Bien sûr. La création des contrôleurs est l'étape logique après les modèles et migrations. Ils contiendront la logique métier pour répondre aux requêtes des utilisateurs.

Voici une structure plausible pour les contrôleurs, en séparant la logique publique (le sondage) de la logique privée (l'administration).

### **Contrôleurs Publics**

#### 1. SurveyController

Ce contrôleur gère l'affichage du sondage, la validation et l'enregistrement des réponses, ainsi que l'affichage de la page de remerciement et des résultats pour l'utilisateur.

**app/Http/Controllers/SurveyController.php**

PHP

<?php  
  
namespace App\Http\Controllers;  
  
use App\Models\Answer;  
use App\Models\Question;  
use App\Models\Submission;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\DB;  
use Illuminate\Support\Str;  
  
class SurveyController extends Controller  
{  
 /\*\*  
 \* Affiche la page du sondage avec toutes les questions.  
 \*/  
 public function show()  
 {  
 // On récupère toutes les questions ordonnées par leur numéro  
 $questions = Question::orderBy('order')->get();  
 return view('survey.show', compact('questions'));  
 }  
  
 /\*\*  
 \* Valide et enregistre les réponses du sondage.  
 \*/  
 public function store(Request $request)  
 {  
 // Validation dynamique basée sur les questions en BDD  
 $rules = [];  
 $questions = Question::all();  
  
 foreach ($questions as $question) {  
 // Toutes les questions sont obligatoires   
 $rule = 'required';  
 // La question 1 (email) doit avoir une validation de format email   
 if ($question->id === 1) {  
 $rule .= '|email';  
 }  
 $rules['answers.' . $question->id] = $rule;  
 }  
  
 $validatedData = $request->validate($rules);  
  
 // On utilise une transaction pour s'assurer que tout est bien enregistré  
 $submission = DB::transaction(function () use ($validatedData) {  
 // 1. Créer une nouvelle soumission avec un jeton unique  
 $submission = Submission::create([  
 'url\_token' => Str::uuid(), // Génère un identifiant unique et sécurisé  
 ]);  
  
 // 2. Enregistrer chaque réponse en l'associant à la soumission  
 foreach ($validatedData['answers'] as $questionId => $value) {  
 Answer::create([  
 'submission\_id' => $submission->id,  
 'question\_id' => $questionId,  
 'value' => $value,  
 ]);  
 }  
  
 return $submission;  
 });  
  
 // Rediriger vers une page de remerciement avec l'URL unique   
 $uniqueUrl = route('survey.results', ['token' => $submission->url\_token]);  
 return view('survey.thankyou', ['uniqueUrl' => $uniqueUrl]);  
 }  
  
 /\*\*  
 \* Affiche les réponses d'un utilisateur via son jeton unique.  
 \*/  
 public function results(string $token)  
 {  
 // Récupère la soumission via le token, ou renvoie une erreur 404 si non trouvée  
 $submission = Submission::where('url\_token', $token)->firstOrFail();  
  
 // On charge les réponses et les questions associées pour optimiser les requêtes  
 $submission->load('answers.question');  
  
 return view('survey.results', compact('submission'));  
 }  
}

### **Contrôleurs de la Section Administration**

Ces contrôleurs seront placés dans un sous-dossier Admin pour une meilleure organisation. Ils gèrent l'affichage des données pour l'administrateur.

#### 2. DashboardController

Ce contrôleur prépare les données pour les graphiques de la page d'accueil de l'administration.

**app/Http/Controllers/Admin/DashboardController.php**

PHP

<?php  
  
namespace App\Http\Controllers\Admin;  
  
use App\Http\Controllers\Controller;  
use App\Models\Answer;  
use Illuminate\Support\Facades\DB;  
  
class DashboardController extends Controller  
{  
 /\*\*  
 \* Affiche le tableau de bord avec les statistiques.  
 \*/  
 public function index()  
 {  
 // Prépare les données pour les 3 graphiques "Pie charts"   
 $pieChartData = [  
 'q6' => $this->getChartDataForQuestion(6), // Marque casque VR  
 'q7' => $this->getChartDataForQuestion(7), // Magasin d'application  
 'q10' => $this->getChartDataForQuestion(10), // Utilisation principale  
 ];  
  
