POUR CRÉER NOUVEAU PROJET LARAVEL :

composer create-project laravel/laravel scolarite

1. **POUR ACCEDER AU PROJET TAPER LA COMMANDE SUIVANTE :**

Cd scolarite

1. TAPER LA COMMANDE SUIVANTE :

npm install

npm run dev

1. **POUR TELECHARGER UN PROJET DE GITHUB :**

ALLER DANS LE DOSSIER DESIGNER POUR COLLER LE PROJET -> BOUTON DROIT DE LA SOURIS -> GIT BASH HERE

1. **TAPER LA COMMANE SUIVANTE :**

Git clone <https://github.com/samir-Gamal/school-managment-system.git> (lien exemple)

Cliquer enter

1. **POUR TELECHARGER LES FICHIER DU « VENDOR » TAPER LA COMMANDE SUIVANTE :**

Composer install

1. **CRÉER LE FICHIER « .ENV » PUIS TAPER LA COMMANE SUIVANTE POUR GENERER « APP\_KEY » :**

php artisan key :generate

1. **POUR AFFICHER LA PAGE SELON LA LANGUE CHOISIE DE LA DROITE VERS LA GAUCHE ET VICE VERSA :**

Dans la page exemple.blade taper le code suivant :

@if (App::getLocale() == 'en')

    <!-- Bootstrap Css -->

    <link href="{{ asset('assets/css/bootstrap.min.css') }}" rel="stylesheet" type="text/css" />

    <!-- App Css-->

    <link href="{{ asset('assets/css/app.min.css') }}" rel="stylesheet" type="text/css" />

    <!-- Icons Css -->

    <link href="{{ asset('assets/css/icons.min.css') }}" rel="stylesheet" type="text/css" />

    <!-- custom Css-->

    <link href="{{ asset('assets/css/custom.min.css') }}" rel="stylesheet" type="text/css" />

@else

    <!-- Bootstrap Css -->

    <link href="{{ asset('assets/css/bootstrap-rtl.min.css') }}" rel="stylesheet" type="text/css" />

    <!-- App Css-->

    <link href="{{ asset('assets/css/app-rtl.min.css') }}" rel="stylesheet" type="text/css" />

    <!-- Icons Css -->

    <link href="{{ asset('assets/css/icons.min.css') }}" rel="stylesheet" type="text/css" />

    <!-- custom Css-->

    <link href="{{ asset('assets/css/custom-rtl.min.css') }}" rel="stylesheet" type="text/css" />

@endif

1. **POUR INSTALLER MCAMARA LARAVEL LOCALIZATION TAPER LA COMMANDE SUIVANTE :**

composer require mcamara/laravel-localization

php artisan vendor :publish –provider= "Mcamara\LaravelLocalization\LaravelLocalizationServiceProvider"

1. **CONFIG->LARAVELLOCALIZATION.PHP:**

modifier return ‘supportedLocales’ on décommentant ‘ar’ pour activer la langue arabe et commenter ‘es’ pour désactiver la langue Espagne

 //'he'=> ['name' => 'Hebrew','script' => 'Hebr', 'native' => 'עברית', 'regional' => 'he\_IL'],

 //'ug'=> ['name' => 'Uyghur', 'script' => 'Arab', 'native' => 'ئۇيغۇرچە', 'regional' => 'ug\_CN'],

 //'ur'=> ['name' => 'Urdu','script' => 'Arab', 'native' => 'اردو', 'regional' => 'ur\_PK'],

 'ar'=> ['name' => 'Arabic','script' => 'Arab', 'native' => 'العربية', 'regional' => 'ar\_AE'],

 //'uz-Arab'=> ['name' => 'Uzbek (Arabic)','script' => 'Arab', 'native' => 'اۉزبېک', 'regional' => ''],

 //'tg-Arab'=> ['name' => 'Tajik (Arabic)', 'script' => 'Arab', 'native' => 'تاجیکی', 'regional' => 'tg\_TJ'],

1. **DANS APP/http/KERNEL.PHP DANS LE protected $routeMiddleware ajouter les lignes de code suivantes :**

'localize' => \Mcamara\LaravelLocalization\Middleware\LaravelLocalizationRoutes::class,

        'localizationRedirect' => \Mcamara\LaravelLocalization\Middleware\LaravelLocalizationRedirectFilter::class,

        'localeSessionRedirect' => \Mcamara\LaravelLocalization\Middleware\LocaleSessionRedirect::class,

        'localeCookieRedirect' => \Mcamara\LaravelLocalization\Middleware\LocaleCookieRedirect::class,

        'localeViewPath' => \Mcamara\LaravelLocalization\Middleware\LaravelLocalizationViewPath::class,

1. **DANS LE FICHIER WEB.PHP AJOUTER CE CODE :**

Route ::group(['prefix'=>LaravelLocalization ::setLocale(),

    /\*\* POUR CONSERVER LA DERNIERE LANGUE ACTIVE \*\*/

    'middleware' => ['localeSessionRedirect','localizationRedirect','localeViewPath']

    ], function(){

    /\*\* ADD ALL LOCALIZED ROUTES INSIDE THIS GROUP\*\*/

    Route::get('/',function()

    {

        return view('test');

    });

});

**/\*\* OTHER PAGES THAT SHOULD BE LOCALIZED\*\*/**

1. **DANS LA PAGE PRINCPALE AJOUTER UN LIEN POUR CHANGER LES LANGUES :**

 <ul>

 @foreach (LaravelLocalization::getSupportedLocales() as $localeCode => $properties)

   <li>

   <a rel="alternate" hreflang="{{ $localeCode}}"

href="{{LaravelLocalization::getLocalizedURL($localeCode, null, [], true) }}">

      {{$properties['native']}}

   </a>

   </li>

 @endforeach

 </ul>

1. **DANS LE DOSSIER RESOURCES->DOSSIER LANG->DOSSIER en FAIRE COPIER COLLER ET CHANGE LE NOM AU DOSSIER ar PUIS CREER DANS LES DEUX DOSSIERS UN FICHIER AVEC LE MEME NOM main\_trans.php**/\*\* NOM FICHIER\*//**POUR FAIRE LA TRADUCTION**

**Exemple :**

**Dans le fichier en :**

return [

    'Dashboard'=>'Dashboard',

    'Dashboard\_page'=>'Dashboard',

    'Main\_title' => 'MoraSoft School Management Program',

    'Programname' => 'MoraSoft School Management Program',

    'change\_language'=>'language',

    'Grades'=>'Grades',

    'Grades\_list'=>'Grades List',

    'classes'=>'Classes',

    'List\_classes' => 'List Classes',

    'sections'=>'Sections',

    'List\_sections'=>'List Sections',

    'students'=>'Students',

    'add\_student'=>'Add student',

    'list\_students'=>'list Students',

    'Students\_Promotions'=>'Students Promotions',

    'information\_student'=>'Information Students',

    'Students\_upgrade'=>'Students Upgrade',

    'Graduate\_students'=>'Graduate Students',

    'Teachers'=>'Teachers',

    'List\_Teachers' => 'List Teachers',

    'Parents'=>'Parents',

    'Add\_Parent'=>'Add Parent',

    'List\_Parents'=>'List Parents',

    'Accounts'=>'Accounts',

    'Attendance'=>'Attendance',

    'Exams'=>'Exams',

    'library'=>'Library',

    'Onlineclasses'=>'Online classes',

    'Settings'=>'Settings',

    'Users'=>'Users',

    'Copyright' => 'Copyright',

    'Name\_Programer' => 'SamirGamal MoraSoft All Rights Reserved'

];

**Dans le fichier ar :**

return [

    'Dashboard'=>'الرئيسية',

    'Dashboard\_page'=>'لوحة التحكم',

    'Main\_title'=>'برنامج مورا سوفت لادارة المدارس',

    'Programname' => 'برنامج مورا سوفت لادارة المدارس',

    'change\_language'=>'تغير اللغة',

    'Grades'=>'المراحل الدراسية',

    'Grades\_list'=>'قائمة المراحل الدراسية',

    'classes'=>'الصفوف',

    'List\_classes'=>'قائمة الصفوف الدراسية',

    'sections'=>'الاقسام',

    'List\_sections'=>'قائمة الاقسام الدراسية',

    'students'=>'الطلاب',

    'Student\_information'=>'معلومات الطلاب',

    'add\_student'=>'اضافة طالب جديد',

    'list\_students'=>'قائمة الطلاب',

    'Students\_Promotions'=>'ترقية الطلاب',

    'add\_Promotion'=>'اضافة ترقية جديدة',

    'list\_Promotions'=>'قائمة الترقيات',

    'Graduate\_students'=>'الطلاب المتخرجين',

    'add\_Graduate'=>'اضافة تخرج جديد',

    'list\_Graduate'=>'قائمة التخرجات',

    'Teachers'=>'المعلمين',

    'List\_Teachers' => 'قائمة المعلمين',

    'Parents'=>'اولياء الامور',

    'List\_Parents'=>'قائمة اولياء الامور',

    'Add\_Parent'=>'اضافة ولي امر',

    'Accounts'=>'الحسابات',

    'Attendance'=>'الحضور والغياب',

    'Exams'=>'الامتحانات',

    'library'=>'المكتبة',

    'Onlineclasses'=>'حصص اونلاين',

    'Settings'=>'الاعدادات',

    'Users'=>'المستخدمين',

    'Copyright' => 'جميع الحقوق محفوظة ',

    'Name\_Programer' => 'سمير جمال مورا سوفت'

