### Solution partielle

#### Solution pour la classe Show

##### Mapping relationnel, migration et données test

La classe *Show* est en relation *OneToMany inverse* avec la classe *Location*.

1. Modifier le modèle *Location* de façon à ajouter la relation *OneToMany* avec la classe *Show*.

//…

class Location extends Model

{

protected $fillable = [

'slug',

'designation',

'address',

'locality\_id',

'website',

'phone',

];

protected $table = 'locations';

public $timestamps = false;

/\*\*

\* Get the locality of the location

\*/

public function locality()

{

return $this->belongsTo('App\Locality');

}

/\*\*

\* Get the shows in this location.

\*/

public function shows()

{

return $this->**hasMany**('App\Show');

}

}

1. Créer le modèle *Show* (et sa relation avec le modèle *Location*)  
   Notez que nous activons le behaviour Timestamp pour ce modèle : date de création et de modification seront automatiquement renseignées.

**php artisan make:model Show**

*app\Show.php*

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Show extends Model

{

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'slug',

'title',

'description',

'poster\_url',

'location\_id',

'bookable',

'price',

];

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'shows';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

**public $timestamps = true;**

/\*\*

\* Get the main location of the show

\*/

public function location()

{

return $this->**belongsTo**('App\Location');

}

}

1. Créer et modifier la migration

php artisan make:migration create\_shows\_table

<https://laravel.com/docs/7.x/migrations>

Modifier le fichier généré : ajouter les champs et leurs contraintes. Définir les contraintes de clé étrangère pour le champ *location\_id*. Ajouter la contrainte d’unicité pour le champ *slug*.

*database\migrations\YYYY\_MM\_DD\_HHMMSS\_create\_locations\_table.php*

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateShowsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('shows', function (Blueprint $table) {

$table->id();

**$table->string('slug',60)->unique();**

$table->string('title',255);

$table->text('description')->nullable();

$table->string('poster\_url',255)->nullable();

**$table->foreignId('location\_id')->nullable();**

$table->boolean('bookable')->default(false);

$table->decimal('price',10,2)->nullable();

**$table->timestamp('created\_at')->useCurrent();**

**$table->timestamp('updated\_at')->nullable();**

**$table->foreign('location\_id')->references('id')->on('locations')**

**->onDelete('restrict')->onUpdate('cascade');**

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('shows');

}

}

1. Exécuter les migrations

php artisan migrate

1. Créer et modifier une classe de *seeders*

php artisan make:seeder ShowsTableSeeder

*database\seeds\ShowsTableSeeder.php*

<?php

use Illuminate\Database\Seeder;

**use Illuminate\Support\Str;**

**use App\Location;**

**use App\Show;**

class ShowsTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

//Empty the table first

Show::truncate();

//Define data

$shows = [

[

'slug'=>null,

'title'=>'Ayiti',

'description'=>**"Un homme est bloqué à l’aéroport.\n "**

. 'Questionné par les douaniers, il doit alors justifier son identité, '

. 'et surtout prouver qu\'il est haïtien – qu\'est-ce qu\'être haïtien ?',

'poster\_url'=>'ayiti.jpg',

**'location\_slug'=>'espace-delvaux-la-venerie',**

'bookable'=>true,

'price'=>8.50,

],

[

'slug'=>null,

'title'=>'Cible mouvante',

'description'=>'Dans ce « thriller d’anticipation », des adultes semblent alimenter '

. 'et véhiculer une crainte féroce envers les enfants âgés entre 10 et 12 ans.',

'poster\_url'=>'cible.jpg',

'location\_slug'=>'dexia-art-center',

'bookable'=>true,

'price'=>9.00,

],

[

'slug'=>null,

'title'=>'Ceci n\'est pas un chanteur belge',

'description'=>**"Non peut-être ?!\nEntre Magritte (pour le surréalisme comique) "**

. 'et Maigret (pour le réalisme mélancolique), ce dixième opus semalien propose '

. 'quatorze nouvelles chansons mêlées à de petits textes humoristiques et '

