Blob – Tipo de armazenamento de dados com tamanhos maior, armazenar esses dados no mysql que ele consegue trabalhar, porque o tamanho do blob é bem menor do que o tamanho do arquivo.

Acrescentar umas informações de retorno:

Refatorando o ProjectFileController.php

**<?php  
  
namespace** CodeProject\Http\Controllers;  
  
**use** CodeProject\Repositories\ProjectFileRepository;  
  
**use** CodeProject\Services\ProjectFileService;  
**use** Illuminate\Contracts\Filesystem\Factory;  
**use** Illuminate\Http\Request;  
  
**use** CodeProject\Http\Requests;  
  
  
  
**class** ProjectFileController **extends** Controller  
{  
  
 **private $repository**;  
 **private $service**;  
 **private $storage**;  
  
 **public function** \_\_construct(ProjectFileRepository $repository, ProjectFileService $sevice, Factory $storage)  
 {  
 $this->**repository** = $repository;  
 $this->**service** = $sevice;  
 $this->**storage** = $storage;  
 }  
  
 **public function** index($id)  
 {  
 **return** $this->**repository**->skipPresenter()->findWhere([**'project\_id'** => $id]);  
 }  
  
 **public function** store($id, Request $request)  
 {  
 **return** [$this->**service**->create(*array\_merge*($request->all(), [**'project\_id'** => $id]))];  
 }  
  
 **public function** update($id, $projectFileId, Request $request)  
 {  
 **return** $this->**service**->update($request->all(), $projectFileId);  
 }  
  
 **public function** destroy($id, $file)  
 {  
 $this->**service**->deleteFile($file);  
 }  
  
 **public function** show($id, $idFile)  
 {  
 **return** $this->**repository**->find($idFile);  
 }  
  
*// public function showFile($id, $idFile)  
// {  
// // dd($id);  
// $filePath = $this->service->getFilePath($idFile);  
// $fileContents = file\_get\_contents($filePath);  
// $file64 = base64\_encode($fileContents);  
// $name = $this->service->getFileName($idFile);  
// return [  
// 'file' => $file64,  
// 'size' => filesize($filePath),  
// 'name' => $name  
// ];  
// }* **public function** showFile($id, $idFile)  
 {  
 $model = $this->**repository**->skipPresenter()->find($idFile);  
 $filePath = $this->**service**->getFilePath($idFile);  
 $fileContents = *file\_get\_contents*($filePath);  
 $file64 = *base64\_encode*($fileContents);  
 $name = $this->**service**->getFileName($idFile);  
 **return** [  
 **'file'** => $file64,  
 **'size'** => *filesize*($filePath),  
 **'name'** => $name,  
 **'mime\_type'** => $this->**storage**->mimeType($name)  
 ];  
 }  
}

App.blade.php

<**script src="**{{ asset(**'build/js/vendor/blob-util.min.js'**) }}**"**></**script**>

Gulpfile.js

config.**bower\_path** + **'/blob-util/dist/blob-util.min.js'**,

ProjectFileDownload.js “DIRETIVA”

**return** {  
 **restrict**: **"E"**,  
 **templateUrl**: appConfig.**baseUrl** + **'/build/views/templates/projectFileDownload.html'**,  
 link: **function** (scope, element, attr) {  
 **var** anchor = element.**children**()[0];  
 scope.$on(**'save-file'**,  
 **function** (event, data) {  
 ***$***(anchor).removeClass(**'disable'**);  
 ***$***(anchor).text(**'Save File!'**);  
  
 **blobUtil**.**base64StringToBlob**(data.file).then(**function** (blob) {  
 ***$***(anchor).attr({  
 **href**: $window.**URL**.**createObjectURL**(blob, data.mime\_type),  
 **download**: data.name  
 });  
 });  
  
  
 $timeout(**function** () {  
 scope.downloadFile = **function** () {};  
 ***$***(anchor)[0].click();  
 });  
  
 });  
 },

Instalar um biblioteca do blob-util

https://github.com/nolanlawson/blob-util

$ bower install blob-util --save