];

1. **DANS LES PAGES HTML REMPLACER LE MOT A TRADUIRE PAR SON EQUIVALENT COMME SUITVANT :**

{{ trans('main\_trans.Dashboard') }}

/\*\* NOM FICHIER.MOT A TRADUIRE\*//

1. **POUR INSTALLER LE SYSTEME DE L’AUTHENTIFICATION TAPER LES COMMANDES SUIVANTES :**

composer require laravel/ui

php artisan ui vue --auth

npm install

npm run dev

php artisan ui vue --auth

1. **DANS LE FICHIER WEB.PHP COUPER LE LIGNE DU CODE**

Route ::get('/{page}','AdminController@index') ;

**APRES la LIGNE**

Auth::routes();

1. **DANS LE FICHIER .ENV CHANGE LE NOM DE LA BASE DE DONNEE COMME SUIVANT :**

DB\_DATABASE= LE NOME DE LA BASE DONNEE ICI

1. **CREER LA BASE DE DONNEES DANS PHPMYADMIN TYPE MYSQL ET DONNER LE LE MEME NOM QUE DB\_DATABASE**
2. **POUR CREER UNE TABLE DANS LA BASE DE DONNEE TAPER LA COMMANDE SUIVANTE :**

D’abord créer la migration en tapant la commande suivante

php artisan make:migration create\_grades\_table

class CreateGradesTable extends Migration {

    public function up()

    {

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

            $table->id();

            $table->timestamps();

            $table->json('Name')->nullable();

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

             $table->charset = 'utf8';

            $table->collation = 'utf8\_unicode\_ci';

        });

    }

    public function down()

    {

        Schema::drop('grades');

    }

}

après :

soit cliquer sur le bouton run migration dans la page d’erreur sur site

soit taper la commande php artisan migrate

1. **CREATION DU MODEL GRADE EN TAPANT CETTE LIGNE DE COMMANDE :**

php artisan make:model Grade (**Grade en singulier avec le premier lettre en majuscule**)

class Grade extends Model

{

    use HasTranslations;

    public $translatable = ['Name'];

    protected $fillable=['Name','Notes'];

    protected $table = 'grades';

    public $timestamps = true;

    // علاقة المراحل الدراسية لجلب الاقسام المتعلقة بكل مرحلة

    public function Sections()

    {

        return $this->hasMany('App\Models\Section', 'Grade\_id');

    }

}

1. **POUR DESACTIVER LA CREATION D’UN USER AJOUTER CETTE LIGNE DE CODE DANS WEB.PHP**

Auth ::routes(['register'=>false]) ;

1. **POUR CHANGER LA REDIRECTION A LA PAGE LOGIN CHANGER LE CODE COMME SUIVANT :**

* **Au lieu de** :

Route ::get(‘/’, function()

{

return view (‘dashboard’) ;

}) ;

* **Lire** :

Route ::get(‘/’, function()

{

return view (‘auth.login’) ;

}) ;

**POUR ACCEDER A LA PAGE DASHBOARD APRES LOGIN MODIFIER LE WEB.PHP COMME SUIVANT :**

Auth::routes();

Route::group(['middleware'=>['guest']],function(){

    Route ::get('/', function()

    {

        return view ('auth.login') ;

    }) ;

});

Route::group(['prefix'=>LaravelLocalization::setLocale(),

    /\*\* POUR CONSERVER LA DERNIERE LANGUE ACTIVE \*\*/

    'middleware' => ['localeSessionRedirect','localizationRedirect','localeViewPath','auth']

    ], function(){

    Route::get('/dashboard', [App\Http\Controllers\HomeController::class, 'index'])->name('home');

    Route::resource('grades', GradeController::class);

    Route::resource('classes', ClasseController::class);

});

* **Puis dans le fichier GradeController ajouter la fonction suivante:**

public function index()

    {

        return  $this->grades->getAllGrades();

    }

* Dans app->providers->RouteServicesProvider : changer Public const HOME = ‘/home’ ; au

public const HOME = '/dashboard';

1. **POUR EVITER DE REDERIGER VERS LOGIN APRES AUTHENTIFICATION TAPER CE CODE DANS WEB.PHP :**

Route::group(['middleware'=>['guest']],function(){

    Route ::get('/', function()

    {

        return view ('auth.login') ;

    }) ;

});

1. **POUR UTILISER LA TRADUCTION DES DONNEES DE LA BASE DE DONNEES NE MET PAS LA TAILLE DU CHAMP LAISSEZ LE VIDE**

* **Par exemple :**

$table->json('Name')->nullable();

**AU LIEU DE**

$table->string('Name',30) ;

1. **SI ON AJOUTE UN DOSSIER DANS LE DOSSIER CONTROLLER ON DOIT AJOUTER CE CODE POUR MODIFIER LE « NAMESPACE »**

* **Route ::group(**

**[‘namespace’=>’nom dossier’], function(){**

**Route ::resource(‘Grades par exemple’,’GradeController’) ;**

**}) ;**

1. **CONCERNANT TEMPLATE UN CODE POUR LE CHEMIN**

* <script type= ‘’text/javascript’’>var plugin\_path=’{{asset(‘assets/js’)}}/’ ;</script>

1. **CREER UN FICHIER DE TRADUCTION POUR CHAQUE PAGE**
2. **POUR AFFICHER MODAL DE SUPPRESSION :**

<button grade\_id="{{ $grade->id }}" class="deleteGrade btn btn-sm btn-danger delete-item-btn"  data-bs-toggle="modal" data-bs-target="#delete{{ $grade->id }}"data-grade="{{ $grade }}">{{ trans('Grades\_trans.delete\_Grade') }}</button>

1. **POUR AFFICHER MODAL D’AJOUT :**

<button type="button" class="btn btn-success add-btn" data-bs-toggle="modal" id="create-btn" data-bs-target="#showModal">

<I class="ri-add-line align-bottom me-1"></i> {{ trans('Grades\_trans.add\_Grade') }}</button>

1. **POUR DONNER UN NUMERO AU 1°COLONNE DE LA PAGE HTML # AJOUTER CE CODE ENTRE @FOREACH**

 @foreach ($grades as $index => $grade)

 <td>{{ $index = $index + 1 }}</td>

<td>autre td ici</td>

 @endforeach

1. **POUR UTILISER UN MODAL D’EDITION AJOUTER LE CODE SUIVANT AU BOUTON EDIT**

 <button grade\_id="{{ $grade->id }}" class="editGrade btn btn-sm btn-success edit-item-btn" data-bs-toggle="modal" data-bs-target="#edit{{ $grade->id }}" data-grade="{{ $grade }}">

{{ trans('Grades\_trans.edit\_Grade') }}

</button>

1. **POUR CREER UN FICHIER DANS VALIDATION AVEC MESSAGE D’ERREUR ON DOIT CREER UNE CLASSE NOMMER GradeRequest PAR EXEMPLE EN TAPANT LA COMMANDE SUIVANTE :**

* php artisan make:request GradeRequest
* modifier le fichier GradeRequest :

 public function authorize()

    {

        return true;

    }

public function rules()

    {

        return [

             'name.\*' => 'required|unique\_translation:grades,Name,'.$this->id,

         ];

    }

    public function messages()

    {

        return [

            'name.\*.required' => trans('validation.required'),

             'name.\*.unique' => trans('validation.unique'),

        ];

    }

* **POUR INSERER** Use Spatie\Translatable\HasTranslations ;

1. **POUR VALIDER UNIQUE DANS DEUX COLONNES :**

'name.\*' =>UniqueTranslationRule::for('sections','nomSection')->where('classe\_id',$this->classe\_id),

1. **UN ENREGISTREMENT DANS LA BDD ON UTILISE SPATIE/LARAVEL-TRANSLATABLE ON TAPANT LA COMMANDE SUIVANTE:**

* composer require spatie/laravel-translatable
* php artisan vendor ::publish –provider=’’Spatie\Translatable\TranslatableServiceProvider”
* puis on modifie le fichier **translatable.php**qui se trouve dans le dossier config comme suivant :

au lieu de : ‘fallback\_locale’=>**null**,

lire : ‘fallback\_locale’=>**’en’**,

1. **DANS CHAQUE MODEL QU’ON DESIRE TRADUIRE ON AJOUTE CES LIGNES DE CODE :**

class Grade extends Model

{

    use HasTranslations;

    public $translatable = ['Name'];

    protected $fillable=['Name','Notes'];

