

TP GUIDEE LARAVEL

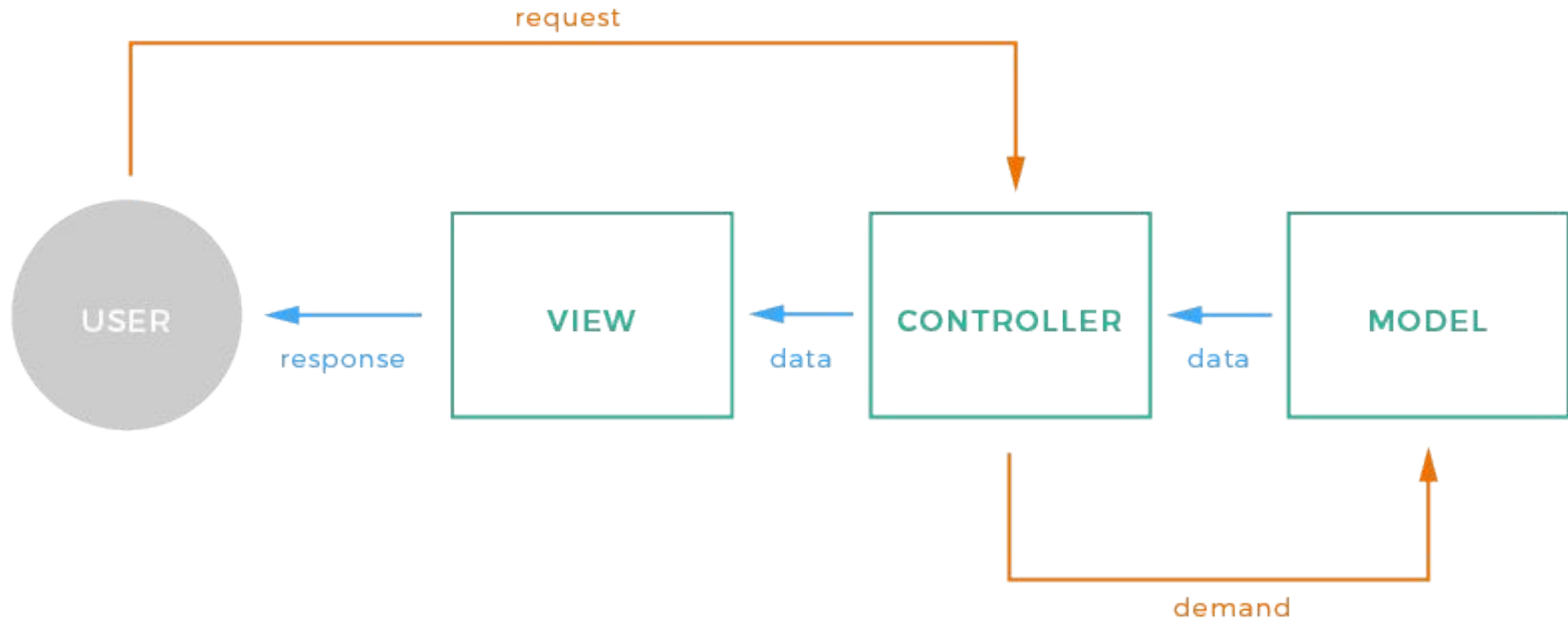
Introduction

Laravel est un Framework PHP libre de droits qui a fait son apparition en 2011. Il est peut-être jeune comparé aux autres de son genre, mais il se démarque par sa facilité, sa syntaxe élégante, et toutes sa documentation disponible à tous. De plus, Laravel utilise la toute dernière version de PHP 5.3 et a fréquemment des patches de disponibles avec de nouveaux éléments et des mises à jour qui règlent les problèmes, ce qui prouve qu'il est en constante évolution et amélioration. En ce moment, il se base sur « Composer », le meilleur outil de dépendance qui gère des projets en PHP jusqu'à maintenant.

MVC

Laravel se base effectivement sur le patron de conception MVC, c'est-à-dire modèle-vue-contrôleur. Le modèle interagit avec la base de données, les regroupe, traite et gère les données. La vue s'occupe principalement de faire afficher ce que le modèle renvoie. Ensuite, elle s'occupe de recevoir toute interaction de l'utilisateur (hover, clic de souris, entrée de texte, etc.). Ce sont ces actions-là que le contrôleur gère. Celui-ci prend en charge de synchroniser le modèle et la vue. Il capte toutes les activités de l'utilisateur et, en fonction de, il actionne les changements à effectuer sur le site. En séparant les composants d'un site internet en ces trois catégories, cela permet une clarté de l'architecture des dossiers et simplifie grandement la tâche aux développeurs.

