Laravel

Projet table et CRUD.

Auteur : ***Berthe Seydou***

10/08/2016

Table des matières

[CRUD sur table: 2](#_Toc459034978)

[1. Initialisation des tables (Categories, Metteur en Scène « Mes » & Acteur) : 2](#_Toc459034979)

[2. Implémentation de fonction du Controller « FilmController » : 5](#_Toc459034980)

[3. Création des vues : 8](#_Toc459034981)

Prérequis : Projet Laravel Table Relation.

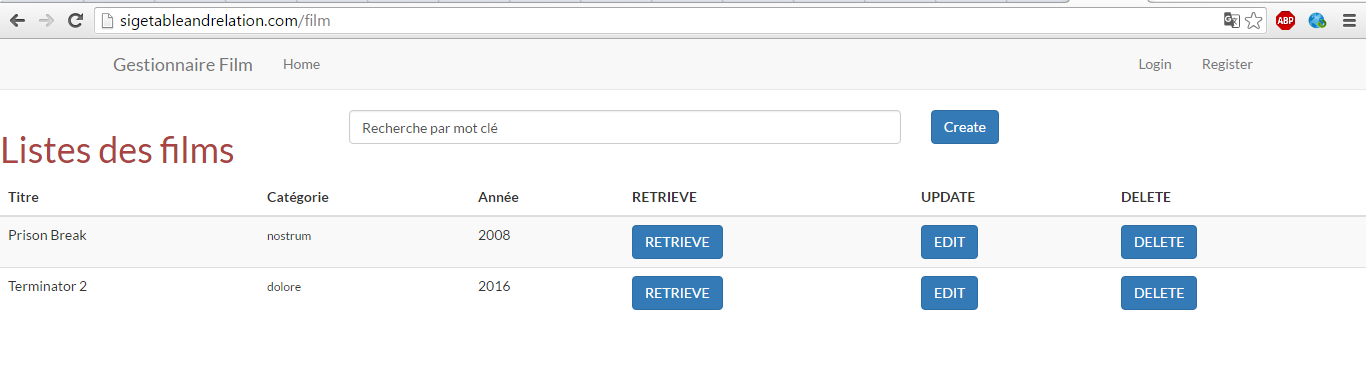
# CRUD sur table:

Dans ce projet, il sera question de mettre en œuvre un exemple d’application effectuant des opérations de création, récupération, mise à jour et suppression (**CRUD « Create, Retrieve, Update and Delete »** en Anglais) sur une ressource.

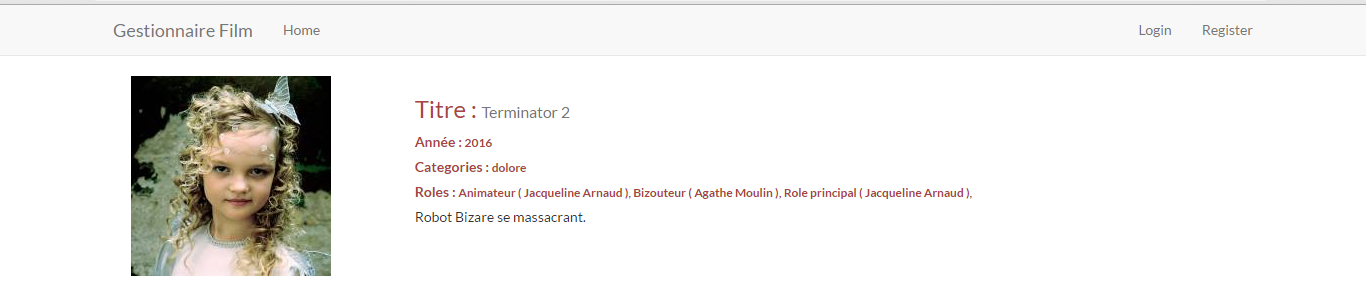
En effet, ce projet consiste pour un utilisateur de s’authentifier et d’ajouter, lister, modifier et supprimer des ressources « **Films** ».

Ci-dessous représente les captures des différentes pages que l’on aura à la fin du projet :