. 'à quelques fortes images poétiques.',

'poster\_url'=>'claudebelgesaison220.jpg',

**'location\_slug'=>null,**

'bookable'=>false,

'price'=>5.50,

],

[

'slug'=>null,

'title'=>'Manneke… !',

'description'=>'A tour de rôle, Pierre se joue de ses oncles, '

. 'tantes, grands-parents et surtout de sa mère.',

'poster\_url'=>'wayburn.jpg',

'location\_slug'=>'la-samaritaine',

'bookable'=>true,

'price'=>10.50,

],

];

//Insert data in the table

foreach ($shows as $data) {

**$location = Location::firstWhere('slug',$data['location\_slug']);**

DB::table('shows')->insert([

**'slug' => Str::slug($data['title'],'-'),**

'title' => $data['title'],

'description' => $data['description'],

'poster\_url' => $data['poster\_url'],

**'location\_id' => $location->id ?? null,**

'bookable' => $data['bookable'],

'price' => $data['price'],

]);

}

}

}

Nous associons un spectacle *Show* avec son lieu de création *Location* en recherchant le lieu par son slug. Si le lieu n’est pas trouvé ou que le spectacle n’a pas de lieu de création, nous mettons *null*. De plus, nous utilisons la méthode *Str::slug* pour générer automatiquement le champ *slug* sur base du titre.

Attention, nous devons aussi modifier la classe *LocationsTableSeeder* puisqu’il y a une contrainte de clé étrangère *ON DELETE restrict*.

1. Modifier la classe de *seeders LocationsTableSeeder*

Nous désactivons la vérification des contraintes de clés étrangères avant de vider la table, puis nous la réactivons. Attention à ne faire cela qu’en mode développement, pas en production.

//…

**use App\Location;**

class LocationsTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

//Empty the table first

**DB::statement('SET FOREIGN\_KEY\_CHECKS=0');**

Location::truncate();

**DB::statement('SET FOREIGN\_KEY\_CHECKS=1');**

//…

1. Exécuter les seeders
   1. Dans le fichier *DatabaseSeeder.php*, il faut d’abord ajouter la classe *ShowsTableSeeder* à la liste des seeders à exécuter :

*database\seeds\DatabaseSeeder.php*

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

/\*\*

\* Seed the application's database.

\*

\* @return void

\*/

public function run()

{

$this->call(**[**

ArtistsTableSeeder::class,

TypesTableSeeder::class,

LocalitiesTableSeeder::class,

RolesTableSeeder::class,

LocationsTableSeeder::class,

ShowsTableSeeder::class,

**]**);

}

}

* + 1. Ensuite, il faut regénérer le fichier autoloader de Composer !

composer dump-autoload

* + 1. Enfin, nous pouvons exécuter les seeders :

php artisan db:seed



1. Versionner le projet

git status

git add .

git commit -m "Modèle Show +Relation ManyToOne with Location"

##### Contrôleurs, templates et routes

1. Créer et modifier le contrôleur *LocationController*

Nous voulons à présent afficher un lieu de spectacle et tous les spectacles qui s’y trouvent. Si ce n’est déjà fait, ajoutons du code dans la méthode *show*.

php artisan make:controller LocationController --resource

*App\Http\Controllers\LocationController.php*

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

**use App\Location;**

class LocationController extends Controller

{

//…

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

$location = Location::find($id);

return view('locations.show',[

'location' => $location,

]);

}

//…

}

1. Créer les templates du modèle Location (index et show)

Créer le dossier *location*, puis copier et modifier le template *index.blade.php*.

*ressources/views/location/index.blade.php*

@extends('layouts.app')

@section('title', 'Liste des lieux de spectacle')

@section('content')

<h1>Liste des {{ $resource }}</h1>

<ul>

@foreach($locations as $location)

<li>{{ $location->designation }}

**@if($location->website)**

- <a href="{{ $location->website }}">{{ $location->website }}</a>

**@endif**

</li>

@endforeach

</ul>

@endsection

<https://laravel.com/docs/7.x/blade#if-statements>

*ressources/views/location/show.blade.php*

@extends('layouts.app')

@section('title', 'Fiche d\'un lieu de spectacle')

@section('content')