…………………….. autre code ici

}

1. **POUR UTILISER LE DESIGN PATTERN ON DOIT CREER L’INTERFACE GradeRepositoryInterface.php**

<?php

namespace App\Interfaces;

Interface GradeRepositoryInterface{

    // get all Grades

    public function getAllGrades();

    // StoreGrades

    public function StoreGrades($request);

    // UpdateGrades

    public function UpdateGrades($request,$id);

    // DeleteGrades

    public function DeleteGrades($request,$id);

}

1. **PUIS CREER REPOSITORY GradeRepository**

<?php

namespace App\Repositories;

use App\Models\Grade;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Request;

use App\Interfaces\GradeRepositoryInterface;

class GradeRepository implements GradeRepositoryInterface{

    // get all Grades

    public function getAllGrades(){

         $grades = Grade::all();

        return view('pages.grades.index',compact('grades'));

}

    // StoreGrades

    public function StoreGrades($request){

        $gradeAr=str\_replace('"','',json\_encode($request->name));

        $gradeFr=json\_encode($request->Name\_en);

    try {

            $validated = $request->validated();

            $Grade = new Grade();

            $Grade->Name =['en' => $request->name['en'], 'ar' => $request->name['ar']];

            $Grade->Notes = $request->Notes;

            $Grade->save();

           flash()->addSuccess(trans('messages.success'));

            return redirect()->route('grades.index');

        }

        catch (\Exception $e){

            return redirect()->back()->withErrors(['error' => $e->getMessage()]);

        }

    }

    // UpdateGrades

    public function UpdateGrades($request,$id){

        try {

       $validated = $request->validated();

       $Grade = Grade::findOrFail($request->id);

       $Grade->update([

        $Grade->Name =['en' => $request->name['en'], 'ar' => $request->name['ar']],

         $Grade->Notes = $request->Notes,

       ]);

           flash()->addSuccess(trans('messages.Update'));

       return redirect()->route('grades.index');

   }

   catch

   (\Exception $e) {

       return redirect()->back()->withErrors(['error' => $e->getMessage()]);

   }

    }

    // DeleteGrades

    public function DeleteGrades($request ,$id){

        // $MyClass\_id = Classroom::where('Grade\_id',$request->id)->pluck('Grade\_id');

    //   if($MyClass\_id->count() == 0){

          $grade = Grade::findOrFail($request->id)->delete();

          flash()->addError(trans('messages.Delete'));

          return redirect()->route('grades.index');

    //   }

    //   else{

          flash()->addError(trans('messages.Delete'));

          return redirect()->route('grades.delete\_error);

    //   }

    }

}

1. **DANS LE CONTROLLER ON AJOUTE LE CODE SUIVANT :**

<?php

namespace App\Http\Controllers\Grades;

use App\Models\Grade;

use Illuminate\Http\Request;

use App\Http\Requests\StoreGrades;

use App\Http\Controllers\Controller;

use App\Interfaces\GradeRepositoryInterface;

class GradeController extends Controller

{

    protected $grades;

    public function \_\_construct(GradeRepositoryInterface $grades)

    {

        $this->grades = $grades;

    }

    public function index()

    {

        return  $this->grades->getAllGrades();

    }

    public function create()

    {

        //

    }

    public function store(StoreGrades $request)

    {

        return  $this->grades->StoreGrades($request);

    }

    public function show($id)

    {

        //

    }

    public function update(StoreGrades $request,$id)

    {

        return  $this->grades->UpdateGrades($request,$id);

    }

    public function destroy(Request $request,$id)

    {

        return  $this->grades->DeleteGrades($request,$id);

    }

}

1. **INSTALLATION PACKAGE php\_flasher :**

* composer require php-flasher/flasher-laravel
* ajouter ce code dans le controller pour afficher un message de success

flash()->addSuccess('The action was completed successfully.');

* ajouter ce code dans le controller pour afficher un message d’erreur

flash()->addError('An error occurred while creating your account.');

* ajouter ce code dans le controller pour afficher un message d’information

flash()->addInfo('Your account has been unlocked and a confirmation email has been sent.');

* ajouter ce code dans le controller pour afficher un message d’avertissement

flash()->addWarning('Your question may not have been submitted.');

1. **UTILISATION DU TRY CATCH :**

* **try {**

CODE DE SUCCESS ICI

**}**

**catch(\Exception $e)**

**{**

CODE EN CAS D’ERREUR

**}**

1. **DANS <FORM> ON DOIT AJOUTER LA METHODE PATCH EN CAS DE MODIFICATION :**

 {{ method\_field('patch') }}

 @csrf

1. **POUR REMPLIR LES CHAMPS DE SAISIE EN CAS DE MODIFICATION :**

<input type="text" class="form-control" autocomplete="off" value="{{ $grade->getTranslation('Name', 'ar') }}"

 name="name[ar]" id="Name">

1. **DANS <FORM> ON DOIT AJOUTER LA METHODE DELETE EN CAS DE SUPPRESSIO :**

{{ method\_field('Delete') }}

@csrf

1. **POUR UTILISER PACKAGE QUI PERMET DE VERIFIER SI UNE VALEUR UNIQUE A ETE DEJA AJOUTE OU NON :**

* composer require codezero/laravel-unique-translation
* dans REQUEST ajouter le code suivant:

 'name.\*' => 'required|unique\_translation:grades,Name,'.$this->id,

1. **POUR ACTUALISER L’APPLICATION:**

* php artisan view:clear

1. **EN CAS DE CREATION D’UN DOSSIER Grades DANS LE DOSSIER CONTROLLER ON DOIT AJOUTER LE NAMESPACE DANS LE FICHIER WEB.PHP :**

* Route ::group([‘namespace’=>’Grades’],function(){

Route::resource(‘Grades’,’GradeController’);

});

1. **POUR SUPPRIMER PAR EXEMPLE LE GRADE VERIFIER S’IL CONTIEN DES CLASSES :**

 $MyClass\_id = Classroom::where('Grade\_id',$request->id)->pluck('Grade\_id');

      if($MyClass\_id->count() == 0){

//CODE DE SUPPRESSION ICI

}

      else{

           flash()->addSuccess(trans('Grades\_trans.delete\_Grade\_Error'));

          return redirect()->route('Grades.index');

      }

1. **POUR INSERER MULTIPLE ROW :**

<input type="text" class="form-control bg-light border-0" id="nameClasse-1" name="name[0][ar]" placeholder="{{ trans('Classes\_trans.nameClasse') }}" required />

1. **SCRIPT POUR AJOUTER INPUT POUR INSERTION MULTIPLE :**

$('#btnRepeater').click(function () {

// POUR RECUPERER LA PREMIERE LIGNE QUI CONTIEN LES CHAMPS DE SAISIE AVEC SELECT REMPLI

    let rowRepeater = $("#1").html();

// POUR COMPTER LE NOMBRE LIGNE SUR HTML

    let nbr=$('.classe').length;

// VARIABLE POUR L’UTILISER DANS LE NOM DE L’ATTRIBUTE TYPE ARRAY OBJET

    let i = nbr;

    // INCREMENTER LE NOMBRE DE LIGNE A CHAQUE CLIQUE D’AJOUT D’UNE LIGNE ;

nbr++;

// POUR AJOUTER NOUVEL LIGNE DES CHAMPS APRES LA DERNIERE LIGNE

    $("#newlink").append("<tr id='"+nbr+"' class='classe'>"+rowRepeater+"</tr>");

// POUR MODIFIER LE TEXT DE LA BALISE « TH » DE LA NOUVEL LIGNE AJOUTEE

    $("#" + nbr).children("th").text(nbr);

// POUR MODIFIER L’ID DE LA BALISE « TH » DE LA NOUVEL LIGNE AJOUTEE

    $("#" + nbr).children("th").attr("id", "nbrClassement" + nbr);

// POUR MODIFIER L’ID DE LA BALISE « INPUT AR» DE LA NOUVEL LIGNE AJOUTEE

    $("#" + nbr).find("input").attr("id", "nameClasse-" + nbr);

// POUR MODIFIER LE NAME DE LA BALISE « INPUT AR » DE LA NOUVEL LIGNE AJOUTEE

    $("#" + nbr).find("#nameClasse-"+nbr).attr("name","name["+i+"][ar]");

// POUR MODIFIER L’ID DE LA BALISE « INPUT EN» DE LA NOUVEL LIGNE AJOUTEE

    $("#" + nbr).find("input").last().attr("id","nameClasse-en-"+nbr);

