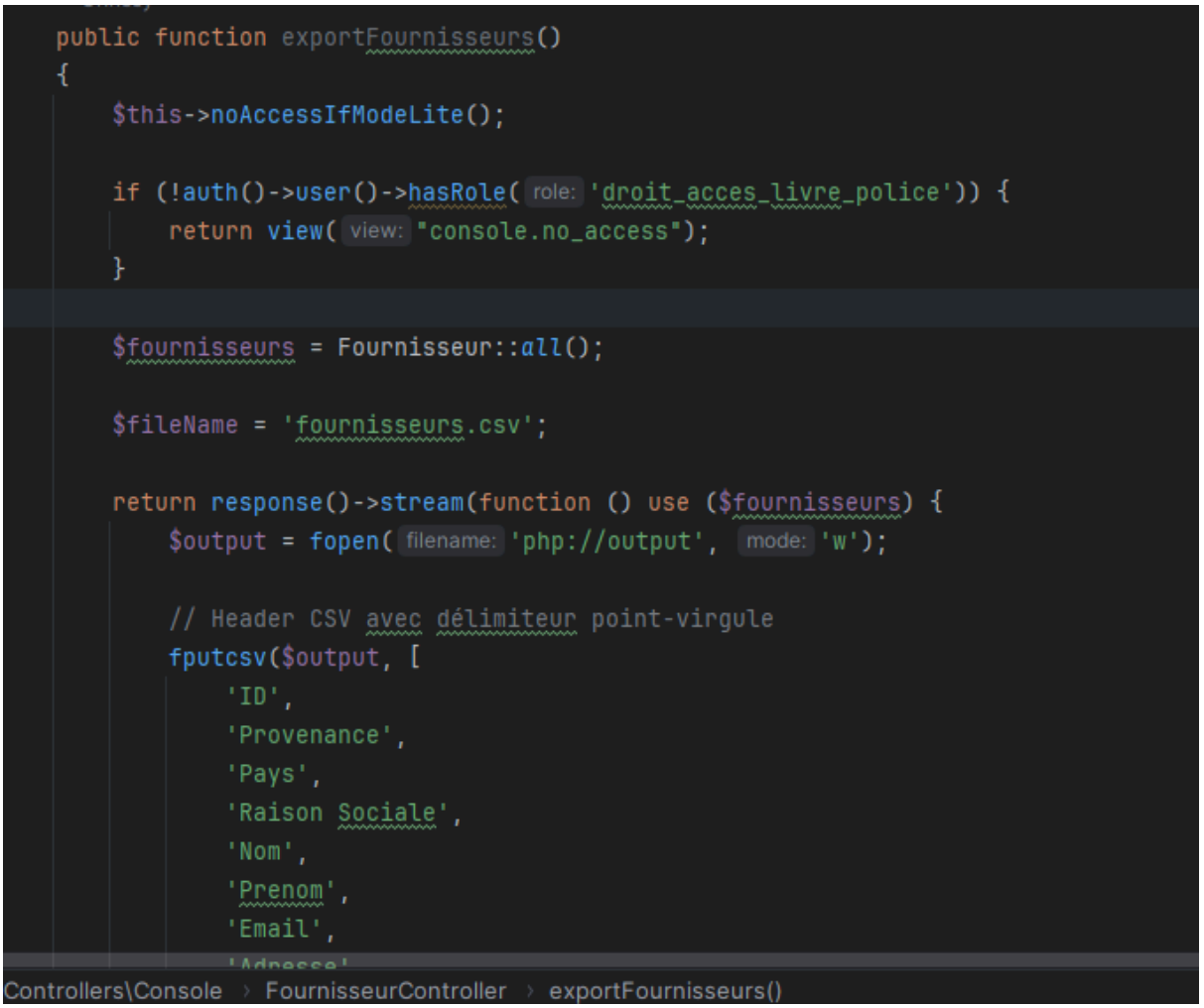






Semaine 6 (et partiellement 7)

Semaine de travail sur une fonctionnalité permettant d'exporter les données de la base de donnée sur pour la table Fournisseur et la table Clients en Laravel PHP.