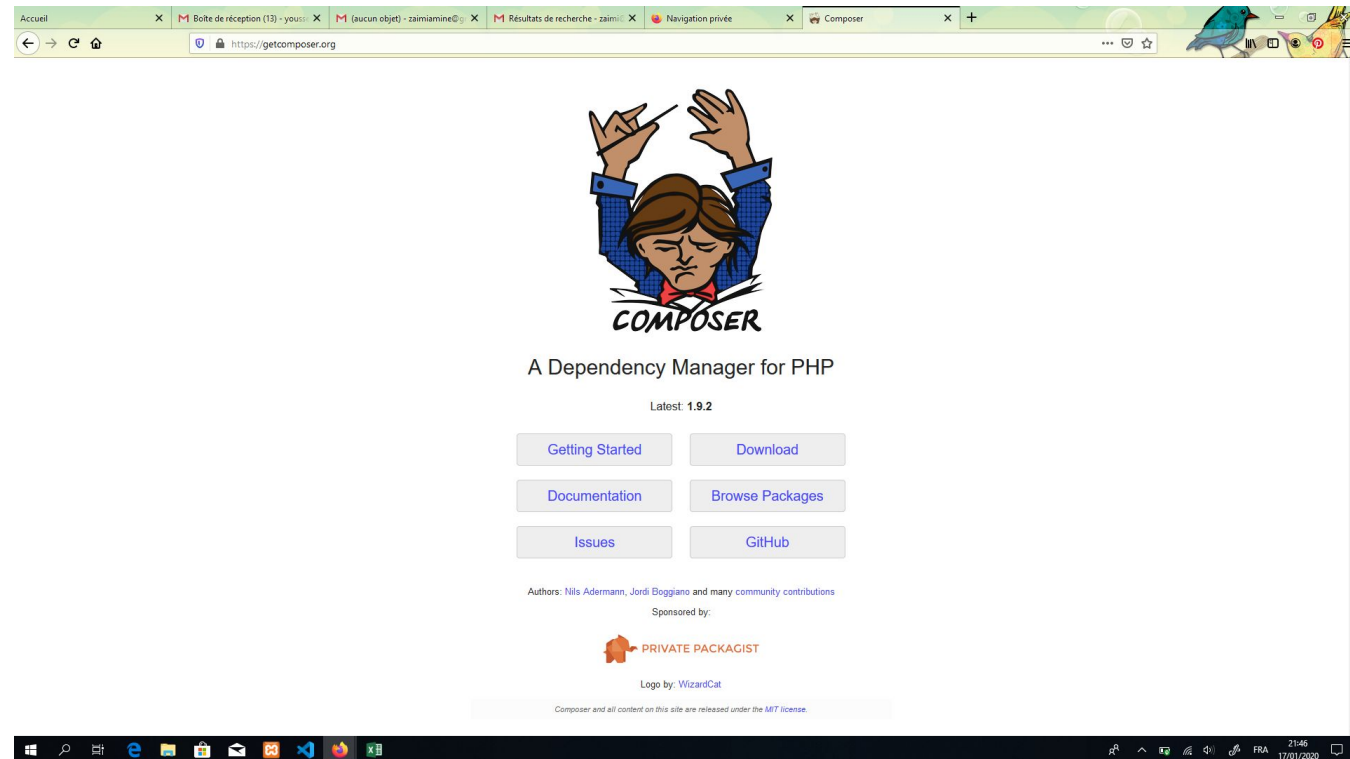
MVC



Installation

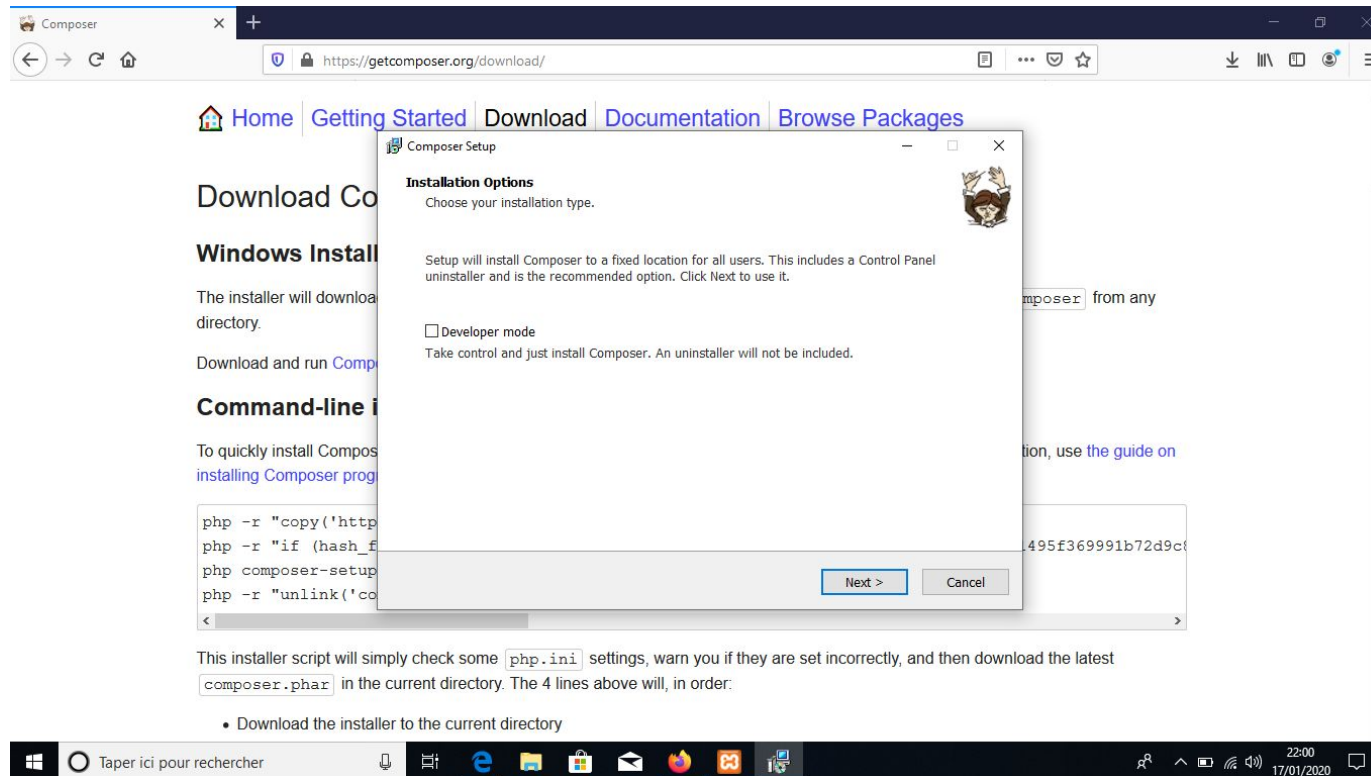
Composer

La première étape serait d'installer le Composer dont laravel utilise. Cependant, qu'est ce qu'un composer? Et bien c'est assez simple, Composer trouve les fichiers PHP qu'on a besoin dans un projet. Il va les chercher et les installer à la bonne place, pour nous. Comme Laravel est un Framework PHP, il est très pratique de l'utiliser pour bien partir un projet.



Installation de Composer

On peut le télécharger sur le site getcomposer.org ou directement sur le site de laravel.com. Ce Composer a une entente avec Laravel et va chercher tous les fichiers que Laravel utilise pour bien fonctionner. En installant le Composer, on pourra installer beaucoup plus facilement et rapidement le Framework Laravel, sans faire d'erreur. Une fois le fichier « Composer-Setup.exe » est installé dans l'ordinateur, on peut maintenant installer Laravel.



Installation de Laravel

Maintenant, il faut savoir où installer le projet. Je vous conseille de travailler en localhost, avant de travailler sur le serveur lui-même, pour simple efficacité. Ouvrez une fenêtre Windows de où vous voulez mettre le dossier de projet et faites click-droit, si vous avez bien installé le Composer, vous devriez voir dans la liste « use Composer here ». Vous le sélectionnez et une fenêtre de commande va apparaître. Vous devez inscrire la ligne de commande cidessous :

```
composer create-project laravel/laravel nom-votre-projet --prefer-dist
```

Composer : signifie qu'on utilise le Composer

create-project : signifie qu'on crée un projet

laravel/laravel : va chercher les fichiers à installer pour le Framework Laravel

Nom-votre-projet : on met le nom qu'on veut donner à notre projet

--prefer-dist : il y a plusieurs sous branches de Laravel, et cette commande-là va chercher la version complète de Laravel.