<article>

<h1>{{ $location->designation }}</h1>

<address>

<p>{{ $location->address }}</p>

<p>{{ $location->locality->postal\_code }}

{{ $location->locality->locality }}

</p>

@if($location->website)

<p><a href="{{ $location->website }}" target="\_blank">{{ $location->website }}</a></p>

@else

<p>Pas de site web</p>

@endif

@if($location->phone)

<p><a href="tel:{{ $location->phone }}">{{ $location->phone }}</a></p>

@else

<p>Pas de téléphone</p>

@endif

</address>

<h2>Liste des spectacles</h2>

<ul>

@foreach($location->shows as $show)

<li>{{ $show->title }}</li>

@endforeach

</ul>

@endsection

1. Enfin, nous devons ajouter une route afin de pouvoir accéder à cette page à travers notre application. La route get('location/{id}') envoie l’id spécifiée dans l’URL comme paramètre de la méthode *show* du contrôleur *LocationController*.

*routes/web.php*

<?php

//…

Route::get('artist', 'ArtistController@index');

Route::get('artist/{id}', 'ArtistController@show');

Route::get('type', 'TypeController@index');

Route::get('type/{id}', 'TypeController@show');

Route::get('locality', 'LocalityController@index');

Route::get(' locality/{id}', 'LocalityController@show');

Route::get('role', 'RoleController@index');

Route::get('role/{id}', 'RoleController@show');

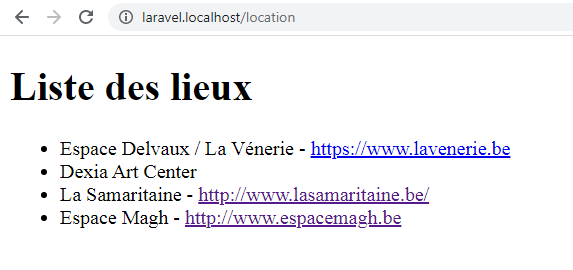
**Route::get('location', 'LocationController@index');**

**Route::get('location/{id}', 'LocationController@show');**

* Lancer le serveur interne (ou accéder directement à l’URL de votre Virtual Host)

php artisan serve

* Accéder à l’URL <http://localhost:8000/location> (ou <http://laravel.localhost/location>)
* Accéder à l’URL <http://localhost:8000/location/1> (ou <http://laravel.localhost/location/>1)

1. Versionner le projet

git status

git add .

git commit -m "Controller Location +templates +routes"

1. Créer et modifier le contrôleur *ShowController*

Nous voulons à présent afficher la liste des spectacles, ainsi qu’un spectacle et son lieu de spectacle. Ajoutons du code dans les méthodes *index* et *show*.

php artisan make:controller ShowController --resource

*App\Http\Controllers\*ShowController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

**use App\Show;**

class LocationController extends Controller

{

//…

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

$shows = Show::all();

return view('show.index',[

'shows' => $shows,

'resource' => 'spectacles',

]);

}

//…

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

$show = Show::find($id);

return view('show.show',[

'show' => $show,

]);

}

//…

}

1. Créer les templates du modèle Show (index et show)

Créer le dossier *show*, puis copier et modifier le template *index.blade.php*.

*ressources/views/show/index.blade.php*

@extends('layouts.app')

@section('title', 'Liste des spectacles')

@section('content')

<h1>Liste des {{ $resource }}</h1>

<ul>

@foreach($shows as $show)

<li>

{{ $show->title }}

@if($show->bookable)

<span>{{ $show->price }} €</span>

@endif

</li>

@endforeach

</ul>

@endsection

<https://laravel.com/docs/7.x/blade#if-statements>

*ressources/views/show/show.blade.php*

@extends('layouts.app')

@section('title', 'Fiche d\'un spectacle')

@section('content')

<article>

<h1>{{ $show->title }}</h1>

@if($show->poster\_url)

<p><img src="{{ **asset('images/'.$show->poster\_url)** }}" alt="{{ $show->title }}" width="200"></p>

