Migration Project Tasks

**<?php  
  
use** Illuminate\Database\Schema\Blueprint;  
**use** Illuminate\Database\Migrations\Migration;  
  
**class** CreateProjectTasksTable **extends** Migration  
{  
 */\*\*  
 \* Run the migrations.  
 \*  
 \** ***@return*** *void  
 \*/* **public function** up()  
 {  
 Schema::*create*('project\_tasks', **function** (Blueprint $table) {  
 $table->increments('id');  
 $table->string('name');  
 $table->integer('project\_id')->unsigned();  
 $table->foreign('project\_id')->references('id')->on('projects');  
 $table->date('start\_date')->nullalbe();  
 $table->date('due\_date')->nullalbe();  
 $table->smallInteger('status')->unsigned();  
 $table->timestamps();  
 });  
 }  
  
 */\*\*  
 \* Reverse the migrations.  
 \*  
 \** ***@return*** *void  
 \*/* **public function** down()  
 {  
 Schema::*drop*('project\_tasks');  
 }  
}

ProjectTaskReopsitoryEloquent

**<?php  
  
namespace** CodeProject\Repositories;  
  
**use** CodeProject\Presenters\ProjectTaskPresenter;  
**use** Prettus\Repository\Eloquent\BaseRepository;  
**use** Prettus\Repository\Criteria\RequestCriteria;  
**use** CodeProject\Repositories\ProjectTaskRepository;  
**use** CodeProject\Entities\ProjectTask;  
**use** CodeProject\Validators\ProjectTaskValidator;;  
  
*/\*\*  
 \* Class ProjectTaskRepositoryEloquent  
 \** ***@package*** *namespace CodeProject\Repositories;  
 \*/***class** ProjectTaskRepositoryEloquent **extends** BaseRepository **implements** ProjectTaskRepository  
{  
 */\*\*  
 \* Specify Model class name  
 \*  
 \** ***@return*** *string  
 \*/* **public function** model()  
 {  
 **return** ProjectTask::*class*;  
 }  
  
   
  
 */\*\*  
 \* Boot up the repository, pushing criteria  
 \*/* **public function** boot()  
 {  
 $this->pushCriteria(app(RequestCriteria::*class*));  
 }  
  
 **public function** presenter()  
 {  
 **return** ProjectTaskPresenter::*class*;  
 }  
}

ProjectTaskService

**<?php  
  
namespace** CodeProject\Services;  
  
**use** CodeProject\Repositories\ProjectTaskRepository;  
**use** CodeProject\Repositories\ProjectRepository;  
**use** CodeProject\Validators\ProjectTaskValidator;  
  
**class** ProjectTaskService **extends** Service  
{  
  
 **protected** $repository;  
 **protected** $projectRepository;  
 **protected** $validator;  
  
 **public function** \_\_construct(ProjectTaskRepository $repository,ProjectRepository $projectRepository, ProjectTaskValidator $validator)  
 {  
 $this->repository = $repository;  
 $this->validator = $validator;  
 $this->projectRepository = $projectRepository;  
 }  
  
  
}

ProjectTaskController.php

**<?php  
  
namespace** CodeProject\Http\Controllers;  
  
**use** Illuminate\Http\Request;  
**use** CodeProject\Repositories\ProjectTaskRepository;  
**use** CodeProject\Services\ProjectTaskService;  
  
**class** ProjectTaskController **extends** Controller  
{  
  
 **private** $repository;  
 **private** $service;  
  
 **public function** \_\_construct(ProjectTaskRepository $repository, ProjectTaskService $sevice)  
 {  
 $this->repository = $repository;  
 $this->service = $sevice;  
 }  
  
 **public function** index($id)  
 {  
 **return** $this->repository->findWhere(['project\_id' => $id]);  
 }  
  
 **public function** store($id, Request $request)  
 {  
 **return** $this->service->create(array\_merge($request->all(), ['project\_id'=>$id]));  
 }  
  
 **public function** show($id, $task)  
 {  
 **return** $this->repository->find($task);  
 }  
  
 **public function** update($id, $task, Request $request)  
 {  
 **return** $this->service->update(array\_merge($request->all(), ['project\_id' => $id]), $task);  
 }  
  
 **public function** destroy($id, $task)  
 {  
 $this->repository->delete($task);  
 }  
  
}