// POUR MODIFIER LE NAME DE LA BALISE « INPUT EN » DE LA NOUVEL LIGNE AJOUTEE

    $("#" + nbr).find("#nameClasse-en-"+nbr).attr("name","name["+i+"][en]");

// POUR MODIFIER L’ID DE LA BALISE « SELECT » DE LA NOUVEL LIGNE AJOUTEE

    $("#" + nbr).find("select").attr("id","nameGrade-"+nbr);

})

1. **POUR SUPPRIMER UNE LIGNE AJOUTER :**

$("#tbleClasse").on('click', '.btnDelete', function () {

// POUR COMPER LE NOMBRE DE LIGNE AYANT LA CLASS= « CLASSE »

    let nbr=$('.classe').length;

// POUR VERIFIER SI LE NOMBRE DE LIGNE SUPERIEUR DE 1 POUR NE PAS SUPPRIMER TOUTES LES LIGNES

    if (nbr > 1) {

// SUPPRIMER LA LIGNE QUI ENVLOPPE CETTE BOUTON QU’ON A CLIQUE

    $(this).closest('tr').remove();

    }

});

1. **POUR AJOUTER PLUSIEUR LIGNES AJOUTER CE CODE A LA METHODE STORE DANS LE CONTROLLER**

 public function Store($request){

    try {

            $validated = $request->validated();

             foreach ($request->name as $index => $listClasse) {

            $classe = new Classe();

            $classe->nomClasse =['en' => $request->name[$index]['en'], 'ar' => $request->name[$index]['ar']];

            $classe->grade\_id = $request->grade\_id[$index];

            $classe->save();

             }

           flash()->addSuccess(trans('messages.success'));

            return redirect()->route('classes.index');

        }

        catch (\Exception $e){

            return redirect()->back()->withErrors(['error' => $e->getMessage()]);

        }

    }

1. **POUR VALIDER CHAMP AVEC ARRAY D’OBJET :**

return [

            'name.\*.\*' => 'required|unique\_translation:classes,nomClasse,'.$this->id,

        ];

1. **POUR AFFICHER MESSAGE DE VALIDATION ARRAY D’OBJET :**

 public function messages()

    {

        return [

            'name.\*.\*.required' => trans('validation.required'),

             'name.\*.\*.unique' => trans('validation.unique'),

        ];

    }

1. **EN CAS D’UTILISATION MODAL EDIT QUI A LE MEME INPUT QUE LE MODAL D’AJOUT**

$('#btnRepeater').click(function () {

    let rowRepeater = $('#showModal').find("#1").html();

    let nbr=$('#showModal').find('.classe').length;

    let i = nbr;

    nbr++;

    $('#showModal').find("#newlink").append("<tr id='"+nbr+"' class='classe'>"+rowRepeater+"</tr>");

    $('#showModal').find("#" + nbr).children("th").text(nbr);

    $('#showModal').find("#" + nbr).children("th").attr("id", "nbrClassement" + nbr);

    $('#showModal').find("#" + nbr).find("input").attr("id", "nameClasse-" + nbr);

    $('#showModal').find("#" + nbr).find("#nameClasse-"+nbr).attr("name","name["+i+"][ar]");

    $('#showModal').find("#" + nbr).find("input").last().attr("id","nameClasse-en-"+nbr);

    $('#showModal').find("#" + nbr).find("#nameClasse-en-"+nbr).attr("name","name["+i+"][en]");

    $('#showModal').find("#" + nbr).find("select").attr("id","nameGrade-"+nbr);

})

$('#showModal').find("#tbleClasse").on('click', '.btnDelete', function () {

    let nbr = $('#showModal').find('.classe').length;

        let i = nbr-1;

    if (nbr > 1) {

        $(this).closest('tr').remove();

    }

    $('#showModal').find("#" + nbr).find("#nameClasse-" + nbr).attr("name", "name[" + i + "][ar]");

    $('#showModal').find("#" + nbr).find("#nameClasse-en-" + nbr).attr("name", "name[" + i + "][en]");

});

**Dans controller**

 // UpdateClasses

    public function UpdateClasses($request,$id){

        try {

       $validated = $request->validated();

       $classe = Classe::findOrFail($request->id);

       $classe->update([

        $classe->nomClasse =['en' => $request->name[0]['en'], 'ar' => $request->name[0]['ar']],

         $classe->grade\_id = $request->grade\_id[0],

       ]);

           flash()->addSuccess(trans('messages.Update'));

       return redirect()->route('classes.index');

   }

   catch

   (\Exception $e) {

       return redirect()->back()->withErrors(['error' => $e->getMessage()]);

   }

    }

**Bouton edit**

<button classe\_id="{{ $classe->id }}" class="editClasse btn btn-sm btn-success edit-item-btn"  data-bs-toggle="modal" data-bs-target="#edit{{ $classe->id }}"data-classe="{{ $classe }}">{{ trans('Classes\_trans.Edit') }}</button>

**Modal d’edition**

 {{-- :::::::::::::::::::::::::::debut modal edit::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --}}

 <div class="modal fade" id="edit{{ $classe->id }}" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

 <div class="modal-dialog modal-xl modal-dialog-centered">

 <div class="modal-content">

 <div class="modal-header bg-light p-3">

<h5 class="modal-title" id="exampleModalLabel"></h5>

 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close" id="close-modal"></button>

 </div>

<div class="p-3">

<form class=" row mb-30" action="{{ route('classes.update', $classe->id) }}" method="POST">

{{ method\_field('patch') }}

  @csrf

  <input id="id" type="hidden" name="id"  class="form-control" value="{{ $classe->id }}">

<div class="card-body">

 <div class="table-responsive" data-repeater-list="List\_Classes">

 <table id="tbleClasse" class="invoice-table table table-borderless table-nowrap mb-0">

 <thead class="align-middle">

 <tr class="table-active">

 <th scope="col" style="width: 50px;">#</th>

 <th scope="col"> {{ trans('Classes\_trans.nameClasse') }}</th>

 <th scope="col"> {{ trans('Classes\_trans.nameClasse\_en') }}</th>

 <th scope="col" class="text-end"> {{ trans('Classes\_trans.name\_Grade') }} </th>

  </tr>

</thead>

<tbody id="newlink" data-repeater-list="List\_Classes">

 <tr id="1" class="classe">\*

<th scope="row"   id="nbrClassement1" class="classe-id">1</th>

<td class="text-start">

<div class="mb-2">

<input type="text" class="form-control bg-light border-0" id="nameClasse-1" value="{{ $classe->

getTranslation('nomClasse', 'ar') }}"  name="name[0][ar]"  placeholder="{{ trans('Classes\_trans.nameClasse') }}" required />

<div class="invalid-feedback"> Please enter a product name </div>

</div>\*

/td>

<td class="text-start">

<div class="mb-2">

<input type="text" class="form-control bg-light border-0" id="nameClasse-en-1"  name="name[0][en]" value="{{ $classe->getTranslation('nomClasse', 'en') }}"

   placeholder="{{ trans('Classes\_trans.nameClasse\_en') }}"required />

<div class="invalid-feedback">  Please enter a product name </div>

</div>

</td>

<td class="text-end">

<div>

<select id="nameGrade-1" name="grade\_id[]" class="form-select form-control mb-3 bg-light nameGrade" aria-label="Default select example">

<option selected value="{{ $classe->grades->id }}">{{ $classe->grades->Name }}</option>

@foreach ($grades as $grade)

<option value="{{ $grade->id }}">{{ $grade->Name }}</option>

@endforeach

</select>

</div>

</td>

</tr>

</tbody>

</table>

<!--end table-->

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">{{ trans('Grades\_trans.Close') }}</button>

<button type="submit" class="btn btn-success">{{ trans('Grades\_trans.submit') }}</button>

</div>

</div>

</form>

</div>

</div>

</div>

</div>

{{-- :::::::::::::::::::::::::::::fin model edit:::::::::::::::::::::::::::::::::::::: --}}

1. **POUR SUPPRIMER PLUSIEURS LIGNES A LA FOIS EN UTILISER UN CHECKBOX :**

**Script de suppression**

$(function() {

        $("#btn\_delete\_all").click(function() {

            var selected = new Array();

            $("#classeTable input[type=checkbox]:checked").each(function() {

                selected.push(this.value);

            });

            if (selected.length > 0) {

                $('#delete\_all').modal('show')

                $('input[id="delete\_all\_id"]').val(selected);

            }

        });

    });

**Modal de suppression**

 <!-- :::::::::::::::::::::::::::multi delete\_modal\_classe ::::::::::::::::::::::-->