@else

<canvas width="200" height="100" style="border:1px solid #000000;"></canvas>

@endif

**@if($show->location)**

<p><strong>Lieu de création:</strong> {{ $show->location->designation }}</p>

**@endif**

<p><strong>Prix:</strong> {{ $show->price }} €</p>

**@if($show->bookable)**

<p><em>Réservable</em></p>

**@else**

<p><em>Non réservable</em></p>

**@endif**

@endsection

1. Ajouter les routes pour les actions *index* et *show*.

*routes/web.php*

<?php

//…

Route::get('artist', 'ArtistController@index');

Route::get('artist/{id}', 'ArtistController@show');

Route::get('type', 'TypeController@index');

Route::get('type/{id}', 'TypeController@show');

Route::get('locality', 'LocalityController@index');

Route::get(' locality/{id}', 'LocalityController@show');

Route::get('role', 'RoleController@index');

Route::get('role/{id}', 'RoleController@show');

Route::get('location', 'LocationController@index');

Route::get('location/{id}', 'LocationController@show');

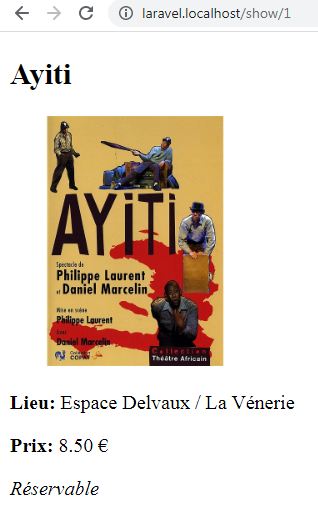
**Route::get('show', 'ShowController@index');**

**Route::get('show/{id}', 'ShowController@show');**

* Lancer le serveur interne (ou accéder directement à l’URL de votre Virtual Host)

php artisan serve

* Accéder à l’URL <http://localhost:8000/show> (ou <http://laravel.localhost/show>)
* Accéder à l’URL <http://localhost:8000/show/1> (ou <http://laravel.localhost/show/>1)

1. Versionner le projet

git status

git add .

git commit -m "Controller Show +templates +routes"

#### Solution pour la classe Representation

##### Mapping relationnel, migration et données test

La classe *Representation* est en relation *OneToMany inverse* avec la classe *Location,* mais aussien relation *OneToMany inverse* avec la classe *Show*.

1. Modifier le modèle *Location* de façon à ajouter la relation *OneToMany* avec la classe *Representation*.

//…

class Location extends Model

{

protected $fillable = [

'slug',

'designation',

'address',

'locality\_id',

'website',

'phone',

];

protected $table = 'locations';

public $timestamps = false;

/\*\*

\* Get the locality of the location

\*/

public function locality()

{

return $this->belongsTo('App\Locality');

}

/\*\*

\* Get the shows in this location.

\*/

public function shows()

{

return $this->hasMany('App\Show');

}

/\*\*

\* Get the representations in this location.

\*/

public function representations()

{

return $this->**hasMany**('App\Representation');

}

}

1. Modifier le modèle *Show* de façon à ajouter la relation *OneToMany* avec la classe *Representation*.

//…

class Show extends Model

{

protected $fillable = [

'slug',

'title',

'poster\_url',

'location\_id',

'bookable',

'price',

];

protected $table = 'shows';

public $timestamps = true;

/\*\*

\* Get the main location of the show

\*/

public function location()

{

return $this->belongsTo('App\Location');

}

/\*\*

\* Get the representations of this show.

\*/

public function representations()

{

return $this->**hasMany**('App\Representation');

}

}

1. Créer le modèle *Representation* (et ses relations avec les modèle *Location* et *Show*)

**php artisan make:model Representation**

*app\Representation.php*

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Representation extends Model

{

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'show\_id',

'when',

'location\_id',

];

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'representations';

/\*\*

\* Indicates if the model should be timestamped.

\*

\* @var bool

\*/

public $timestamps = false;

/\*\*

\* Get the actual location of the representation

\*/

public function location()

{

return $this->**belongsTo**('App\Location');

}

/\*\*

\* Get the show of the representation

\*/

public function show()

{

return $this->**belongsTo**('App\Show');

}

}

1. Créer et modifier la migration

php artisan make:migration create\_representations\_table

<https://laravel.com/docs/7.x/migrations>

Modifier le fichier généré : ajouter les champs et leurs contraintes. Définir les contraintes de clé étrangère pour les champs *location\_id* et *show\_id*. Ajouter la contrainte d’unicité pour le champ *slug*.