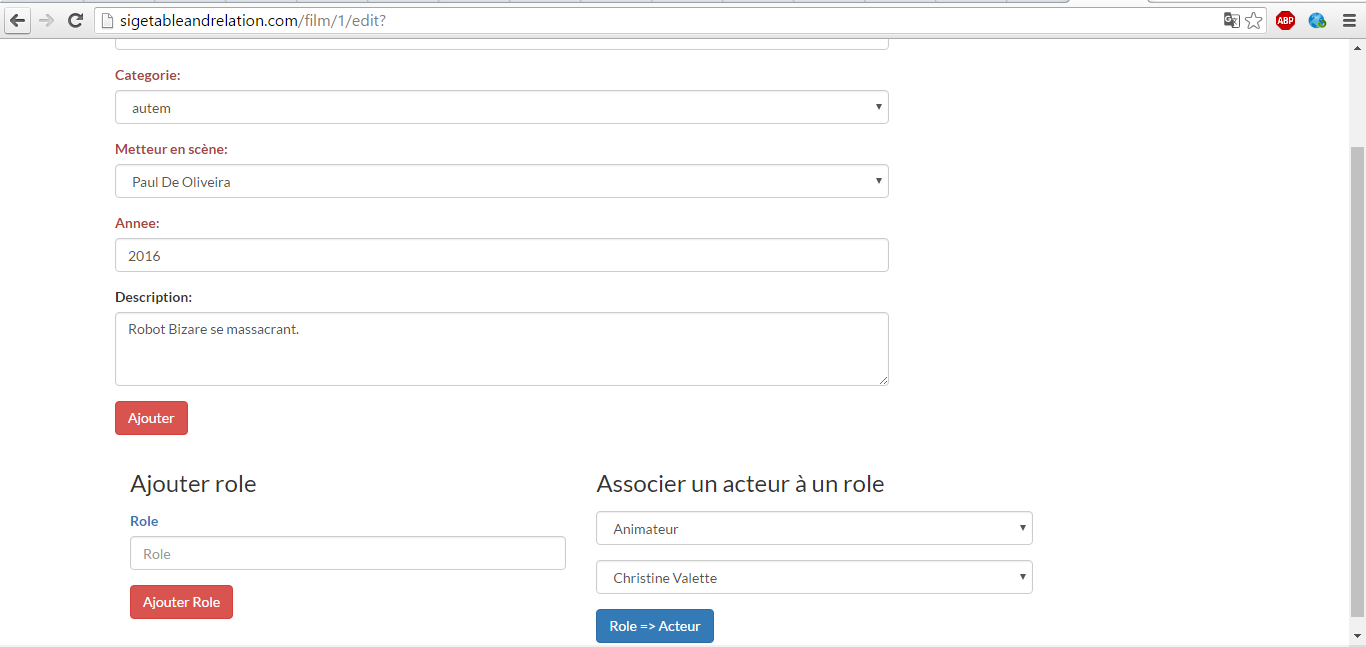
**🡪La page d’accueil :**



**🡪Page d’affichage d’un film :**

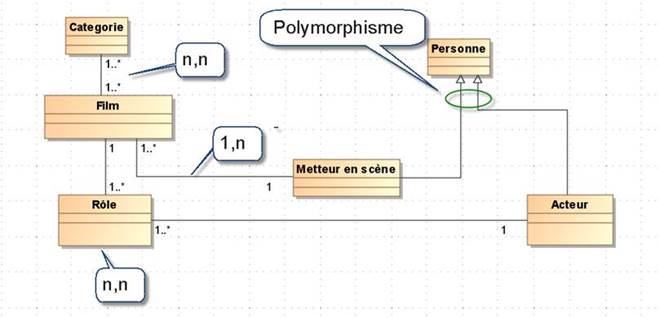


**🡪Page d’édition d’un film :**



Pour ce faire, nous allons partir d’un projet qui contient déjà un squelette des composants nécessaires à la mise en place des opérations de **CRUD**.

Alors commencer tout d’abord par télécharger ce projet sur GIT à partir du lien suivant : [TableAndRelation](https://github.com/ArianeExtern/EtudeLaravel/tree/TableRelation). Après téléchargement, vous remarquerez que ce projet contient déjà tous ce qui est **Controller**, **Model**, **Migration** et **Vues** fait à partir du MCD (Modèle Conceptuel de Donnée) ci-dessous :

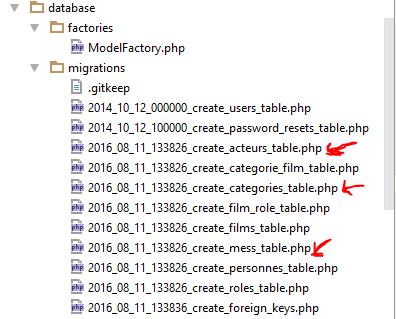


A la suite de la mise en place du projet « **TableAndRelation** », il faudra suivre les étapes suivantes pour compléter ce projet :

1. Charger les tables (**Catégories, Metteur en Scène & Acteur** par des informations en guise de test).
2. Implémenter les fonctions (**CRUD => « Create, Retrieve, Update and Delete**) des **Controller** pour assurer l’interaction avec les tables concernées.
3. Compléter les vues et Associer les formulaires (**resources/views/pages**) de saisies aux routes. (Les formulaires en questions ne sont que des squelettes).

# Initialisation des tables (Categories, Metteur en Scène « Mes » & Acteur) :

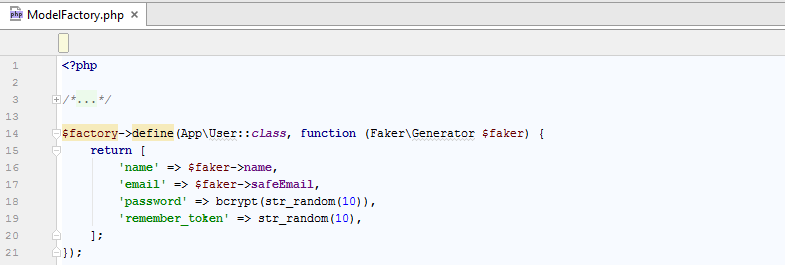
Pour initialiser ces tables, nous allons commencer par voir les migrations du projet :



Comme on peut le voir, il existe plusieurs Migration dans le projet mais nous nous intéressons à ceux qui ont la flèche rouge devant à savoir celui de « **Categories, Mess, et Acteurs**».

Pour initialiser ces tables, **Laravel** propose une solution assez simple à travers le « **ModelFactory** », comme le nom l’indique, il s’agit d’une fabrique de **Model** c’est-à-dire qu’il va nous permettre de fabriquer autant d’instance d’un **Model** que l’on veut et éventuellement le persister dans la table associer au **Model**.

Pour ce faire, jetons un coup d’œil sur le fichier « **database/factories/ModelFactory.php** » :



Le code ci-dessous permet de créer une fabrique pour un utilisateur « **User** » (Générer par défaut dans un projet **Laravel**), la variable « **$faker** » permet de générer des informations aléatoire en fonction de la propriété indexer, par exemple sur l’image :

|  |  |
| --- | --- |
| **Expression** | **Sens** |
| **$faker->name** | génère un **nom** aléatoire |
| **$faker->safeEmail** | génère une **adresse mail** aléatoire |

C’est ainsi que pour chaque champs du Model « **User** », on génère des informations aléatoires.

