Model

ProjectMember.php

**<?php  
  
namespace** CodeProject\Entities;  
  
**use** Illuminate\Database\Eloquent\Model;  
**use** Prettus\Repository\Contracts\Transformable;  
**use** Prettus\Repository\Traits\TransformableTrait;  
  
**class** ProjectMember **extends** Model **implements** Transformable  
{  
 **use** TransformableTrait;  
  
 **protected** $fillable = [  
 'project\_id',  
 'member\_id'  
 ];  
  
 **public** $timestamps = **false**;  
  
 **public function** project()  
 {  
 **return** $this->belongsTo(Project::*class*);  
 }  
  
 **public function** member()  
 {  
 **return** $this->belongsTo(User::*class*);  
 }  
  
}

RepositoryMemberEloquent

**<?php  
  
namespace** CodeProject\Repositories;  
  
**use** CodeProject\Presenters\ProjectMemberPresenter;  
**use** Prettus\Repository\Eloquent\BaseRepository;  
**use** Prettus\Repository\Criteria\RequestCriteria;  
**use** CodeProject\Entities\ProjectMember;  
  
*/\*\*  
 \* Class ProjectMemberRepositoryEloquent  
 \** ***@package*** *namespace CodeProject\Repositories;  
 \*/***class** ProjectMemberRepositoryEloquent **extends** BaseRepository **implements** ProjectMemberRepository  
{  
 */\*\*  
 \* Specify Model class name  
 \*  
 \** ***@return*** *string  
 \*/* **public function** model()  
 {  
 **return** ProjectMember::*class*;  
 }  
  
   
  
 */\*\*  
 \* Boot up the repository, pushing criteria  
 \*/* **public function** boot()  
 {  
 $this->pushCriteria(app(RequestCriteria::*class*));  
 }  
  
 **public function** getMembers($projectId)  
 {  
 **return** $this->findWhere(['project\_id' => $projectId]);  
 }  
  
 **public function** presenter()  
 {  
 **return** ProjectMemberPresenter::*class*;  
 }  
}

RespositoryUser

**<?php  
  
  
namespace** CodeProject\Repositories;  
  
  
  
**use** CodeProject\Entities\User;  
**use** CodeProject\Presenters\UserPresenter;  
**use** Prettus\Repository\Eloquent\BaseRepository;  
  
**class** UserRepositoryEloquent **extends** BaseRepository **implements** UserRepository  
{  
  
 **protected** $fieldSearchhable = [  
 'name'  
 ];  
  
 **public function** model()  
 {  
 **return** User::*class*;  
 }  
  
 **public function** presenter()  
 {  
 **return** UserPresenter::*class*;  
 }  
  
 */\*\*  
 \* Boot up the repository, pushing criteria  
 \*/* **public function** boot()  
 {  
 $this->pushCriteria(app(RequestCriteria::*class*));  
 }  
  
}

ProjectMemberTrasnform

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: rafael.franco  
 \* Date: 04/04/2016  
 \* Time: 10:26  
 \*/***namespace** CodeProject\Transformers;  
  
**use** CodeProject\Entities\User;  
**use** League\Fractal\TransformerAbstract;  
  
**class** ProjectMemberTransformer **extends** TransformerAbstract  
{  
  
 **protected** $defaultIncludes = ['user'];  
  
 **public function** transform(ProjectMember $model)  
 {  
  
 **return** [  
 'id' => $model->id,  
 'project\_id' => $model->project\_id,  
 ];  
 }  
  
 **public function** includeUser(ProjectMember $model)  
 {  
 **return** $this->collection($model->member, **new** MemberTransformer());  
 }  
}

MemberTransform

**<?php  
  
namespace** CodeProject\Transformers;  
  
**use** CodeProject\Entities\ProjectMember;  
**use** League\Fractal\TransformerAbstract;  
  
  
**class** MemberTransformer **extends** TransformerAbstract  
{  
 **public function** transform(ProjectMember $member)  
 {  
 **return** [  
 'project\_id' => $member->project->id,  
 'user\_id' => $member->member->id,  
 'name' => $member->member->name  
 ];  
 }  
}

ProjectMemberService

**<?php  
  
  
namespace** CodeProject\Services;  
  
  
**use** CodeProject\Repositories\ProjectMemberRepository;  
**use** CodeProject\Validators\ProjectMemberValidator;  
  
  
**class** ProjectMemberService **extends** Service  
{  
  
 **protected** $repository;  
 **protected** $validator;  
  
 **public function** \_\_construct(ProjectMemberRepository $repository, ProjectMemberValidator $validator)  
 {  
 $this->repository = $repository;  
 $this->validator = $validator;  
 }  
  