*database\migrations\YYYY\_MM\_DD\_HHMMSS\_create\_representations\_table.php*

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateShowsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('shows', function (Blueprint $table) {

$table->id();

**$table->foreignId('location\_id')->nullable();**

**$table->foreignId('show\_id');**

$table->datetime('when');

**$table->foreign('location\_id')->references('id')->on('locations')**

**->onDelete('restrict')->onUpdate('cascade');**

**$table->foreign('show\_id')->references('id')->on('shows')**

**->onDelete('restrict')->onUpdate('cascade');**

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('shows');

}

}

1. Exécuter les migrations

php artisan migrate

1. Créer et modifier une classe de *seeders*

php artisan make:seeder RepresentationsTableSeeder

*database\seeds\RepresentationsTableSeeder.php*

<?php

use Illuminate\Database\Seeder;

**use App\Representation;**

**use App\Location;**

**use App\Show;**

class RepresentationsTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

//Empty the table first

**DB::statement('SET FOREIGN\_KEY\_CHECKS=0');**

Representation::truncate();

**DB::statement('SET FOREIGN\_KEY\_CHECKS=1');**

//Define data

$representations = [

[

'location\_slug'=>'espace-delvaux-la-venerie',

'show\_slug'=>'ayiti',

'when'=>'2012-10-12 13:30',

],

[

'location\_slug'=>'dexia-art-center',

'show\_slug'=>'ayiti',

'when'=>'2012-10-12 20:30',

],

[

**'location\_slug'=>null,**

'show\_slug'=>'cible-mouvante',

'when'=>'2012-10-02 20:30',

],

[

'location\_slug'=>null,

'show\_slug'=>'ceci-nest-pas-un-chanteur-belge',

'when'=>'2012-10-16 20:30',

],

];

//Insert data in the table

foreach ($representations as $data) {

**$location = Location::firstWhere('slug',$data['location\_slug']);**

**$show = Show::firstWhere('slug',$data['show\_slug']);**

DB::table('representations')->insert([

**'location\_id' => $location->id ?? null,**

'show\_id' => $show->id,

'when' => $data['when'],

]);

}

}

}

Nous associons une représentation *Representation* avec son lieu de prestation *Location* en recherchant le lieu par son slug. Si le lieu n’est pas trouvé ou que la représentation n’a pas de lieu de prestation, nous mettons *null*. De même, nous associons la représentation avec son spectacle *Show* en recherchant le spectacle par son slug.

Attention, nous devons aussi modifier la classe *ShowsTableSeeder* puisqu’il y a une contrainte de clé étrangère *ON DELETE restrict*.

1. Modifier la classe de *seeders ShowsTableSeeder*

Nous désactivons la vérification des contraintes de clés étrangères avant de vider la table, puis nous la réactivons. Attention à ne faire cela qu’en mode développement, pas en production.

//…

use App\Location;

**use App\Show;**

class ShowsTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

//Empty the table first

**DB::statement('SET FOREIGN\_KEY\_CHECKS=0');**

Show::truncate();

**DB::statement('SET FOREIGN\_KEY\_CHECKS=1');**

//…

1. Exécuter les seeders
   1. Dans le fichier *DatabaseSeeder.php*, il faut d’abord ajouter la classe *ShowsTableSeeder* à la liste des seeders à exécuter :

*database\seeds\DatabaseSeeder.php*

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

/\*\*

\* Seed the application's database.