 // Prépare les données pour le "Radar chart" (questions 11 à 15)   
 $radarChartData = [  
 'labels' => ['Qualité image', 'Confort interface', 'Connexion réseau', 'Graphismes 3D', 'Qualité audio'],  
 'averages' => [  
 Answer::where('question\_id', 11)->avg(DB::raw('CAST(value AS UNSIGNED)')),  
 Answer::where('question\_id', 12)->avg(DB::raw('CAST(value AS UNSIGNED)')),  
 Answer::where('question\_id', 13)->avg(DB::raw('CAST(value AS UNSIGNED)')),  
 Answer::where('question\_id', 14)->avg(DB::raw('CAST(value AS UNSIGNED)')),  
 Answer::where('question\_id', 15)->avg(DB::raw('CAST(value AS UNSIGNED)')),  
 ],  
 ];  
  
 return view('admin.dashboard', compact('pieChartData', 'radarChartData'));  
 }  
  
 /\*\*  
 \* Helper pour récupérer et formater les données d'un graphique.  
 \*/  
 private function getChartDataForQuestion(int $questionId)  
 {  
 return Answer::where('question\_id', $questionId)  
 ->select('value', DB::raw('count(\*) as total'))  
 ->groupBy('value')  
 ->pluck('total', 'value');  
 }  
}

#### 3. QuestionController

Conformément au brief, ce contrôleur ne fait qu'afficher la liste des questions1.

**app/Http/Controllers/Admin/QuestionController.php**

PHP

<?php  
  
namespace App\Http\Controllers\Admin;  
  
use App\Http\Controllers\Controller;  
use App\Models\Question;  
  
class QuestionController extends Controller  
{  
 /\*\*  
 \* Affiche la liste des questions du sondage.   
 \*/  
 public function index()  
 {  
 $questions = Question::orderBy('order')->get();  
 return view('admin.questions.index', compact('questions'));  
 }  
}

#### 4. AnswerController

Ce contrôleur affiche toutes les réponses soumises, regroupées par participant.

**app/Http/Controllers/Admin/AnswerController.php**

PHP

<?php  
  
namespace App\Http\Controllers\Admin;  
  
use App\Http\Controllers\Controller;  
use App\Models\Submission;  
  
class AnswerController extends Controller  
{  
 /\*\*  
 \* Affiche toutes les réponses, groupées par soumission.   
 \*/  
 public function index()  
 {  
 // On utilise la pagination pour la performance  
 // On charge les relations pour éviter le problème N+1  
 $submissions = Submission::with('answers.question')  
 ->latest() // Les plus récentes d'abord  
 ->paginate(10);   
  
 return view('admin.answers.index', compact('submissions'));  
 }  
}

### **Déclaration des Routes**

Enfin, vous devez déclarer les routes correspondantes dans routes/web.php.

**routes/web.php**

PHP

<?php  
  
use App\Http\Controllers\Admin\DashboardController;  
use App\Http\Controllers\Admin\QuestionController;  
use App\Http\Controllers\Admin\AnswerController;  
use App\Http\Controllers\SurveyController;  
use Illuminate\Support\Facades\Route;  
  
/\*  
|--------------------------------------------------------------------------  
| Routes Publiques  
|--------------------------------------------------------------------------  
\*/  
Route::get('/', [SurveyController::class, 'show'])->name('survey.show');  
Route::post('/', [SurveyController::class, 'store'])->name('survey.store');  
Route::get('/merci', fn() => view('survey.thankyou'))->name('survey.thankyou'); // Page de remerciement simple  
Route::get('/resultats/{token}', [SurveyController::class, 'results'])->name('survey.results');  
  
/\*  
|--------------------------------------------------------------------------  
| Routes Administration  
|--------------------------------------------------------------------------  
\*/  
// Les routes de l'administration sont groupées, préfixées par /administration  
// et protégées par le middleware 'auth' pour n'être accessibles qu'aux utilisateurs connectés.  
Route::middleware('auth')->prefix('administration')->name('admin.')->group(function () {  
 Route::get('/', [DashboardController::class, 'index'])->name('dashboard');  
 Route::get('/questionnaire', [QuestionController::class, 'index'])->name('questions.index');  
 Route::get('/reponses', [AnswerController::class, 'index'])->name('answers.index');  
});  
  
// Inclure les routes d'authentification (login, etc.) si vous utilisez un starter kit  
// Par exemple, pour Laravel Breeze ou Jetstream :  
require \_\_DIR\_\_.'/auth.php';

**Sources**

1. <https://github.com/okyadyy/submission>

2. <https://github.com/hendri2212/konseling>

3. <https://github.com/AliniCanedo/tcc-aura>

4. <https://github.com/Ahmad-Mateen/OnlineQuizSystem>

5. <https://github.com/MrIsmayil/survey_project>