Dans notre cas, nous voulons générer des informations aléatoires des tables « **Catégorie**, **Metteur en scène (Mes) et Acteurs** ». Ainsi nous allons ajouter les lignes suivantes dans le même fichier « **databases/fatories/ModelFactory.php** » :

*//Frabrique pour le model Categorie*$factory->define(App\Model\Categorie::***class***, **function** (Faker\Generator $faker) {  
 $faker = Faker\Factory::*create*(**'fr\_FR'**); *//Element générer en francais* **return** [  
 **'nom'** => $faker->**word**,  
 ];  
});  
  
*//Fabrique pour le Model Personne*$factory->define(App\Model\Personne::***class***, **function** (Faker\Generator $faker) {  
 $faker = Faker\Factory::*create*(**'fr\_FR'**); *//Element générer en francais* **return** [  
 **'nom'** => $faker->**name**,  
 **'age'** => $faker->**year**,  
 **'sexe'** => ($faker->**boolean** ? **'Male'** : **'Femelle'**),  
 ];  
});  
  
*//Fabrique pour le Model Metteur en Scène*$factory->define(App\Model\Mes::***class***, **function** (Faker\Generator $faker) {  
 $faker = Faker\Factory::*create*(**'fr\_FR'**); *//Element générer en francais* **return** [  
 **'trophe'** => $faker->**boolean**,  
 ];  
});  
  
*//Fabrique pour le Model Acteur*$factory->define(App\Model\Acteur::***class***, **function** (Faker\Generator $faker) {  
 $faker = Faker\Factory::*create*(**'fr\_FR'**); *//Element générer en francais* **return** [  
 **'star'** => $faker->**words**,  
 ];  
});

A ce niveau on a créé les fabriques qui permettront d’initialiser nos tables. On va maintenant faire appel à nos fabriques dans le **Seeder** « **database/seeds/DatabaseSeeder.php**», on ajoute le code ci-dessous dans la méthode « **run()** » du Seeder :

**public function** run()  
{  
 *//Creation et insertion dans la base 20 categories aléatoires.* factory(Categorie::***class***, 20)->create();  
  
 *//Creation de 100 Personnes* $personnes = factory(Personne::***class***, 100)->make()->toArray();  
  
 *//Creation et Sauvegarde de 50 Acteur & 50 Metteur en scène* factory(Acteur::***class***, 50)->create();  
 factory(Mes::***class***, 50)->create();  
  
 $j = 0;  
 $k = 50;  
 **foreach** (Acteur::*get*() **as** $acteur){  
 $tmp = **new** Personne($personnes[$j]);  
 *//Association de l'Acteur à une personne et sauvegarde* $acteur->personne()->save($tmp);  
 $j++;  
 }  
 **foreach** (Mes::*get*() **as** $mes){  
 $tmp = **new** Personne($personnes[$k]);  
 $mes->personne()->save($tmp);  
 $k++;  
 }  
}

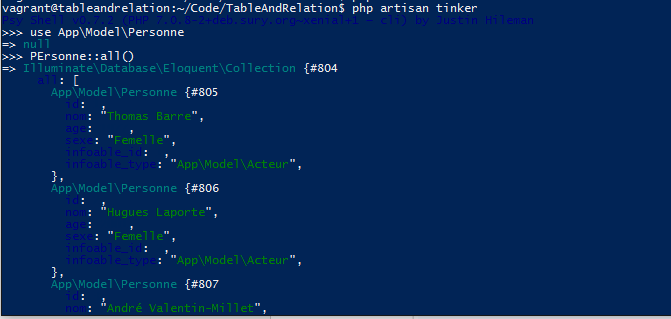
On a ajouté le code nécessaire pour initialiser les tables, on va donc laisser **Laravel** initialise les tables, étant placé dans le répertoire du projet, exécuter les commandes suivantes :

**php artisan migrate** //Pour mettre à jour la base de données avec les nouvelles tables

**NB : S’il y’a un message « Nothing to migrate », il faudra exécuter la commande « composer dump-autoload » pour recharger les fichiers auto-charger (Ceci avant d’exécuter la commande ci-dessous).**

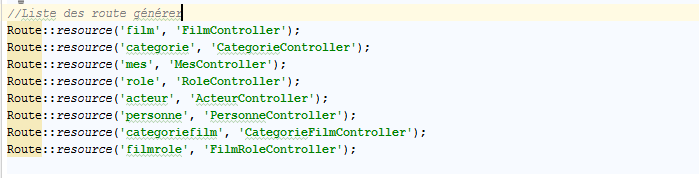
**php artisan db:seed** //Initialise les tables à partir de « DatabaseSeeder.php »

Pour vérifier que les tables ont été bien initialisé, on peut utiliser l’outil **artisan** (**tinker**) avec une option qui permet d’interagir avec notre application en ligne de commande, voir l’image ci-dessous pour la vérification de la table « personnes » :



# Implémentation des fonctions du Controller « FilmController » :

Comme l’application est un gestionnaire de film, on va commencer à effectuer le CRUD sur la table film. Pour ce faire, on va implémenter les méthodes du Controller « **FilmController** » mais avant jetons un coup d’œil sur le fichier **« routes.php** » :



Comme on peut voir la route d’accès aux Films est configuré comme une ressource, Ici on parle de **ressource Film** à laquelle on peut ajouter, récupérer, modifier et supprimer des films. Tous l’intérêt de ce genre de configuration de la route est qu’elle fournit une logique entre les **routes** et les **fonctions du Controller** pour interagir avec la ressource en question, cette logique peut être résumé par le schéma ci-dessous dans le cas de la ressource « **film => Route::resource(‘film’, ‘FilmController’)**»  :

|  |  |  |  |
| --- | --- | --- | --- |
| **Type de requête** | **Route** | **Action (méthode du Controller** | **Nom de la route (utilise au niveau du formulaire)** |
| **GET** | **/film** | **index** | **film.index** |
| **GET** | **/film/create** | **create** | **film.create** |
| **POST** | **/film** | **store** | **film.store** |
| **GET** | **/film/{film}** | **show** | **film.show** |
| **GET** | **/film/{film}/edit** | **edit** | **film.edit** |
| **PUT/PATCH** | **/film/{film}** | **update** | **film.update** |
| **DELETE** | **/film/{film}** | **destroy** | **film.destroy** |

Ce que veut dire par exemple la 1ière ligne est que : **Pour une requête de type « GET » vers la route « /film », la méthode du Controller « FilmController » à exécuter est « index() » et cette route est connu sous le nom de « film.index » si on veut le spécifier dans le formulaire (On verra ça par la suite).**