\*

\* @return void

\*/

public function run()

{

$this->call(**[**

ArtistsTableSeeder::class,

TypesTableSeeder::class,

LocalitiesTableSeeder::class,

RolesTableSeeder::class,

LocationsTableSeeder::class,

ShowsTableSeeder::class,

RepresentationsTableSeeder::class,

**]**);

}

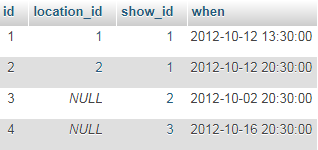
}

* + 1. Ensuite, il faut regénérer le fichier autoloader de Composer !

composer dump-autoload

* + 1. Enfin, nous pouvons exécuter les seeders :

php artisan db:seed



1. Versionner le projet

git status

git add .

git commit -m "Modèle Representation +Relation ManyToOne with Location & Show"

##### Contrôleurs, templates et routes

1. Créer et modifier le contrôleur *RepresentationController*

Nous voulons à présent afficher une représentation avec le titre du spectacle et le lieu de prestation de la représentation ou, à défaut, le lieu de création du spectacle. Ajoutons du code dans la méthode *show*.

php artisan make:controller RepresentationController --resource

*App\Http\Controllers\RepresentationController.php*

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

**use App\Representation;**

**use Carbon\Carbon;**

class RepresentationController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

**$representations = Representation::all();**

return view('representation.index',[

**'representations' => $representations,**

**'resource' => 'représentations',**

]);

}

//…

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

$representation = Representation::find($id);

**$date = Carbon::parse($representation->when)->format('d/m/Y');**

**$heure = Carbon::parse($representation->when)->format('G:i');**

return view('representation.show',[

'representation' => $representation,

**'date' => $date,**

**'heure' => $heure,**

]);

}

//…

}

Notez comme nous utilisons la librairie *Carbon* pour formater la date et l’heure de la représentation.

1. Créer le template du modèle Representation (show)

Créer le dossier *representation*, puis copier et modifier les templates *index.blade.php* et *show.blade.php*.

*ressources/views/representation/index.html.twig*

@extends('layouts.app')

@section('title', 'Liste des représentations')

@section('content')

<h1>Liste des {{ $resource }}</h1>

<ul>

@foreach($representations as $representation)

<li>{{ $representation->show->title }}

**@if($representation->location)**

- <span>{{ $representation->location->designation }}</span>

**@endif**

- <datetime>{{ **substr**($representation->when,0,-3) }}</datetime>

</li>

@endforeach

</ul>

@endsection

*ressources/views/representation/show.blade.php*

@extends('layouts.app')

@section('title', 'Fiche d\'une représentation')

@section('content')

<article>

<h1>Représentation du {{ $date }} à {{ $heure }}</h1>

<p><strong>Spectacle:</strong> {{ $representation->show->title }}</p>

<p><strong>Lieu:</strong>

**@if($representation->location)**

{{ $representation->location->designation }}

**@elseif($representation->show->location)**

{{ $representation->show->location->designation }}

**@else**

<em>à déterminer</em>

**@endif**

</p>

@endsection

1. Enfin, nous devons ajouter les routes afin de pouvoir accéder à ces pages à travers notre application. La route get('representation/{id}') envoie l’id spécifiée dans l’URL comme paramètre de la méthode *show* du contrôleur *RepresentationController*.

*routes/web.php*

<?php

//…

Route::get('artist', 'ArtistController@index');

Route::get('artist/{id}', 'ArtistController@show');

Route::get('type', 'TypeController@index');

Route::get('type/{id}', 'TypeController@show');

Route::get('locality', 'LocalityController@index');

Route::get(' locality/{id}', 'LocalityController@show');

Route::get('role', 'RoleController@index');

Route::get('role/{id}', 'RoleController@show');

Route::get('location', 'LocationController@index');

Route::get('location/{id}', 'LocationController@show');

**Route::get('representation', 'RepresentationController@index');**

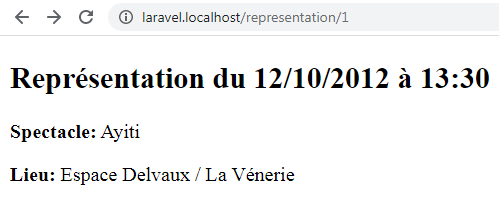
**Route::get('representation/{id}', 'RepresentationController@show');**

* Lancer le serveur interne (ou accéder directement à l’URL de votre Virtual Host)

php artisan serve

* Accéder aux URLs suivantes :
  1. <http://localhost:8000/representation> (ou <http://laravel.localhost/representation>)
  2. <http://localhost:8000/representation/1> (ou <http://laravel.localhost/representation/1>)
  3. <http://localhost:8000/representation/4> (ou <http://laravel.localhost/representation/4>)



1. Versionner le projet

git status

git add .

git commit -m "Controller Representation +templates +routes"

1. Modifier le contrôleur *ShowController*

Nous voulons à présent afficher le nombre total et la liste des représentations d’un spectacle. Ajoutons du code dans les templates *index* et *show*.