    <div id="delete\_all" class="modal fade zoomIn" tabindex="-1" aria-labelledby="zoomInModalLabel" aria-hidden="true"

        style="display: none;">

        <div class="modal-dialog modal-dialog-centered">

            <div class="modal-content">

                <div class="modal-header">

                    <h5 class="modal-title" id="zoomInModalLabel">

                        {{ trans('Classes\_trans.delete\_class') }}

                    </h5>

                    <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

                </div>

                <form action="{{ route('delete\_all') }}" method="post">

                    {{-- {{ method\_field('Delete') }} --}}

                    @csrf

                    <div class="modal-body">

                        {{ trans('Grades\_trans.Warning\_Grade') }}

                        <input class="text" type="hidden" id="delete\_all\_id" name="delete\_all\_id" value=''>

                    </div>

                    <div class="modal-footer">

                        <button type="button" class="btn btn-success"

                            data-bs-dismiss="modal">{{ trans('Classes\_trans.Close') }}</button>

                        <button type="submit" class="btn btn-danger ">{{ trans('Classes\_trans.Delete') }}

                        </button>

                    </div>

                </form>

            </div><!-- /.modal-content -->

        </div><!-- /.modal-dialog -->

    </div><!-- /.modal -->

**Bouton supprimer**

<button classe\_id="{{ $classe->id }}" class="deleteClasse btn btn-sm btn-danger delete-item-btn" data-bs-toggle="modal" data-bs-target="#delete{{ $classe->id }}"data-classe="{{ $classe }}">{{ trans('Classes\_trans.Delete') }}</button>

**Dans controller**

 public function delete\_all($request)

    {

        $delete\_all\_id = explode(",", $request->delete\_all\_id);

        Classe::whereIn('id', $delete\_all\_id)->Delete();

        flash()->addError(trans('messages.Delete'));

        return redirect()->route('classes.index');

    }

1. **CREER UN SEEDER**

**Ligne de commande**

**php artisan make:seeder NationalitieTableSeeder**

**dans la classe seeder**

public function run()

    {

        DB::table('nationalities')->delete();

        $nationals = [

            [

                'en'=> 'Afghan',

                'ar'=> 'أفغانستاني'

            ],

            [

                'en'=> 'Albanian',

                'ar'=> 'ألباني'

            ],

            [

                'en'=> 'Aland Islander',

                'ar'=> 'آلاندي'

            ]

];

foreach ($nationals as $n) {

    Nationalitie::create(['name' => $n]);

  }

}

**pour executer seeder :**

**soit Ligne de commande**

**php artisan db:seed --class=NationalitieTableSeeder**

**soit dans classe DatabaseSeeder**

 public function run()

    {

        $this->call(NationalitiesTableSeeder::class);

    }

**Pui taper ligne de commande**

**php artisan db:seed**

**dans model**

class Nationalitie extends Model

{

    use HasFactory;

     use HasTranslations;

    public $translatable = ['name'];

    protected $fillable =['name'];

}

1. [**INSTALL LIVEWIRE**](https://laravel-livewire.com/docs/2.x/quickstart#install-livewire)

**composer require livewire/livewire**

1. **CREATION CONTROLLER AND VIEW LIVEWIRE**

**php artisan make:livewire Add\_Parent**

1. **CREATION COMPONENT PAR EXEMPLE ‘add-parent’**

<div>

    <h1>livewire page</h1>

</div>

1. **AJOUTER COMPONENT DANS LA PAGE**

<div class="table-responsive table-card mt-3 mb-1">

@livewire('add-parent')

</div>

1. **AJOUTER SCRIPT DE LIVEWIRE**

    @livewireScripts()

1. **DANS CLASS AddParent DECLARER TOUTES LES VARIABLES UTILISES DANS LA PAGE HTML**

public $currentStep=1;

1. **DANS INPUT UTILISER wire :model AU LIEU DE name**
2. **LIER VARIABLE DU CONTROLLER AVEC INPUT DANS BLADE.PHP**
3. **POUR LES PHOTOS ET FICHIER ON CREE MODEL « ATTACHMENT » AVEC ATTRIBUTS NAME\_FILE ET CLE ETRANGER D’AUTRE TABLE :**

class ParentAttachement extends Model

{

    use HasFactory;

    protected $fillable=['file\_name','parent\_id'];

}

1. **DANS LE CONTROLLER AJOUTER use WithFileUploads POUR UPLOADER PHOTOS**

use WithFileUploads;

    public $successMessage = '';

    public $catchError;

    public $currentStep=1,

    $photos,

1. **DANS CONFIG PUIS DANS filesystems AJOUTER LE CHEMIN DE SAUVEGARDE DE LA PHOTOS**

'parent\_attachements' => [

            'driver' => 'local',

            'root' => storage\_path('app/parent\_attachements'),

        ],

1. **AJOUTER A LA BALISE FORM L’ATTRIBUT enctype="multipart/form-data"**

 <form wire:submit.prevent="submitForm" method="POST" enctype="multipart/form-data"

1. **DANS LA METHODE STORE AJOUTER LE CODE SUIVANT :**

if (!empty($this->photos)){

  foreach ($this->photos as $photo)   
$photo->storeAs($this->National\_ID\_Father, $photo->getClientOriginalName(), $disk = 'parent\_attachements');

                    ParentAttachement::create([

                        'file\_name' => $photo->getClientOriginalName(),

                        'parent\_id' => MyParent::latest()->first()->id,

                    ]);

                }

            }

1. **POUR AFFICHER TABLE OU FORMULAIRE D’AJOUR SUR LA MEME PAGE PROCEDER COMME SUIVANT :**

créer variable le nomme

Public $show\_table = true,

Créer une méthode qui cache table et affiche formulaire d’ajout

//methode qui cache table et affiche form ajout

    public function showformadd(){

        $this->show\_table = false;

    }

Puis dans la page blade créer un bouton et mettre une condition

 <button type="button" class="btn btn-success add-btn" wire:click="showformadd" id="create-btn"><i class="ri-add-line align-bottom me-1"></i> {{ trans('Parent\_trans.add\_parent') }}</button>

@if ($show\_table)

        @include('livewire.tableParents')

@else

        @include('livewire.formAjout')

@end if

1. **POUR SUPPRIMER AVEC LIVEWIRE :**

**Dans le controller**

 public function delete($id){

        MyParent::findOrFail($id)->delete();

        return redirect()->to('/add\_parent');

    }

**Dans blade**

<button class="deleteGrade btn btn-sm btn-danger delete-item-btn"  wire:click="delete({{ $myParent->id }})">{{ trans('Parent\_trans.Delete') }}</button>

1. **POUR EDITER DANS UNE MEME PAGE AVEC LE MEME FORMULAIRE :**

**Dans le controller**

   //methode edit

    public function edit($id)

    {

        $this->show\_table = false;

        $this->updateMode = true;

        $My\_Parent = MyParent::where('id',$id)->first();

        $this->Parent\_id = $id;

        $this->Email = $My\_Parent->email;

        $this->Password = $My\_Parent->password;

        $this->Name\_Father = $My\_Parent->getTranslation('name\_Father', 'ar');

        $this->Name\_Father\_en = $My\_Parent->getTranslation('name\_Father', 'en');

        $this->Job\_Father = $My\_Parent->getTranslation('job\_Father', 'ar');;

        $this->Job\_Father\_en = $My\_Parent->getTranslation('job\_Father', 'en');

        $this->National\_ID\_Father =$My\_Parent->cin\_Father;

        $this->Passport\_ID\_Father = $My\_Parent->passeport\_ID\_Father;

        $this->Phone\_Father = $My\_Parent->phone\_Father;

        $this->Nationality\_Father\_id = $My\_Parent->nationality\_Father\_id;

        $this->Blood\_Type\_Father\_id = $My\_Parent->blood\_Father\_id;

        $this->Address\_Father =$My\_Parent->addresse\_Father;

        $this->Religion\_Father\_id =$My\_Parent->religion\_Father\_id;

        $this->Name\_Mother = $My\_Parent->getTranslation('name\_Mother', 'ar');

        $this->Name\_Mother\_en = $My\_Parent->getTranslation('name\_Father', 'en');

        $this->Job\_Mother = $My\_Parent->getTranslation('job\_Mother', 'ar');;

        $this->Job\_Mother\_en = $My\_Parent->getTranslation('job\_Mother', 'en');

        $this->National\_ID\_Mother =$My\_Parent->cin\_Mother;

        $this->Passport\_ID\_Mother = $My\_Parent->passeport\_ID\_Mother;

        $this->Phone\_Mother = $My\_Parent->phone\_Mother;

        $this->Nationality\_Mother\_id = $My\_Parent->nationality\_Mother\_id;

        $this->Blood\_Type\_Mother\_id = $My\_Parent->blood\_Mother\_id;

        $this->Address\_Mother =$My\_Parent->addresse\_Mother;

        $this->Religion\_Mother\_id =$My\_Parent->religion\_Mother\_id;

    }

    public function submitForm\_edit(){

        if ($this->Parent\_id){

            $parent = MyParent::find($this->Parent\_id);