Donc l’implémentation des méthodes du Controller « **FilmController** » sera comme suivante :

**class** FilmController **extends** Controller {  
  
 */\*\*  
 \* Display a listing of the resource.  
 \*  
 \** ***@return*** *Response  
 \*/* **public function** index()  
 {  
 *//On recupere la liste des films selon le titre ascendant  
 //On fait la pagination de 10 films/page* $films = \App\Model\Film::*orderBy*(**'titre'**, **'asc'**)->paginate(10);  
  
 **return** view(**'pages.index'**, [**'films'** => $films]);  
 }  
  
 */\*\*  
 \* Show the form for creating a new resource.  
 \*  
 \** ***@return*** *Response  
 \*/* **public function** create()  
 {

*//On recupere toutes les categories*$categories = \App\Model\Categorie::*orderBy*(**'nom'**, **'asc'**)->get();  
  
*//On recupère le metteur en scène avec les personne asscoier*$mess = \App\Model\Mes::*with*(**'personne'**)->get();  
  
**return** view(**'pages.films'**, [**'categories'** => $categories, **'mess'** => $mess]);

}

*/\*\*  
 \* Store a newly created resource in storage.  
 \*  
 \** ***@return*** *Response  
 \*/* **public function** store()  
 {  
$params = request()->all();  
  
 *//On recupere le metteur en scene* $mes = \App\Model\Personne::*find*($params[**'mes\_id'**]);  
 $mes = $mes->**infoable**;  
  
  
 *//Création d'un nouveau film* $film = \App\Model\Film::*create*($params);  
  
*//Association du film au metteur en scene*$film->mes()->associate($mes)->save();  
  
*//Ajout du film à une categorie*$film->categories()->attach($params[**'cat\_id'**]);  
  
 **return** redirect(**'/film'**);

}  
  
 */\*\*  
 \* Display the specified resource.  
 \*  
 \** ***@param*** *int $id  
 \** ***@return*** *Response  
 \*/* **public function** show($id)  
 {  
 *//On recupere le film dont l'id est passé en paramètre* $film = \App\Model\Film::*with*(**'categories'**)->find($id);  
  
 *//Liste des catégories associer au film* $categories = $film->categories()->get();  
  
 **return** view(**'pages.show\_film'**, [**'film'** => $film, **'categories'** => $categories]);

}  
  
 */\*\*  
 \* Show the form for editing the specified resource.  
 \*  
 \** ***@param*** *int $id  
 \** ***@return*** *Response  
 \*/* **public function** edit($id)  
 {  
 *//On recupère l'instance de film* $film = \App\Model\Film::*find*($id);  
  
 *//On recupere toutes les categories aussi* $categories = \App\Model\Categorie::*orderBy*(**'nom'**, **'asc'**);  
  
 **return** view(**'pages.films'**, [**'film'** => $film, **'categories'** => $categories]);  
 }  
  
 */\*\*  
 \* Update the specified resource in storage.  
 \*  
 \** ***@param*** *int $id  
 \** ***@return*** *Response  
 \*/* **public function** update($id)  
 {  
 $film = \App\Model\Film::*find*($id);  
$film->update(request()->all());  
**return** redirect(**'/film'**);

**return** redirect(**'/film'**);  
 }  
  
 */\*\*  
 \* Remove the specified resource from storage.  
 \*  
 \** ***@param*** *int $id  
 \** ***@return*** *Response  
 \*/* **public function** destroy($id)  
 {  
 \App\Model\Film::*find*($id)->delete();  
 **return** redirect(**'/film'**);  
 }  
   
}

A ce niveau le Controller de Film est prêt à être utiliser, nous allons définir les vues qui ont été retourné depuis le Controller. On retrouve les vues suivantes :

🡪**pages.index** dans la méthode **index()** : qui représente la page qui va afficher la liste des Films. Il faut remarquer qu’un paramètre a été passé à cette page à savoir : « **$films** : pour afficher la liste de films».

🡪**pages.films** dans la méthode **create()** : qui est le formulaire de création d’un film. (Cette page existe déjà dans le projet donc on va la compléter). Un paramètre contenant les catégories de films sont retourné à cette vue.

🡪**pages.show\_film** dans la méthode **show()** : qui une page qui va afficher les informations relative à un films. Un paramètre représentant le film à afficher est passer à cette vue.

Alors passons maintenant à la création des pages.

# Création des vues :

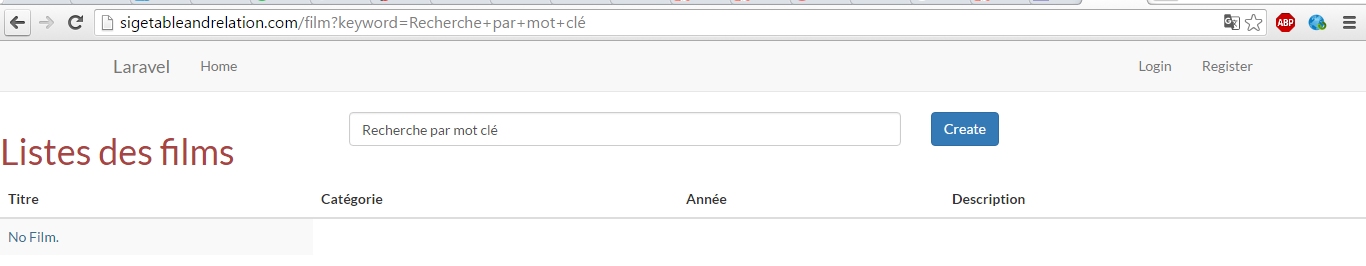
Comme analyser ci-haut, nous allons entamer la création des vues ; Pour ce faire nous allons utiliser un composant **Laravel** nommé « **Collective** **Form»** pour la création de formulaire. Pour l’installation et l’utilisation de ce formulaire, référer vous à ce lien : [FORM DEPENDANCE.](https://laravelcollective.com/docs/5.0/html#installation)

Après l’installation de ce composant commençons à présent la création des vues :