 /\*public function create(array $data)  
 {  
 // dd(isset($data['project\_id']) && isset($data['member\_id']) && !is\_null($data['member\_id']));  
 if (isset($data['project\_id']) && isset($data['member\_id']) && !is\_null($data['member\_id'])) {  
 $pm = $this->repository->skipPresenter()->findWhere(['project\_id' => $data['project\_id'], 'member\_id' => $data['member\_id']]);  
 dd($pm);  
 if (!$pm):  
 return parent::create($data);  
 endif;  
 }  
 return ['error'=>'Membro Inválido'];  
 }\*/  
  
}

Routes.php

**<?php**/\*  
|--------------------------------------------------------------------------  
| Application Routes  
|--------------------------------------------------------------------------  
|  
| Here is where you can register all of the routes for an application.  
| It's a breeze. Simply tell Laravel the URIs it should respond to  
| and give it the controller to call when that URI is requested.  
|  
\*/  
  
Route::*get*('/', **function** () {  
 **return** view('app');  
});  
  
  
Route::*post*('oauth/access\_token', **function**() {  
 **return** Response::*json*(Authorizer::*issueAccessToken*());  
});  
  
  
Route::*group*(['middleware'=>'oauth'], **function**(){  
  
 Route::*resource*('client', 'ClientController', ['except' => ['create', 'edit']]);  
 Route::*resource*('project', 'ProjectController', ['except' => ['create', 'edit']]);  
 Route::*resource*('project.member', 'ProjectMemberController', ['except' => ['edit', 'create', 'update']]);  
// Route::group(['middleware'=> 'CheckProjectOwner'], function() {  
// Route::resource('project', 'ProjectController', ['except' => ['create', 'edit']]);  
// });  
  
 Route::*group*(['middleware'=>'check.project.permission', 'prefix'=>'project'], **function**() {  
  
 Route::*resource*('{project}/note', 'ProjectNoteController', ['except' => ['edit', 'create']]);  
 Route::*resource*('{project}/task', 'ProjectTaskController', ['except' => ['edit', 'create']]);  
 Route::*resource*('{project}/file', 'ProjectFileController', ['except' => ['edit', 'create']]);  
 Route::*get*('{project}/file/{file}/download', 'ProjectFileController@showFile');  
  
  
  
 Route::*get*('{projectId}/member', 'ProjectMemberController@index');  
 Route::*post*('{projectId}/member', 'ProjectMemberController@store');  
 Route::*get*('{projectId}/member/{id}', 'ProjectMemberController@show');  
 Route::*delete*('{projectId}/member/{id}', 'ProjectMemberController@destroy');  
  
  
 });  
  
 Route::*get*('user/authenticated', 'UserController@authenticated');  
  
});

ProjectMemeberController.php

**<?php  
  
namespace** CodeProject\Http\Controllers;  
  
**use** Illuminate\Http\Request;  
**use** CodeProject\Http\Controllers\Controller;  
**use** CodeProject\Repositories\ProjectMemberRepository;  
**use** CodeProject\Services\ProjectMemberService;  
  
**class** ProjectMemberController **extends** Controller  
{  
  
 **private** $repository;  
 **private** $service;  
  
 **public function** \_\_construct(ProjectMemberRepository $repository, ProjectMemberService $service)  
 {  
 $this->repository = $repository;  
 $this->service = $service;  
 $this->middleware('check.project.owner', ['except' => ['index', 'show']]);  
 $this->middleware('check.project.permission', ['except' => ['store', 'destroy']]);  
 }  
  
 **public function** index($id)  
 {  
 **return** $this->repository->findWhere(['project\_id' => $id]);  
 }  
  
 **public function** store(Request $request, $id)  
 {  
 **if**($request->member\_id):  
 $member = $this->repository->skipPresenter()->findWhere(['member\_id'=>$request->member\_id, 'project\_id'=>$id]);  
 **if**(count($member) == 0):  
 **return** $this->service->create(array\_merge($request->all(), ['project\_id' => $id]));  
 **endif**;  
 **endif**;  
 **return** ['error'=>'O membro já pertence ao projeto!'] ;  
 }  
  
 **public function** show($id, $memberId)  
 {  
 **return** $this->repository->find($memberId);  
 }  
  
 **public function** destroy($id, $memberId)  
 {  
 **return** ['deleted'=>$this->service->delete($memberId)];  
 }  
  
}