            $parent->update([

            'email' => $this->Email,

            'name\_Father' => ['en' => $this->Name\_Father\_en, 'ar' => $this->Name\_Father],

            'passeport\_ID\_Father' => $this->Passport\_ID\_Father,

            'cin\_Father' => $this->National\_ID\_Father,

            'phone\_Father' => $this->Phone\_Father,

            'job\_Father' => ['en' => $this->Job\_Father\_en, 'ar' => $this->Job\_Father],

            'nationality\_Father\_id' => $this->Nationality\_Father\_id,

            'blood\_Father\_id' => $this->Blood\_Type\_Father\_id,

            'religion\_Father\_id' => $this->Religion\_Father\_id,

            'addresse\_Father' => $this->Address\_Father,

            // Mother\_INPUTS

            'name\_Mother' => ['en' => $this->Name\_Mother\_en, 'ar' => $this->Name\_Mother],

            'cin\_Mother' => $this->National\_ID\_Mother,

            'passeport\_ID\_Mother' => $this->Passport\_ID\_Mother,

            'phone\_Mother' => $this->Phone\_Mother,

            'job\_Mother' => ['en' => $this->Job\_Mother\_en, 'ar' => $this->Job\_Mother],

            'nationality\_Mother\_id' => $this->Nationality\_Mother\_id,

            'blood\_Mother\_id' => $this->Blood\_Type\_Mother\_id,

            'religion\_Mother\_id' => $this->Religion\_Mother\_id,

            'addresse\_Mother' => $this->Address\_Mother,

            ]);

        }

//firstStepSubmit

    public function firstStepSubmit\_edit()

    {

        $this->updateMode = true;

        $this->currentStep = 2;

    }

        //secondStepSubmit\_edit

    public function secondStepSubmit\_edit()

    {

        $this->updateMode = true;

        $this->currentStep = 3;

    }

    //back

    public function back($step)

    {

        $this->currentStep = $step;

    }

**Dans blade**

<button class="editGrade btn btn-sm btn-success edit-item-btn" data-bs-toggle="modal" wire:click="edit({{ $myParent->id }})">{{ trans('Parent\_trans.Edit') }}</button>

 @if ($updateMode)

      <button class="btn btn-success btn-sm nextBtn btn-lg pull-right" wire:click="firstStepSubmit\_edit" type="button">{{ trans('Parent\_trans.Next') }}

</button>

@else

<button class="btn btn-success btn-sm nextBtn btn-lg pull-right" wire:click="firstStepSubmit" type="button">{{ trans('Parent\_trans.Next') }}

</button>

@endif

 @if ($updateMode)

<button wire:click="submitForm\_edit" class="btn btn-success btn-sm btn-lg pull-right"

type="button">{{ trans('Parent\_trans.Finish') }}</button>

@else

<button wire:click="submitForm" class="btn btn-success btn-sm btn-lg pull-right"

type="button">{{ trans('Parent\_trans.Finish') }}</button>

@endif

1. **RELATION MANY TO MANY ENTRE TEACHER ET SECTION**

**DANS MIGRATION**

Schema::create('teacher\_section', function (Blueprint $table) {

$table->id();

$table->unsignedBigInteger('teacher\_id');

$table->unsignedBigInteger('section\_id');

            // foreign keys

$table->foreign('teacher\_id')->references('id')->on('teachers')->onDelete('cascade');

$table->foreign('section\_id')->references('id')->on('sections')->onDelete('cascade');

});

**DANS MODEL TEACHER**

// علاقة المعلمين مع الاقسام

    public function sections()

    {

        return $this->belongsToMany('App\Models\Section','teacher\_section');

    }

**DANS MODEL SECTION**

// علاقة الاقسام مع المعلمين

    public function teachers()

    {

        return $this->belongsToMany('App\Models\Teacher','teacher\_section');

    }

1. **POUR UTILISER SELECT2 ON DOIT AJOUTER LE SCRIPT SUIVANT**

$(document).ready(function() {

            $('#teacher\_id').select2();

            $('#teacher\_id\_edit').select2({

            });

});

**Si on a plusieurs modal contient select2 il faut ajout a chaque id un numéro**

<select multiple id="teacher\_id\_edit{{ $section->id }}"

**On cas de l’ajout**

<select multiple id="teacher\_id" name="teacher\_id[]"

class="form-select js-example-basic-multiple form-control mb-3 bg-light nameTeacher" aria-label="Default select example">

@foreach ($listeteachers as $teacher)

<option value="{{ $teacher->id }}">{{ $teacher->name }}

</option>

@endforeach

</select>

**On cas d’édition**

<select multiple id="teacher\_id\_edit{{ $section->id }}" name="teacher\_id[]" class="form-select js-example-basic-multiple form-control mb-3 bg-light nameTeacher" aria-label="Default select example">

@foreach ($section->teachers as $teacher)

<option selected value="{{ $teacher['id'] }}">{{ $teacher['name'] }}</option>

@endforeach

@foreach ($listeteachers as $teacher)

<option value="{{ $teacher->id }}">{{ $teacher->name }}

</option>

@endforeach

</select>

1. **AJOUT DANS PIVOT TABLE DANS LA METHOD STORE AJOUTER CE CODE**

$section->teachers()->attach($request->teacher\_id);

1. **MODIFICATION DANS PIVOT TABLE DANS LA METHOD UPDATE AJOUTER CE CODE**

if (isset($request->teacher\_id)) {

$section->teachers()->sync($request->teacher\_id);

} else {

$section->teachers()->sync(array());

}

1. **VALIDATION UNIQUE COLONNE AVEC WHERE CLAUSE**

'name.\*.ar' => Rule::unique('classes','nomClasse->ar')->where(fn ($query) => $query->where('grade\_id',$this->grade\_id)->whereNot('id',$this->id)),

            'name.\*.en' => Rule::unique('classes','nomClasse->en')->where(fn ($query) => $query->where('grade\_id',$this->grade\_id)->whereNot('id',$this->id)),

1. **AJOUTER UN ATTRIBUT A CHAQUE BUTTON AVEC ID OBJET EXEMPLE ATTRIBUT student\_id**

 <button id="btnDelete" student\_id="{{ $student->id }}" class="deleteStudent btn btn-sm btn-danger delete-item-btn"

data-bs-toggle="modal" data-bs-target="#delete" data-student="{{ $student }}">{{ trans('Students\_trans.Delete') }}</button>

1. **RECUPERER VALEUR ATTRIBUT student id AVEC JQUERY ET LE METTRE DANS UN INPUT**

$('.deleteStudent').on('click', function() {

            console.log('hello');

            console.log($(this).attr('student\_id'));

            $('#id').val($(this).attr('student\_id'));

})

1. **RELATION POLYMORPHIC ON UTLISE UNE SEULE TABLE IMAGES POUR LA RELIER AVEC D’AUTRE TABLE UTILISANT LE MEME RENSEIGNEMENT DE CETTE TABLE**

**Migration**

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

            $table->id();

            $table->string('filename');

            $table->integer('imageable\_id');

            $table->string('imageable\_type');

            $table->timestamps();

        });

**Model Image**

public function imageable()

    {

        return $this->morphTo();

    }

**Model Student avec imageable et le nom de la relation dans le model Image**

public function images()

    {

        return $this->morphMany(Image::class, 'imageable');

    }

**Model MyParent avec imageable et le nom de la relation dans le model Image**

public function images()

    {

        return $this->morphMany(Image::class, 'imageable');

    }

**DANS FILESYSTEMS.PHP**

'upload\_attachments' => [

'driver' => 'local',

'root' => public\_path('/'),

'url' => env('APP\_URL').'/storage',

'visibility' => 'public',

],

**DANS CONTROLLER**

   // insert img

            if($request->hasfile('photos'))

            {

                foreach($request->file('photos') as $file)

                {

                    $name = $file->getClientOriginalName();

                    $file->storeAs('attachments/students/'.$student->name, $file->getClientOriginalName(),'upload\_attachments');

                    // insert in image\_table

                    $images= new Image();

                    $images->filename=$name;

                    $images->imageable\_id= $student->id;

                    $images->imageable\_type = 'App\Models\Student';

                    $images->save();

                }

            }

**EN CAS D’INSERTION DANS DEUX TABLES IL FAUT UTILISER TRANSACTION**

DB::beginTransaction();

//………………………….