🡪**index.blade.php :** Cette vue est celle qui listera la liste des films, commencer par créer un fichier « **index.blade.php** » dans le répertoire « **resources/views/pages** » et mettez le code ci-dessous :

**@extends('layouts.app')  
  
@section('content')** <**div class="row"**>  
 <**div class="col-md-3"**>  
 <**h1 class="text-danger"**>Listes des films</**h1**>  
 </**div**>  
  
 <**div class="col-md-5"**>  
  
 *<!--Rechercher un film-->* {!! Form::*open*(**array**(**'method'** => **'GET'**)) !!}  
 <**div class="form-group"**>  
 {{Form::*text*(**'keyword'**, **'Recherche par mot clé'**, [**'class'** => **'form-control'**])}}  
 </**div**>  
 {!! Form::*close*() !!}  
 </**div**>  
  
 <**div class="col-md-3"**>  
 *<!--Button Create pour créer un film-->* {!! Form::*open*(**array**(**'method'** => **'GET'**, **'route'** => **'film.create'**)) !!}  
 {!! Form::*submit*(**'Create'**, [**'class'** => **'btn btn-primary'**]) !!}  
 {!! Form::*close*() !!}  
  
 </**div**>  
 </**div**>  
  
 *<!--Affichage de la liste des videos-->* <**table class="table table-striped"**>  
 <**thead**>  
 <**tr**>  
 <**th**>Titre</**th**>  
 <**th**>Catégorie</**th**>  
 <**th**>Année</**th**>  
 <**th**>RETRIEVE</**th**>  
 <**th**>UPDATE</**th**>  
 <**th**>DELETE</**th**>  
 </**tr**>  
 </**thead**>  
 <**tbody**>  
 **@forelse(**$films **as** $film**)** <**tr**>  
 <**td**>{{$film->**titre**}}</**td**>  
 <**td**>  
 **@forelse(**$film->categories()->get() **as** $categorie**)** <**small**>{{$categorie->**nom**.**" "**}}</**small**>  
 **@empty** <**small**>Pas de catégorie.</**small**>  
 **@endforelse** </**td**>  
 <**td**>{{$film->**annee**}}</**td**>  
 <**td**>  
 {!! Form::*open*(**array**(**'method'** => **'GET'**, **'route'** => [**'film.show'**, $film->**id**])) !!}  
 {!! Form::*submit*(**'RETRIEVE'**, [**'class'** => **'btn btn-primary'**]) !!}  
 {!! Form::*close*() !!}  
 </**td**>  
  
 <**td**>  
 {!! Form::*open*(**array**(**'method'** => **'GET'**, **'url'** => **'film/'**.$film->**id**.**'/edit'**)) !!}  
 {!! Form::*submit*(**'EDIT'**, [**'class'** => **'btn btn-primary'**]) !!}  
 {!! Form::*close*() !!}  
 </**td**>  
  
 <**td**>  
 {!! Form::*open*(**array**(**'method'** => **'DELETE'**, **'route'** => [**'film.destroy'**, $film->**id**])) !!}  
 {!! Form::*submit*(**'DELETE'**, [**'class'** => **'btn btn-primary'**]) !!}  
 {!! Form::*close*() !!}  
 </**td**>  
 </**tr**>  
 **@empty** <**tr**>  
 <**td class="text-info"**><**span**>No Film.</**span**></**td**>  
 </**tr**>  
 **@endforelse** </**tbody**>  
 </**table**>  
  
 *<!--Affichage des boutons de pagination -->* {!! $films->render() !!}  
**@endsection**

A ce niveau, si vous lancer l’application sur la route « **/film** », on aura l’illustration ci-dessous :

C’est normal car il n’y’a pas de film insérer par contre, il y’a un bouton « Create » qui nous permet de créer un film, du coup nous allons implémenter cette vue pour pouvoir ajouter un nouveau film.

🡪**films.blade.php :** ce fichier contient le formulaire qui va permettre d’ajouter ou de modifier un film et ce fichier existe déjà dans le projet donc pas besoin de le créer on va juste remplace son contenu par :

**@extends('layouts.app')  
  
@section('content')** <**div class="row"**>  
 <**div class="col-md-offset-1 col-md-7"**>  
 {!! Form::*open*(**array**(**'route'** => **'film.store'**, **'method'** => **'POST'**)) !!}  
 <**div class="form-group"**>  
 {!! Form::*label*(**'titre'**, **'Titre:'**, [**'class'** => **'text-danger'**]) !!}  
 {!! Form::*text*(**'titre'**, **'Titre..'**, [**'class'** => **'form-control'**]) !!}  
 </**div**>  
  
 <**div class="form-group"**>  
 {!! Form::*label*(**'categorie'**, **'Categorie:'**, [**'class'** => **'text-danger'**]) !!}  
  
 <**select name="cat\_id" class="form-control"**>  
 **@foreach(**$categories **as** $cat**)** <**option value="**{{$cat->**id**}}**"**>{{$cat->**nom**}}</**option**>  
 **@endforeach** </**select**>  
 </**div**>  
  
 <**div class="form-group"**>  
 {!! Form::*label*(**'mes'**, **'Metteur en scène:'**, [**'class'** => **'text-danger'**]) !!}  
  
 <**select name="mes\_id" class="form-control"**>  
 **@foreach(**$mess **as** $mes**)** <**option value="**{{$mes->personne()->first()->**id**}}**"**>{{$mes->personne()->first()->**nom**}}</**option**>  
 **@endforeach** </**select**>  
 </**div**>  
  
 <**div class="form-group"**>  
 {!! Form::*label*(**'annee'**, **'Annee:'**, [**'class'** => **'text-danger'**]) !!}  
 {!! Form::*number*(**'annee'**, **'2016'**, [**'class'** => **'form-control'**]) !!}  
 </**div**>  
  
 <**div class="form-group"**>  
 {!! Form::*label*(**'description'**, **'Description:'**) !!}  
 {!! Form::*textarea*(**'description'**, **'Description...'**, [**'class'** => **'form-control'**, **'cols'** => **'35'**, **'rows'** => **'3'**]) !!}  
 </**div**>  
  
 <**div class="form-group"**>  
 {!! Form::*submit*(**'Ajouter'**, [**'class'** => **'btn btn-danger'**]) !!}  
 </**div**>  
 {!! Form::*close*() !!}  
 </**div**>  
 </**div**>  
**@endsection**

A ce niveau on est en mesure d’ajouter un film et de le supprimer par contre un clique sur le l’édition (Bouton « **EDIT** ») ou la récupération (Bouton « **Retrieve** ») pose un problème, donc on va s’attaquer à la récupération pour afficher le film.