Exemple : composer create-project laravel/laravel INSCRIPTION --prefer-dist

**Composer global require laravel/installer
laravel new inscription**

Une fois l'installation complétée, testez si ça a bel et bien fonctionné. Pour ce faire, ouvrez votre fenêtre de navigation, google chrome ou firefox, et entrez votre lien pour le site soit :

localhost/nom-de-votre-dossier/public

OU

localhost:8000/

php artisan serv

Laravel

[DOCS](#) [LARACASTS](#) [NEWS](#) [BLOG](#) [NOVA](#) [FORGE](#) [VAPOR](#) [GITHUB](#)

Structure des dossiers

Il est important de d'abord analyser la structure des dossiers pour savoir comment la hiérarchie fonctionne. Voici les dossiers importants à retenir :

Projet	app	config
		controllers
		models
		views
		routes.php
	public	css
		img
		js
		less

Il y a d'autres dossiers mais moins importants quand on commence à utiliser Laravel. Le dossier `app` contient tous les éléments nécessaires à la programmation « back-end » du site. Le dossier `public`, lui, contient les médias et les autres langages de programmation, soit le CSS, le JS et il y a même un dossier `less` pour ceux qui veulent s'aventurer avec ce langage, ce que je conseille très fortement, `less` est très facile à utiliser et simplifie beaucoup l'apparence du `css` ! Il y a également un dossier pour `bootstrap` si jamais on veut l'utiliser pour le site internet. On peut voir ici que Laravel utilise les dernières nouveautés dans la programmation pour apporter la meilleure expérience aux utilisateurs de ce Framework.

Base de données

Maintenant, il faut relier la base de données au projet. Dans le dossier racine de votre projet, il existe le fichier **.env**. En l'ouvrant, on peut voir plusieurs array, dont un qui est associé à mysql.

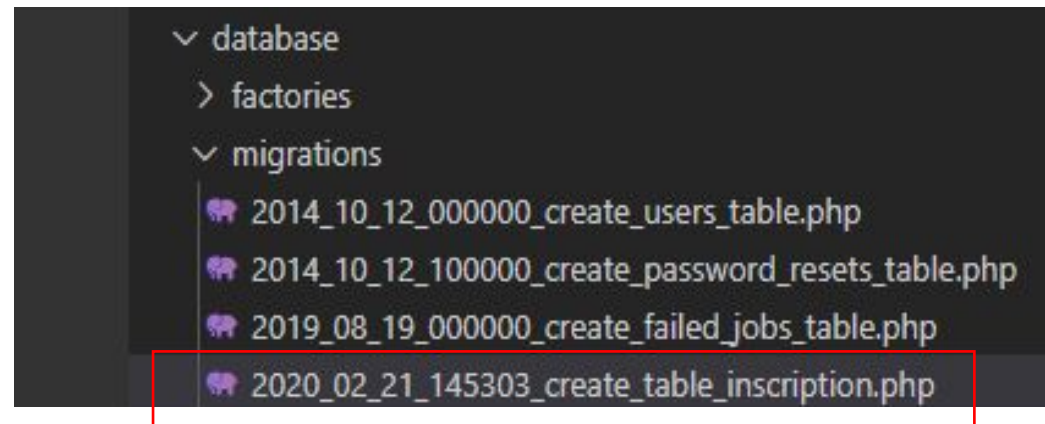
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=inscription
DB_USERNAME=root
DB_PASSWORD=
```

Créer une table (Base de données)

Si vous utilisiez MySQL dans vos applications PHP, vous avez sans doute déjà utilisé PHPMyAdmin pour visualiser et créer vos tables dans votre base de données. Le principal inconvénient de cette approche est pour le partage de la structure de la base de données. Si un collègue a besoin de tester l'application vous devez lui envoyer un export de la base ou si vous devez mettre en ligne l'application. Même chose lors de la mise en ligne de l'application. Laravel vient avec une solution à ce problème : les migrations.

php artisan make:migration create_inscription

une commande pour créer une migration pré-remplie



Créer une table (Base de données)

Contenu du fichier de la table inscription

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateInscription extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('inscriptions', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('inscriptions');
    }
}
```

Créer une table (Base de données)