*ressources/views/show/index.blade.php*

@extends('layouts.app')

@section('title', 'Liste des spectacles')

@section('content')

<h1>Liste des {{ $resource }}</h1>

<ul>

@foreach($shows as $show)

<li>

{{ $show->title }}

@if($show->bookable)

<span>{{ $show->price }} €</span>

@endif

**@if($show->representations->count()==1)**

- <span>1 représentation</span>

**@elseif($show->representations->count()>1)**

- <span>{{ $show->representations->count() }} représentations</span>

**@else**

- <em>aucune représentation</em>

**@endif**

</li>

@endforeach

</ul>

@endsection

<https://laravel.com/docs/7.x/blade#if-statements>

*ressources/views/show/show.blade.php*

@extends('layouts.app')

@section('title', 'Fiche d\'un spectacle')

@section('content')

<article>

<h1>{{ $show->title }}</h1>

@if($show->poster\_url)

<p><img src="{{ **asset('images/'.$show->poster\_url)** }}" alt="{{ $show->title }}" width="200"></p>

@else

<canvas width="200" height="100" style="border:1px solid #000000;"></canvas>

@endif

@if($show->location)

<p><strong>Lieu de création:</strong> {{ $show->location->designation }}</p>

@endif

<p><strong>Prix:</strong> {{ $show->price }} €</p>

@if($show->bookable)

<p><em>Réservable</em></p>

@else

<p><em>Non réservable</em></p>

@endif

<h2>Liste des représentations</h2>

**@if($show->representations->count()>=1)**

<ul>

**@foreach ($show->representations as $representation)**

<li>{{ $representation->when }}</li>

**@endforeach**

</ul>

**@else**

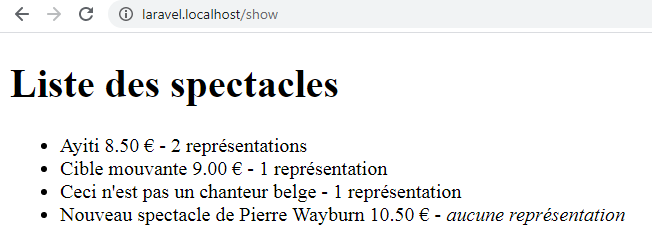
<p>Aucune représentation</p>

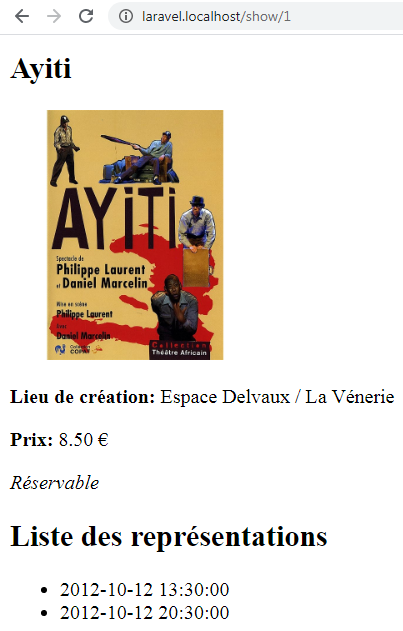
**@endif**

@endsection

* Lancer le serveur interne (ou accéder directement à l’URL de votre Virtual Host)

php artisan serve

* Accéder à l’URL <http://localhost:8000/show> (ou <http://laravel.localhost/show>)
* Accéder à l’URL <http://localhost:8000/show/1> (ou <http://laravel.localhost/show/>1)

1. Versionner le projet

git status

git add .

git commit -m "Templates Show: affichage des représentations"