🡪**show\_film.balde.php** : est le fichier qui va afficher les informations sur un film lorsque l’on clique sur le Bouton « **RETRIEVE** ». Créer ce fichier et mettre le contenue ci-dessous :

**@extends('layouts.app')  
  
@section('content')** <**div class="row"**>  
 <**div class="col-md-10 col-md-offset-1"**>  
   
 <**div class="col-md-3"**>  
 <**img src="http://lorempixel.com/200/200"** />  
 </**div**>  
   
 <**div class="col-md-7"**>  
 <**h3 class="text-danger"**>Titre : <**small**>{{$film->**titre**}}</**small**></**h3**>  
 <**label class="text-danger"**>Année : <**small**>{{$film->**annee**}}</**small**></**label**> <**br**/>  
 <**label class="text-danger"**>Categories :  
 **@forelse(**$categories **as** $category**)** <**small**>{{$category->**nom**}} </**small**>  
 **@empty** <**small**>Aucune catégorie.</**small**>  
 **@endforelse** </**label**> <**br**/>  
  
 <**label class="text-danger"**>Roles :  
 **@forelse(**$film->roles()->get() **as** $role**)** <**small**>{{$role->**nom**}} </**small**>  
 **@empty** <**small**>Aucun role.</**small**>  
 **@endforelse** </**label**> <**br**/>  
  
 <**p**>  
 {{$film->**description**}}  
 </**p**>  
 </**div**>  
 </**div**>  
 </**div**>  
  
**@endsection**

A ce niveau l’affichage d’un film marche bien, il reste maintenant l’édition du film.

🡪 Avant de commencer l’édition du formulaire, on doit remarquer que le formulaire de création et d’édition de film contiennent les mêmes informations, donc on peut mettre ces formulaires dans un fichier et l’importer dans les différents formulaires.

Pour adapter les deux formulaires « Création et Edition », on va procéder comme suit :

1. Créer un fichier « **formulaire.blade.php** » qui va contenir le formulaire.
2. Modifier le fichier de création du formulaire « **films.blade.php** » pour inclure le formulaire créer précédemment et enlever le formulaire créer en dur.
3. Adapter les Controller à cette nouvelle structure.
4. Commençons par créer un fichier formulaire « **formulaire.blade.php** » et mettons le formulaire ci-dessous (Celui de création et l’édition de film) :
5. **<?php** *//Permet de spécifier la bonne route* **if**($film->**id**){  
    $option = **array**(**'route'** => [**'film.update'**, $id], **'method'** => **'PUT'**);  
    }**else**{  
    $option = **array**(**'route'** => **'film.store'**, **'method'** => **'POST'**);  
    }  
   **?>**{!! Form::*model*($film, $option) !!}  
   <**div class="form-group"**>  
    {!! Form::*label*(**'titre'**, **'Titre:'**, [**'class'** => **'text-danger'**]) !!}  
    {!! Form::*text*(**'titre'**, **null**, [**'class'** => **'form-control'**]) !!}  
   </**div**>  
     
   <**div class="form-group"**>  
    {!! Form::*label*(**'categorie'**, **'Categorie:'**, [**'class'** => **'text-danger'**]) !!}  
     
    <**select name="cat\_id" class="form-control"**>  
    **@foreach(**$categories **as** $cat**)** <**option value="**{{$cat->**id**}}**"**>{{$cat->**nom**}}</**option**>  
    **@endforeach** </**select**>  
   </**div**>  
     
   <**div class="form-group"**>  
    {!! Form::*label*(**'mes'**, **'Metteur en scène:'**, [**'class'** => **'text-danger'**]) !!}  
     
    <**select name="mes\_id" class="form-control"**>  
    **@foreach(**$mess **as** $mes**)** <**option value="**{{$mes->personne()->first()->**id**}}**"**>{{$mes->personne()->first()->**nom**}}</**option**>  
    **@endforeach** </**select**>  
   </**div**>  
     
   <**div class="form-group"**>  
    {!! Form::*label*(**'annee'**, **'Annee:'**, [**'class'** => **'text-danger'**]) !!}  
    {!! Form::*number*(**'annee'**, **null**, [**'class'** => **'form-control'**]) !!}  
   </**div**>  
     
   <**div class="form-group"**>  
    {!! Form::*label*(**'description'**, **'Description:'**) !!}  
    {!! Form::*textarea*(**'description'**, **null**, [**'class'** => **'form-control'**, **'cols'** => **'35'**, **'rows'** => **'3'**]) !!}  
   </**div**>  
     
   <**div class="form-group"**>  
    {!! Form::*submit*(**'Ajouter'**, [**'class'** => **'btn btn-danger'**]) !!}  
   </**div**>  
   {!! Form::*close*() !!}
6. On modifie ensuite le contenue de fichier « **films.blade.php** » par le contenue suivante :

**@extends('layouts.app')  
  
@section('content')** <**div class="row"**>  
 <**div class="col-md-offset-1 col-md-7"**>  
 **@include('pages.formulaire')** </**div**>  
 </**div**>  
**@endsection**