Il faut compléter le script pour la création de la table inscription

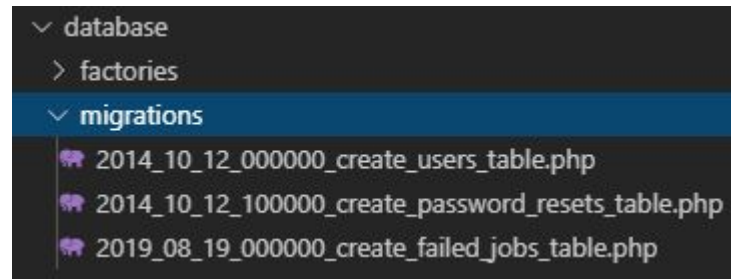
```
public function up()
{
    Schema::create('inscriptions', function (Blueprint $table)
    {
        $table->Increments('id_ind');
        $table->string('nom');
        $table->string('prenom');
        $table->date('datenais');
        $table->string('villenais');
    });
}
```

La commande pour exécuter la création de la table inscription

php artisan migrate

Par défaut la partie de l'authentification (c'est à dire la manière de retrouver les renseignements des utilisateurs) avec Laravel se fait en base de données avec Eloquent et part du principe qu'il y a un modèle **App\User** (dans le dossier **app**).

Liste des fichiers de création des tables d'authentification



Commande d'exécution des script de création des tables d'authentification :

php artisan migrate

```
PS C:\xampp\htdocs\INSCRIPTION> php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (0.08 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (0.07 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (0.04 seconds)
PS C:\xampp\htdocs\INSCRIPTION>
```

Sur le phpmyadmin

Contenant le mot :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> failed_jobs	★	0	InnoDB	utf8_general_ci	16 kio	-
<input type="checkbox"/> migrations	★	3	InnoDB	utf8_general_ci	16 kio	-
<input type="checkbox"/> password_resets	★	0	InnoDB	utf8_general_ci	16 kio	-
<input type="checkbox"/> users	★	0	InnoDB	utf8_general_ci	16 kio	-
4 tables	Somme	3	InnoDB	latin1_swedish_ci	64 kio	0 o

Les routes de l’authentification

Dans l’installation de base vous ne trouvez aucune route pour l’authentification. Pour les créer (et ça ne créera pas seulement les routes) il faut déjà installer un package :

```
composer require laravel/ui --dev
```

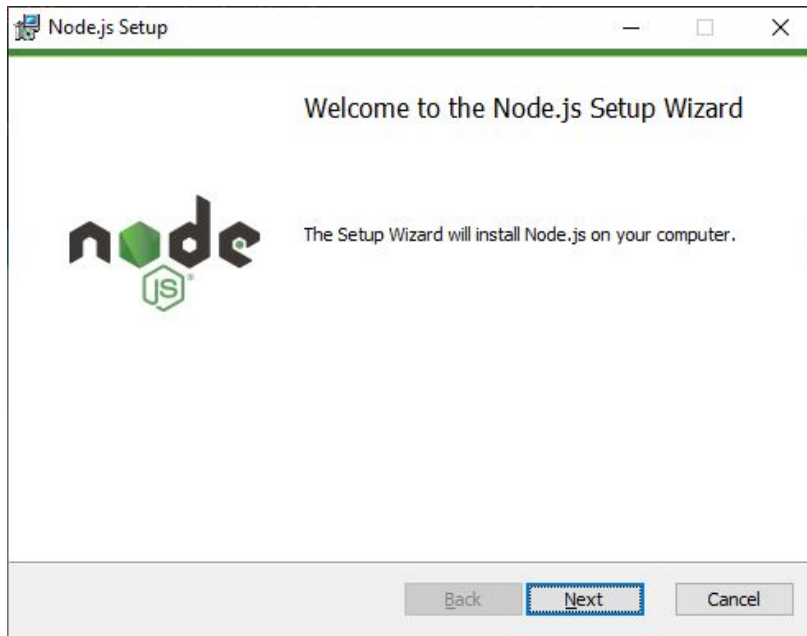

Suite

On voit qu'on dispose de 3 preset : bootstrap, vue et react. On a donc le choix !

On voit aussi qu'on a l'option **–auth** pour générer le scaffolding.