DB::commit();

// EN CAS D’ERREUR

DB::rollback();

1. **POUR UTILISER FORMAT DATE HUMAIN**

{{ $attachment->created\_at->diffForHumans() }}

1. **POUR TELECHARGER UN FICHIER SOIT IMAGE SOIT AUTRES**

 public function Download\_attachment($studentsname, $filename)

    {

        return response()->download(public\_path('attachments/students/'.$studentsname.'/'.$filename));

    }

1. **POUR SUPPRIMER UN FICHIER DE LA BASE DE DONNE ET DU DOSSIER**

public function Delete\_attachment($request)

    {

        // Delete img in server disk

        Storage::disk('upload\_attachments')->delete('attachments/students/'.$request->student\_name.'/'.$request->filename);

        // Delete in data

        Image::where('id',$request->id)->where('filename',$request->filename)->delete();

        flash()->addError(trans('messages.Delete'));

        return redirect()->route('students.show',$request->student\_id);

    }

1. **POUR SUPPRIMER UN DOSSIER**

 $path = public\_path('attachments/students/'.$request->studentName);

            if (File::exists($path)) {

                File::deleteDirectory($path);

            }

1. **RETOURNER ERREUR AVEC SESSION**

**DANS CONTROLLER**

  if($students->count() < 1){

                return redirect()->back()->with('error\_promotions',trans('Students\_trans.msgStudent'));

            }

**DANS PAGE BLADE**

@if (Session::has('error\_promotions'))

<div class="alert alert-danger">

<strong>{{ Session::get('error\_promotions') }}</strong>

</div>

@endif

1. **RETOURNER LES ID DANS UN ARRAY POUR MISE AJOUR**

$students = Student::where('grade\_id',$request->grade\_id)->where('classe\_id',$request->classe\_id)->where('section\_id',$request->section\_id)->get();

if($students->count() < 1){

return redirect()->back()->with('error\_promotions',trans('Students\_trans.msgStudent'));

}

// update in table student

foreach ($students as $student){

$ids = explode(',',$student->id);

Student::whereIn('id', $ids)

->update([

'grade\_id'=>$request->to\_grade\_id,

'classe\_id'=>$request->to\_classe\_id,

'section\_id'=>$request->to\_section\_id,

'academic\_year'=>$request->academic\_year\_new,

]);

1. **UTILISER UPDATEORCREATE**

 Promotion::updateOrCreate([

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

                    'from\_grade'=>$request->grade\_id,

                    'from\_Classe'=>$request->classe\_id,

                    'from\_section'=>$request->section\_id,

                    'to\_grade'=>$request->to\_grade\_id,

                    'to\_Classe'=>$request->to\_classe\_id,

                    'to\_section'=>$request->to\_section\_id,

                    'academic\_year'=>$request->academic\_year,

                    'academic\_year\_new'=>$request->academic\_year\_new,

                ]);

1. **CREER UNE SEULE FONCTION POUR SUPPRESSION SOIT TOUT SOIT UN SEUL ENREGISTREMENT**

**dans page blade on creer deux modal**

<!-- delete\_modal\_toutes promotions -->

<div id="delete" class="modal fade zoomIn" tabindex="-1" aria-labelledby="zoomInModalLabel" aria-hidden="true"

    style="display: none;">

    <div class="modal-dialog modal-dialog-centered">

        <div class="modal-content">

            <div class="modal-header">

                <h5 class="modal-title" id="zoomInModalLabel">

                    {{ trans('Students\_trans.Annuler\_Promotion') }} </h5>

                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <form action="{{ route('promotions.destroy', 'test') }}" method="post">

                {{ method\_field('Delete') }}

                @csrf

                <div class="modal-body">

                    <input id="promotionDelete" type="hidden" name="page\_id" class="form-control" value="1">

                    <h5 style="font-family: 'Cairo', sans-serif;"> {{ trans('Students\_trans.confirmation') }}</h5>

                </div>

                <div class="modal-footer">

                    <button type="button" class="btn btn-success"

                        data-bs-dismiss="modal">{{ trans('Students\_trans.Close') }}</button>

                    <button type="submit" class="btn btn-danger ">{{ trans('Students\_trans.Delete') }}

                    </button>

                </div>

            </form>

        </div><!-- /.modal-content -->

    </div><!-- /.modal-dialog -->

</div><!-- /.modal -->

<!-- delete\_modal\_promotion -->

<div id="Delete\_one{{ $promotion->id }}" class="modal fade zoomIn" tabindex="-1" aria-labelledby="zoomInModalLabel"

    aria-hidden="true" style="display: none;">

    <div class="modal-dialog modal-dialog-centered">

        <div class="modal-content">

            <div class="modal-header">

                <h5 class="modal-title" id="zoomInModalLabel">

                    {{ trans('Students\_trans.annulerTransfert') }} </h5>

                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <form action="{{ route('promotions.destroy', 'test') }}" method="post">

                {{ method\_field('Delete') }}

                @csrf

                <div class="modal-body">

                    <input id="promotionDelete" type="hidden" name="promotion\_id" class="form-control"

                        value="{{ $promotion->id }}">

                    <h5 style="font-family: 'Cairo', sans-serif;">

                        {{ trans('Students\_trans.confirmationOne') }} {{ $promotion->student->name }}</h5>

                </div>

                <div class="modal-footer">

                    <button type="button" class="btn btn-success"

                        data-bs-dismiss="modal">{{ trans('Students\_trans.Close') }}</button>

                    <button type="submit" class="btn btn-danger ">{{ trans('Students\_trans.Delete') }}

                    </button>

                </div>

            </form>

        </div><!-- /.modal-content -->

    </div><!-- /.modal-dialog -->

</div><!-- /.modal -->

**Dans controller**

    // DeletePromotions

    public function DeletePromotion($request){

        DB::beginTransaction();

        try {

            // التراجع عن الكل

            if($request->page\_id ==1){

             $promotions = Promotion::all();

             foreach ($promotions as $promotion){

                 //التحديث في جدول الطلاب

                 $ids = explode(',',$promotion->student\_id);

                 student::whereIn('id', $ids)

                 ->update([

                 'grade\_id'=>$promotion->from\_grade,

                 'classe\_id'=>$promotion->from\_Classe,

                 'section\_id'=> $promotion->from\_section,

                 'academic\_year'=>$promotion->academic\_year,

               ]);

                 //حذف جدول الترقيات

                 Promotion::truncate();

             }

                DB::commit();

                flash()->addError(trans('messages.Delete'));

                return redirect()->back();

            }

            else{

                $promotion = Promotion::findorfail($request->promotion\_id);

                Student::where('id', $promotion->student\_id)

                    ->update([

                        'grade\_id'=>$promotion->from\_grade,

                        'classe\_id'=>$promotion->from\_Classe,

                        'section\_id'=> $promotion->from\_section,

                        'academic\_year'=>$promotion->academic\_year,

                    ]);

                Promotion::destroy($request->promotion\_id);

                DB::commit();

                flash()->addError(trans('messages.Delete'));

                return redirect()->back();

            }

        }

        catch (\Exception $e) {

            DB::rollback();

            return redirect()->back()->withErrors(['error' => $e->getMessage()]);

        }

    }

1. **POUR RECUPERER LES ELEVES SUPPRIMER AVEC SOFTDELETE**

 $students = Student::onlyTrashed()->get();

        return view('pages.students.graduateds.management',compact('students'));

1. **POUR ANNULER LA SUPPRESSION SOFTDELETE**

         Student::onlyTrashed()->where('id', $request->student\_id)->first()->restore();

1. **POUR CHANGER UN SELECT INPUT DYNAMIQUEMENT DANS ROW REPEATER**

 $(document).on('change', '.fee-select', function() {

            var selectedFee = $(this).val();

            console.log(selectedFee);

            var amountSelect = $(this).closest('tr').find('.amount-select');

            updateAmountOptions(selectedFee, amountSelect);  });

function updateAmountOptions(selectedFee, amountSelect) {

var langueActive = $('html').attr('lang');

$.ajax({

url: "{{ URL::to('fees') }}/" + selectedFee,

type: "post",

dataType: "json",

success: function(data) {

amountSelect.empty();

$.each(data, function(key, value) {

$(amountSelect).append(

'<option value="' + value.amount + '">' + value.amount + '</option>');

});

}

});

}

1. **POUR AJOUTER UN ARRAY FROM INPUT**

**DANS PAGE BLADE**

<input name="attendences[{{ $student->id }}]" class="leading-tight" type="radio" value="presence">

<input name="attendences[{{ $student->id }}]" class="leading-tight" type="radio" value="absent">

**DANS CONTROLLER**

foreach ($request->attendences as $studentid => $attendence) {

// avec $studentid : key et $attendence :value

{‘1’ :’presence’,’10’ :’presence’} avec presence ou absence selon le radio cocher, 1 et 10 id student

                if( $attendence == 'presence' ) {

                    $attendence\_status = true;

                } else if( $attendence == 'absent' ){

                    $attendence\_status = false;

                }

                Attendance::create([

                    'student\_id'=> $studentid,

                    'grade\_id'=> $request->grade\_id,

                    'classe\_id'=> $request->classe\_id,

                    'section\_id'=> $request->section\_id,

                    'teacher\_id'=> 1,

                    'attendence\_date'=> date('Y-m-d'),

                    'attendence\_status'=> $attendence\_status

                ]);