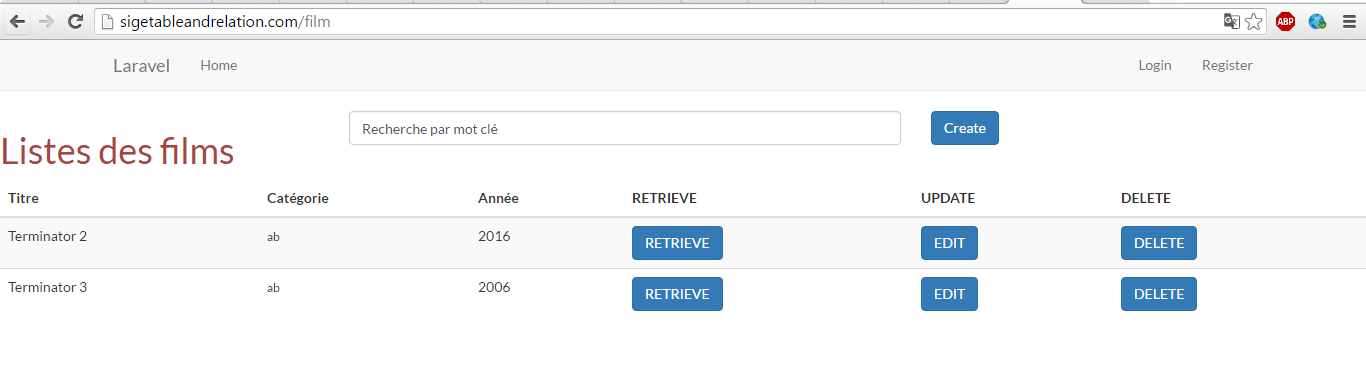
1. On doit adapter le Controller de « **FilmController** » précisément la méthode « **create()** » pour qu’il retourne un film vide vu que pour l’édition « méthode **edit()** » on retourne le film à éditer, ainsi c’est nécessaire pour avoir un formulaire commun. Les méthode « **create() & edit()** » du Controller « **FilmController** » devient :
2. **public function** create()  
   {  
    *//On recupere toutes les categories* $categories = \App\Model\Categorie::*orderBy*(**'nom'**, **'asc'**)->get();  
     
    *//On recupère le metteur en scène avec les personne asscoier* $mess = \App\Model\Mes::*with*(**'personne'**)->get();  
     
    *//Creation d'un film vide pour le formulaire* $film = **new** \App\Model\Film;  
     
    **return** view(**'pages.films'**, [**'categories'** => $categories, **'mess'** => $mess, **'film'** => $film]);  
   }

**public function** edit($id)  
{  
 *//On recupère l'instance de film* $film = \App\Model\Film::*find*($id);  
  
 *//On recupere toutes les categories* $categories = \App\Model\Categorie::*orderBy*(**'nom'**, **'asc'**)->get();  
  
 *//On recupère le metteur en scène avec les personne asscoier* $mess = \App\Model\Mes::*with*(**'personne'**)->get();  
  
 **return** view(**'pages.films'**, [**'film'** => $film, **'categories'** => $categories, **'mess'** => $mess]);  
}

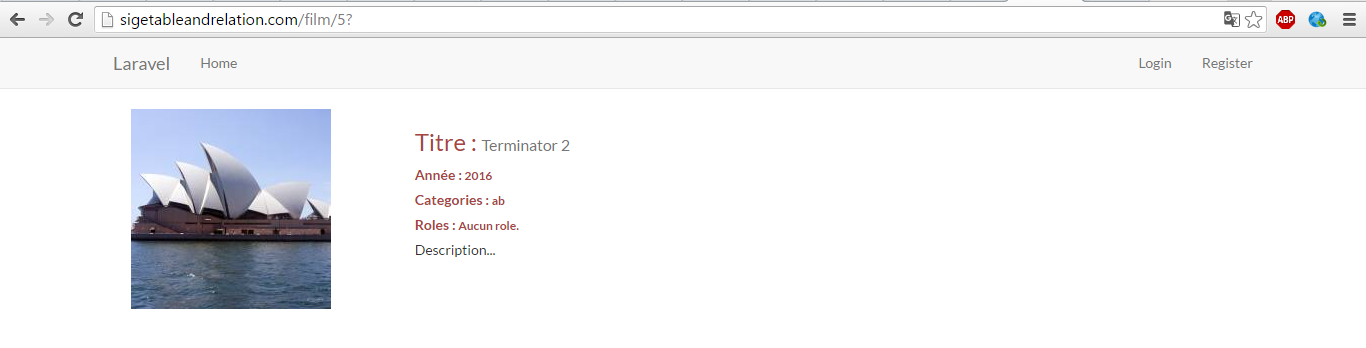
A ce niveau, l’application est fonctionnelle, on peut y ajouter, modifier et récupérer des films.

Ci-dessous une capture d’écran des pages :

Liste des films :



Affichage d’un film :



On remarque que nos films n’ont pas de rôles, donc ceci sera l’objet la partie suivante.

En considérant que l’ajout de rôle à un film est une modification, nous allons intéresser à vue utiliser pour l’édition de formulaire « **films.blade.php** », comme ce fichier est aussi utilisé pour la création de film, nous allons tester si c’est l’édition on inclut la vue nécessaire pour ajouter des rôles à au film.

Pour ce faire nous allons procéder en 3 temps :

1. Création de la vue nécessaire pour l’ajout de rôles «**ajout\_role.blade.php**».
2. Modifier le fichier « **films.blade.php** » pour pouvoir ajouter la vue d’ajout de rôle.
3. Configurer les routes pour gérer ces actions.
4. Modifier le Controller **FilmController** pour ajouter l’action nécessaire à ajout de role.
5. Créer le fichier « **ajout\_role.blade.php** » et ajouter le code ci-dessous qui permet d’ajouter le rôle :

<**div class="row"**>  
 <**div class="col-md-10 col-md-offset-1"**>  
 <**div class="col-md-5"**>  
 {{Form::*open*([**'method'** => **'POST'**, **'action'** => **'FilmController@addRole'**])}}  
  
 <**div class="form-group"**>  
 <**h3**>Ajouter role</**h3**>  
 <**input name="idFilm" type="hidden" value="**{{$id}}**"**/>  
 </**div**>  
  