Le choix n'est pas encore vraiment important pour le moment et on va prendre par exemple **vue**

```
php artisan ui vue --auth
```

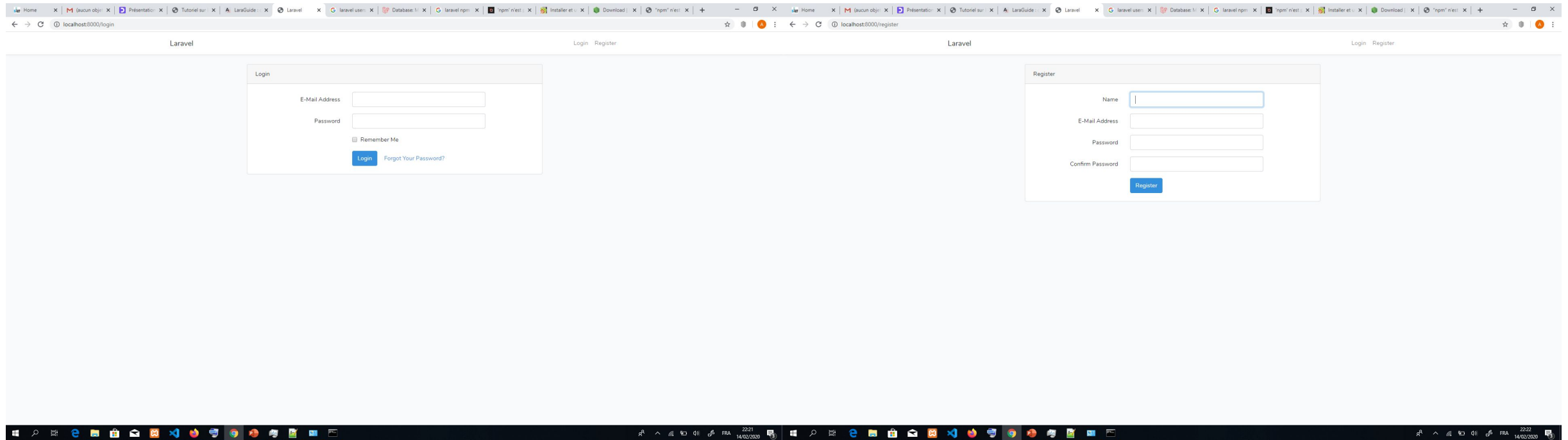


```
npm install  
npm run dev
```

Sur web.php le système ajoute les chemins suivants

```
Auth::routes();
```

```
Route::get('/', 'HomeController@index')->name('home');
```



Création d'un modèle

Si vous utilisiez simplement PHP pour votre application, vous devriez avoir à écrire vos requêtes SQL à la main. Ce n'est pas le cas avec Laravel qui fournit un outil, Eloquent, qui permet, dans la majorité des cas, d'utiliser de simples fonctions PHP à la place de requêtes SQL.

Eloquent fonctionne avec un système de modèle. Un modèle est une simple classe PHP qui représente une table de notre application. Nous allons créer le fichier inscription.php de cette classe dans le dossier app qui contiendra toutes nos classes PHP

```
php artisan make:model inscription
```

```
<?php  
  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
  
class inscription extends Model  
{  
    //  
}
```

Création d'un blade(Vue)

Afin d'afficher le formulaire HTML, nous avons besoin de créer une nouvelle route avec une nouvelle vue.