Pour se faire il a fallu créer une nouvelle fonction "exportFournisseurs" dans le FournisseurController, cette fonction aura pour fonction d'exporter les données et de les ordonner dans un tableau.

```php
public function exportClients()
{
    $this->noAccessIfModeLite();

    if (!auth()->user()->hasRole( role: 'droit_acces_livre_police')) {
        return view( view: "console.no_access");
    }

    $header = [
        'Identifiant',
        'Identifiant de l\'utilisateur',
        'Identifiant du type de client',
        'Numéro de compte',
        'Date de création',
        'Date de mise à jour',
        'Date d\'exportation du relevé',
        'Date d\'envoi du relevé',
        'Type de client',
        // Colonnes pour les données d'adresse
        'Nom',
        'Prénom',
        'Téléphone',
        'Email',
        'Adresse',
        'Code Postal',
```

Controllers\Console › ClientController › exportClients()

La fonction exportClient a pour le même but, mais une contrainte a été rajouté car on a besoin de mettre dans le tableau l'adresse des clients, c'est-à-dire l'adresse de livraison. la relation pour celàn fut la relation client.adresse mais comme il possède 2 adresses, il fallait prendre celui de l'adresse de facturation de livraison.

```php
Route::get( uri: '/notification/ajax-show/{notification}', action: 'NotificationController@ajaxShow')->name( name: 'notification.ajaxShow');

Route::get( uri: '/export-fournisseurs', action: 'Console\FournisseurController@exportFournisseurs')->name( name: 'export.fournisseurs.csv');
Route::get( uri: '/export-clients', action: 'Console\ClientController@exportClients')->name( name: 'export.clients');
```

après avoir crée les fonctions, il a fallu créer les routes qui feront appelle à ces fonctions, ici les routes seront pour la fonction "exportFournisseur": export.fournisseurs.csv

et pour la fonction "ExportClients": export.clients"

```blade
@if( !isset($ajaxMode) || (isset($ajaxMode) && !$ajaxMode ) )
    <div class="btn_ajouter pull-right">
        <a href="{{route('console::client.create')}}" title="Ajouter un client"><span class="ajouter-client">Ajouter un client</span></a>

        <a href="{{ route('export.clients') }}">Exporter les clients</a>

    </div>
```

Après avoir ajouté tout celà, il a fallu ajouter le bouton “exporter” dans les vues blades de l’application avec le nom des routes spécifiées, et dans les vues différentes.

```
@if( !isset($ajaxMode) || (isset($ajaxMode) && !$ajaxMode ) )
    <div class="btn_ajouter pull-right">
        <a href="{{route('console::fournisseur.create')}}" title="Ajouter un fournisseur"><span class="ajouter-client ajouter-fourni
        <a href="{{route('export.fournisseurs.csv')}}"> Exporter</a>


    </div>
```

Plusieurs erreurs se sont accumulées, le contenu de l’export était affiché directement dans la fenêtre du navigateur au lieu d’être téléchargé. Pour celà, il a fallu changer la fonction export pour qu’il retourne efficacement ce que l’on lui a demandé.

```
            fclose($output);
        }, status: 200, [
            'Content-Type' => 'text/csv',
            'Content-Disposition' => 'attachment; filename="' . $fileName . '"',
        ]);
    }
```

Les données sont belles et bien téléchargées mais les entêtes de pages du fichier excel ne sont pas correctements configurées et toutes les données sont désordonnées et mélangées.

```
            'SIRET',
            'Telephone',
            'Date de Création',
            'Date de Modification'
        ], separator: ';'); // Spécification du délimiteur point-virgule

        // Data rows avec délimiteur point-virgule
        foreach ($fournisseurs as $fournisseur) {
```

Pour se faire, il a fallu ajouter un délimiteur pour permettre de séparer les données et les headers du fichier exporté.