}

1. **POUR RECUPERER DONNE D’UNE AUTRE TABLE LIEE AVEC UNE TABLE AVEC PIVOTE TABLE**

//get all teachers with ajax

     public function getTeachers($id)

    {

        // return Teacher::select('name', 'id')->where("section\_id",$id)->get();

       $teachers = Teacher::whereHas('sections', function ($query) use ($id) {

        $query->where('section\_id', $id);

    })->get();

return $teachers;

    }

1. **SI ON UTILISE L’INSTANCIATION NEW ON N’EST PAS OBLIGE DE FAIRE $fillables MAIS SI ON UTILISE LE MOT CREATE ON DOIT UTILISER $fillables MAIS CREATE ET PLUS SECURISE QUE NEW**

 $quizze = new Quizze();

 $quizze->name = ['en' => $request->name['en'], 'ar' => $request->name['ar']];

$quizze->grade\_id = $request->grade\_id;

$quizze->classe\_id = $request->classe\_id;

$quizze->section\_id = $request->section\_id;

$quizze->teacher\_id = $request->teacher\_id;

$quizze->save();

// ou on utilise ce code avec $fillables

Quizze::create([

'name'=> ['en' => $request->name['en'], 'ar' => $request->name['ar']],

'grade\_id'=> $request->grade\_id,

'classe\_id'=> $request->classe\_id,

'section\_id'=> $request->section\_id,

'teacher\_id'=> $request->teacher\_id;,

]);

1. **INTEGRATION ZOOM**

**-On vas utiliser un package**

**-Pour installer en tapant cette commande ligne**

composer require macsidigital/laravel-zoom

**-Taper cette commande ligne pour créer un fichier de configuration**

php artisan vendor:publish --provider="MacsiDigital\Zoom\Providers\ZoomServiceProvider"

**-dans le fichier .env on va ajouter ZOOM\_CLIENT\_KEY et ZOOM\_CLIENT\_SECRET**

return [

    'api\_key' => env('ZOOM\_CLIENT\_KEY'),

    'api\_secret' => env('ZOOM\_CLIENT\_SECRET'),

    'base\_url' => 'https://api.zoom.us/v2/',

    'token\_life' => 60 \* 60 \* 24 \* 7, // In seconds, default 1 week

    'authentication\_method' => 'jwt', // Only jwt compatible at present but will add OAuth2

    'max\_api\_calls\_per\_request' => '5' // how many times can we hit the api to return results for an all() request

];

**-Créer un compte sur la plateforme zoom**

**-Se connecter à Zoom**

**-dans l’onglet solutions choisir Marketplace**

**-dans option de choix choisir Build App**

**-choisir jwt en cliquant sur créer**

**-donne le nom de ton application : exemple « afnaneSoft »**

**-copier api key KX44EjH1QeGfWxR71Nm4Nw est le coller dans ZOOM\_CLIENT\_KEY dans .env**

**-copier api secret 0i7oLEnBYr27d7VNLxLIpoD3DaueW3d2rgn6 est le coller dans ZOOM\_CLIENT\_SECRET dans .env**

**-créer migration**

 public function up()

    {

        Schema::create('online\_classes', function (Blueprint $table) {

            $table->id();

            $table->boolean('integration');

            $table->foreignId('grade\_id')->references('id')->on('grades')->onDelete('cascade');

            $table->foreignId('classe\_id')->references('id')->on('classes')->onDelete('cascade');

            $table->foreignId('section\_id')->references('id')->on('sections')->onDelete('cascade');

            //$table->foreignId('user\_id')->references('id')->on('users')->onDelete('cascade');

            $table->string('created\_by');

            $table->string('meeting\_id');

            $table->string('topic');

            $table->dateTime('start\_at');

            $table->integer('duration')->comment('minutes');

            $table->string('password')->comment('meeting password');

            $table->text('start\_url');

            $table->text('join\_url');

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('online\_classes');

    }

**Controller**

<?php

namespace App\Http\Controllers\Students;

use App\Models\Grade;

use App\Models\OnlineClasse;

use Illuminate\Http\Request;

use MacsiDigital\Zoom\Facades\Zoom;

use App\Http\Controllers\Controller;

class OnlineClasseController extends Controller

{

    use MeetingZoomTrait;

    public function index()

    {

        $online\_classes = OnlineClasse::where('created\_by',auth()->user()->email)->get();

        return view('pages.online\_classes.index', compact('online\_classes'));

    }

    public function create()

    {

        $grades = Grade::all();

        return view('pages.online\_classes.add', compact('grades'));

    }

    public function indirectCreate()

    {

        $grades = Grade::all();

        return view('pages.online\_classes.indirect', compact('grades'));

    }

    public function store(Request $request)

    {

        try {

            $meeting = $this->createMeeting($request);

            OnlineClasse::create([

                'integration' => true,

                'grade\_id' => $request->grade\_id,

                'classe\_id' => $request->classe\_id,

                'section\_id' => $request->section\_id,

                'created\_by' => auth()->user()->email,

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

                'topic' => $request->topic,

                'start\_at' => $request->start\_time,

                'duration' => $meeting->duration,

                'password' => $meeting->password,

                'start\_url' => $meeting->start\_url,

                'join\_url' => $meeting->join\_url,

            ]);

            flash()->addSuccess(trans('messages.success'));

            return redirect()->route('online\_classes.index');

        } catch (\Exception $e) {

            return redirect()->back()->withErrors(['error' => $e->getMessage()]);

        }

    }

    public function storeIndirect(Request $request)

    {

        try {

            online\_classe::create([

                'integration' => false,

                'grade\_id' => $request->grade\_id,

                'classe\_id' => $request->classe\_id,

                'section\_id' => $request->section\_id,

                'created\_by' => auth()->user()->email,

                'meeting\_id' => $request->meeting\_id,

                'topic' => $request->topic,

                'start\_at' => $request->start\_time,

                'duration' => $request->duration,

                'password' => $request->password,

                'start\_url' => $request->start\_url,

                'join\_url' => $request->join\_url,

            ]);

            flash()->addSuccess(trans('messages.success'));

            return redirect()->route('online\_classes.index');

        } catch (\Exception $e) {

            return redirect()->back()->with(['error' => $e->getMessage()]);

        }

    }

    public function show($id)

    {

        //

    }

    public function edit($id)

    {

        //

    }

public function update(Request $request, $id)

    {

        //

    }

    public function destroy(Request $request)

    {

        try {

            $info = OnlineClasse::find($request->id);

            if($info->integration == true){

                $meeting = Zoom::meeting()->find($request->meeting\_id);

                $meeting->delete();

               // online\_classe::where('meeting\_id', $request->id)->delete();

                OnlineClasse::destroy($request->id);

            }

            else{

               // online\_classe::where('meeting\_id', $request->id)->delete();

                OnlineClasse::destroy($request->id);

            }

            flash()->addSuccess(trans('messages.Delete'));

            return redirect()->route('online\_classes.index');

        } catch (\Exception $e) {

            return redirect()->back()->with(['error' => $e->getMessage()]);

        }

    }

}

**Créer un trait qu’on a utilisé dans controller**

<?php

namespace App\Http\Traits;

use MacsiDigital\Zoom\Facades\Zoom;

trait MeetingZoomTrait

{

    public function createMeeting($request){

        $user = Zoom::user()->first();

        $meetingData = [

            'topic' => $request->topic,

            'duration' => $request->duration,

            'password' => $request->password,

            'start\_time' => $request->start\_time,

            'timezone' => config('zoom.timezone')

          // 'timezone' => 'Africa/Cairo'

        ];

        $meeting = Zoom::meeting()->make($meetingData);

        $meeting->settings()->make([

            'join\_before\_host' => false,

            'host\_video' => false,

            'participant\_video' => false,

            'mute\_upon\_entry' => true,

            'waiting\_room' => true,

            'approval\_type' => config('zoom.approval\_type'),

            'audio' => config('zoom.audio'),

            'auto\_recording' => config('zoom.auto\_recording')

        ]);

        return  $user->meetings()->save($meeting);

    }

}

1. **TRAIT POUR UPLOAD ET DELETE FILE**

<?php

namespace App\Http\Traits;

use Illuminate\Support\Facades\Storage;

trait AttachFilesTrait

{

    public function uploadFile($request,$name,$folder)

    {

        $file\_name = $request->file($name)->getClientOriginalName();

        $request->file($name)->storeAs('attachments/',$folder.'/'.$file\_name,'upload\_attachments');

    }

    public function deleteFile($name)

    {

        $exists = Storage::disk('upload\_attachments')->exists('attachments/library/'.$name);

        if($exists)

        {

            Storage::disk('upload\_attachments')->delete('attachments/library/'.$name);

        }

    }

}

1. **TELECHARGER FILE**

 public function download($filename)

    {

        return response()->download(public\_path('attachments/library/'.$filename));

    }