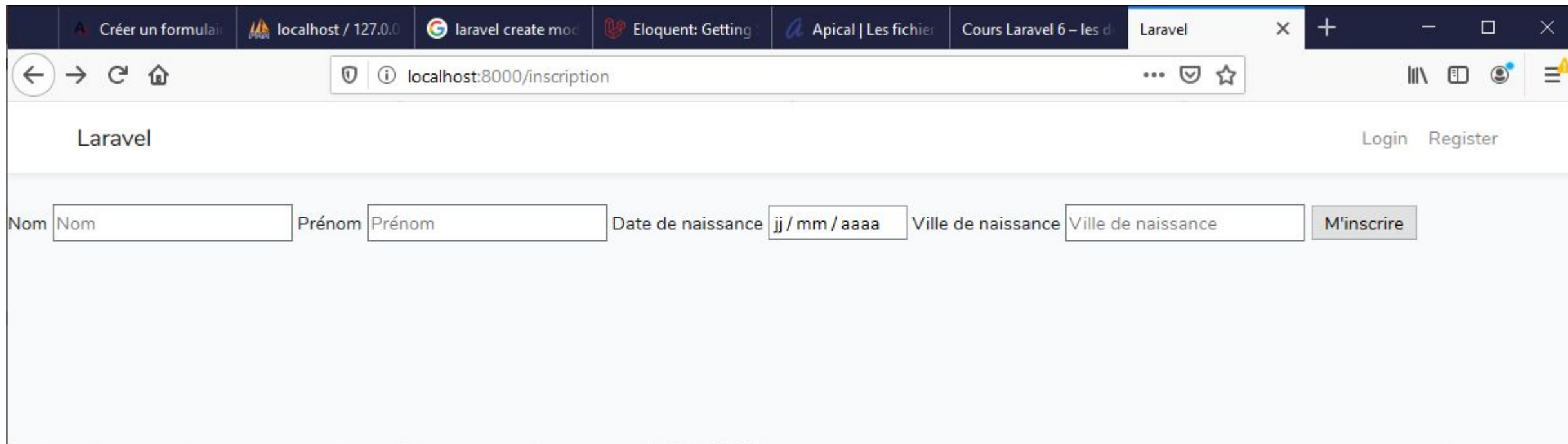
Dans l'arborescence de votre projet, cliquer avec le bouton droit sur views puis sur new file (inscription.blade.php)

```
@extends('layouts.app')

@section('content')
    <form action="/inscription" method="post">
        <label for="nom">Nom</label>
        <input type="text" name="nom" placeholder="Nom">
        <label for="prenom">Prénom</label>
        <input type="text" name="prenom" placeholder="Prénom">
        <label for="datenais">Date de naissance</label>
        <input type="date" name="datenais" placeholder="Date de naissance">
        <label for="villenaais">Ville de naissance</label>
        <input type="text" name="villenaais" placeholder="Ville de naissance">
        <input type="submit" value="M'inscrire">
    </form>
@endsection
```

Création d'un blade(Vue)

Résultat :



The screenshot shows a web browser window with the address bar displaying `localhost:8000/inscription`. The page title is "Laravel". In the top right corner, there are links for "Login" and "Register". The main content area contains a registration form with the following fields and labels:

- Nom**: A text input field with the placeholder text "Nom".
- Prénom**: A text input field with the placeholder text "Prénom".
- Date de naissance**: A text input field with a date mask `jj / mm / aaaa`.
- Ville de naissance**: A text input field with the placeholder text "Ville de naissance".
- M'inscrire**: A button to submit the registration form.

Création d'un blade(Vue)

Routage : Dans le fichier web.php dans le dossier routes ajouter le code suivant

```
Auth::routes();

Route::get('/', 'HomeController@index')->name('home');

Route::get('/inscription', function () {
    return view('/inscription');
});
```

Création d'un Contrôleur

Les contrôleurs sont le reste de notre application PHP, ils vont se charger de faire le lien entre nos modèles et nos vues.

La commande pour la création d'un contrôleur :

```
php artisan make:controller InscriptionController
```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class InscriptionController extends Controller
{
    //
}
```

Création d'un Contrôleur

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\inscription;

use Auth;
use DB;

class InscriptionController extends Controller
{
    public function enginscription(Request $request){

if (Auth::check()) {
    $inscriptions = inscription::where('id_ind',Auth::user()->id)->get();
    if($inscriptions->count()>0){
        $inscriptions = new inscription();
        DB::table('inscriptions')
            ->where('id_ind', Auth::user()->id)
            ->update(['nom' =>request('nom')

```


Modification du blade

```
@extends('layouts.app')

@section('content')
@if($inscriptions->count())
@foreach($inscriptions as $inscription)
<form method="POST" action="{{ url('inscription') }}" enctype="multipart/form-data" autocomplete="off">
    <input type="hidden" name="_token" value="{{ csrf_token() }}">
        <label for="nom">Nom</label>
        <input type="text" name="nom" placeholder="Nom" value="{{ $inscription->nom }}" >
        <label for="prenom">Prénom</label>
        <input type="text" name="prenom" placeholder="Prénom" value="{{ $inscription->prenom }}">
        <label for="datenaiss">Date de naissance</label>
        <input type="date" name="datenaiss" placeholder="Date de naissance" value="{{ $inscription->datenaiss }}">
        <label for="villenaiss">Ville de naissance</label>
        <input type="text" name="villenaiss" placeholder="Ville de naissance" value="{{ $inscription->villenaiss }}">
        <input type="submit" value="M'inscrire">
    </form>
</div>
@endforeach
@endif
@endsection
```