 <**div class="form-group"**>  
 {{Form::*label*(**'Role'**, **'Role'**, [**'class'** => **'text-primary'**])}}  
 {{Form::*text*(**'role'**, **null**, [**'class'** => **'form-control'**, **'placeholder'** => **'Role'**])}}  
 </**div**>  
  
 <**div class="form-group"**>  
 {{Form::*submit*(**'Ajouter Role'**, [**'class'** => **'btn btn-danger'**])}}  
 </**div**>  
 {{Form::*close*()}}  
 </**div**>  
  
 <**div class="col-md-5"**>  
 {{Form::*open*([**'method'** => **'POST'**, **'action'** => **'FilmController@roleToActeur'**])}}  
  
 <**div class="form-group"**>  
 <**h3**>Associer un acteur à un role</**h3**>  
 <**input name="idFilm" type="hidden" value="**{{$id}}**"**/>  
 </**div**>  
  
 *<!-- Liste des roles films -->* <**div class="form-group"**>  
 <**select name="role\_id" class="form-control"**>  
 **@forelse(**$film->roles()->get() **as** $role**)** <**option value="**{{$role->**id**}}**"**>{{$role->**nom**}}</**option**>  
 **@empty** <**option value="-11"**>Pas de role</**option**>  
 **@endforelse** </**select**>  
 </**div**>  
  
 *<!--Liste des acteurs-->* <**div class="form-group"**>  
 <**select name="acteur\_id" class="form-control"**>  
 **@forelse(**$acteurs **as** $acteur**)** <**option value="**{{$acteur->personne()->first()->**id**}}**"**>{{$acteur->personne()->first()->**nom**}}</**option**>  
 **@empty** <**option value="-12"**>Pas d'acteur</**option**>  
 **@endforelse** </**select**>  
 </**div**>  
  
 <**div class="form-group"**>  
 {{Form::*submit*(**'Role => Acteur'**, [**'class'** => **'btn btn-primary'**])}}  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>

1. On va maintenant ajouter le code ci-dessous à dans le fichier « **films.balde.php** » après le formulaire :

**@if(**$film->**id)  
 @include('pages.ajout\_role')  
@endif**

1. A ce niveau l’édition ne marche pas à cause des routes et le Controller, du coup on va configurer les routes pour gérer l’action des formulaires, on ajouter les 2 routes ci-dessous dans le fichier « **routes.php** » :

Route::*post*(**'film/addRole'**, **'FilmController@addRole'**);  
Route::*post*(**'film/roleToActeur'**, **'FilmController@roleToActeur'**);

Comme on peut le remarquer, on doit à ce point définir deux méthodes dans « **FilmController** » à savoir (**addRole() & roleToActeur()**).

1. La définition des 2 actions sont les suivantes « **FilmController** » :

*/\*\*  
 \* Fonction utiliser pour ajouter le role à film  
 \*/***public function** addRole()  
{  
 $film = \App\Model\Film::*find*(request()->get(**'idFilm'**));  
  
 $film->roles()->save(**new** \App\Model\Role([**'nom'** => request()->get(**'role'**)]));  
  
 **return** redirect(**'/film/'**.request()->get(**'idFilm'**).**'/edit'**);  
}  
  
  
 */\*\*  
 \** ***@return*** *\Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector  
 \*/* **public function** roleToActeur(){  
  
 *//Recupere le film en question* $film = \App\Model\Film::*with*(**'roles'**)  
 ->find(request()->get(**'idFilm'**));  
  
 *//On recupère l'acteur* $acteur = \App\Model\Acteur::*find*(request()->get(**'acteur\_id'**));  
  
 *//Association du role à l'acteur* $film->roles()  
 ->where(**'id'**, request()->get(**'role\_id'**))  
 ->get()->first()  
 ->acteur()->associate($acteur)->save();  
  
 **return** redirect(**'/film/'**.request()->get(**'idFilm'**).**'/edit'**);  
}

A ce niveau on peut effectuer toute sorte d’opération sur le film, Mais lors de l’affichage on n’affiche pas les acteurs, nous allons donc ajouter cette dernière fonctionnalité.

Pour ce faire nous allons remplacer le contenu de la vue **« show\_film.blade.php** » par :

**@extends('layouts.app')  
  
@section('content')** <**div class="row"**>  
 <**div class="col-md-10 col-md-offset-1"**>  
   
 <**div class="col-md-3"**>  
 <**img src="http://lorempixel.com/200/200"** />  
 </**div**>  
   
 <**div class="col-md-7"**>  
 <**h3 class="text-danger"**>Titre : <**small**>{{$film->**titre**}}</**small**></**h3**>  
 <**label class="text-danger"**>Année : <**small**>{{$film->**annee**}}</**small**></**label**> <**br**/>  
 <**label class="text-danger"**>Categories :  
 **@forelse(**$categories **as** $category**)** <**small**>{{$category->**nom**}} </**small**>  
 **@empty** <**small**>Aucune catégorie.</**small**>  
 **@endforelse** </**label**> <**br**/>  
  
 <**label class="text-danger"**>Roles :  
 **@forelse(**$film->roles()->get() **as** $role**)** <**small**>{{$role->**nom**.**' ( '**.$role->acteur()->first()->personne()->first()->**nom**.**' ), '**}} </**small**>  
 **@empty** <**small**>Aucun role.</**small**>  
 **@endforelse** </**label**> <**br**/>  
  
 <**p**>  
 {{$film->**description**}}  
 </**p**>  
 </**div**>  
 </**div**>  
 </**div**>  
  
**@endsection**

Finalement, tout doit marcher à merveille.

**Conclusion :**

Pour exécuter ce projet, il faut suivre les étapes suivantes :

1. Télécharger le projet depuis git sur le lien : [TableAndRealtion](https://github.com/ArianeExtern/EtudeLaravel/tree/Crud).
2. Si vous utilisez **Homestead**, configurer le fichier **Homestead.yaml** selon votre machine.
3. Lancer l’hôte spécifié dans le fichier « **Homestead.yaml** » après l’avoir ajouter dans le fichier « **hosts** ».