Absolument. Voici le code pour les migrations et les modèles Eloquent correspondant à la base de données que nous avons définie. Ce code respecte les conventions de Laravel.

### **Migrations**

Les migrations créent la structure des tables dans votre base de données. Vous pouvez générer ces fichiers avec la commande php artisan make:migration create\_nom\_de\_la\_table\_table. L'ordre de création est important à cause des clés étrangères.

#### 1. Table users

Ce fichier est généralement déjà présent dans un nouveau projet Laravel.

<br>

database/migrations/YYYY\_MM\_DD\_000001\_create\_users\_table.php

PHP

<?php  
  
use Illuminate\\Database\\Migrations\\Migration;  
use Illuminate\\Database\\Schema\\Blueprint;  
use Illuminate\\Support\\Facades\\Schema;  
  
return new class extends Migration  
{  
/\\*\\*  
\\* Run the migrations.  
\\*/  
public function up(): void  
{  
Schema::create('users', function (Blueprint $table) {  
$table-\>id();  
$table-\>string('name');  
$table-\>string('email')-\>unique();  
$table-\>timestamp('email\\_verified\\_at')-\>nullable();  
$table-\>string('password');  
$table-\>rememberToken();  
$table-\>timestamps();  
});  
}

/\*\*

* Reverse the migrations. \*/ public function down(): void { Schema::dropIfExists('users'); }

<!-- end list -->

};

#### 2. Table questions

<br>

database/migrations/YYYY\_MM\_DD\_000002\_create\_questions\_table.php

PHP

<?php  
  
use Illuminate\\Database\\Migrations\\Migration;  
use Illuminate\\Database\\Schema\\Blueprint;  
use Illuminate\\Support\\Facades\\Schema;  
  
return new class extends Migration  
{  
/\\*\\*  
\\* Run the migrations.  
\\*/  
public function up(): void  
{  
Schema::create('questions', function (Blueprint $table) {  
$table-\>id();  
$table-\>string('title');  
[cite\\_start]$table-\>enum('type', ['A', 'B', 'C']); // Type A, B ou C   
$table-\>json('options')-\>nullable(); // Pour les choix des questions de type A  
$table-\>integer('order'); // Pour l'ordre d'affichage  
$table-\>timestamps();  
});  
}

/\*\*

* Reverse the migrations. \*/ public function down(): void { Schema::dropIfExists('questions'); }

};

#### **3. Table submissions**

<br>

database/migrations/YYYY\_MM\_DD\_000003\_create\_submissions\_table.php

PHP

<?php  
  
use Illuminate\\Database\\Migrations\\Migration;  
use Illuminate\\Database\\Schema\\Blueprint;  
use Illuminate\\Support\\Facades\\Schema;  
  
return new class extends Migration  
{  
/\\*\\*  
\\* Run the migrations.  
\\*/  
public function up(): void  
{  
Schema::create('submissions', function (Blueprint $table) {  
$table-\>id();  
[cite\\_start]// Jeton unique pour l'URL de consultation des réponses   
$table-\>string('url\\_token')-\>unique();  
$table-\>timestamps();  
});  
}

/\*\*

* Reverse the migrations. \*/ public1 function down(): void { Schema::dropIfExists('submissions'); }

};

#### 4. Table answers

<br>

database/migrations/YYYY\_MM\_DD\_000004\_create\_answers\_table.php

PHP

<?php  
  
use Illuminate\\Database\\Migrations\\Migration;  
use Illuminate\\Database\\Schema\\Blueprint;  
use Illuminate\\Support\\Facades\\Schema;  
  
return new class extends Migration  
{  
/\\*\\*  
\\* Run the migrations.  
\\*/  
public function up(): void  
{  
Schema::create('answers', function (Blueprint $table) {  
$table-\>id();  
// Clé étrangère vers la table submissions  
$table-\>foreignId('submission\\_id')-\>constrained()-\>onDelete('cascade');  
// Clé étrangère vers la table questions  
$table-\>foreignId('question\\_id')-\>constrained()-\>onDelete('cascade');  
$table-\>text('value'); // La réponse de l'utilisateur  
$table-\>timestamps();  
});  
}

/\*\*

* Reverse the migrations. \*/ public function down(): void { Schema::dropIfExists('answers'); }

};

**Note :** L'option onDelete('cascade') assure que si une soumission ou une question est supprimée, toutes les réponses associées le seront également, garantissant ainsi l'intégrité de la base de données.

### **Modèles Eloquent**

Les modèles permettent d'interagir avec les tables de manière intuitive et de définir les relations entre elles. Vous pouvez les générer avec php artisan make:model NomDuModele.

#### 1. Modèle User

Ce modèle est généralement déjà présent et configuré.

<br>

app/Models/User.php

PHP

<?php  
  
namespace App\\Models;  
  
use Illuminate\\Foundation\\Auth\\User as Authenticatable;  
use Illuminate\\Notifications\\Notifiable;  
  
class User extends Authenticatable  
{  
use Notifiable;

protected $fillable = [

'name',

'email',

'password',

];

protected $hidden = [

'password',

'remember\_token',

];

protected $casts = [

'email\_verified\_at' => 'datetime',

'password' => 'hashed',

];

}

#### **2. Modèle Question**

<br>

app/Models/Question.php

PHP

<?php  
  
namespace App\\Models;  
  
use Illuminate\\Database\\Eloquent\\Factories\\HasFactory;  
use Illuminate\\Database\\Eloquent\\Model;  
use Illuminate\\Database\\Eloquent\\Relations\\HasMany;  
  
class Question extends Model  
{  
use HasFactory;

protected $fillable2 = [

'title',

'type',

'options',

'order',

];

// Le champ3 'options' sera automatiquement converti en tableau/objet PHP

protected $casts = [

'options' => 'array',

];

/\*\*

* Une question peut avoir plusieurs réponses (à travers différentes soumissions). \*/ public function answers(): HasMany { return $this->hasMany(Answer::class); }

}

#### **3. Modèle Submission**

<br>

app/Models/Submission.php4

PHP

<?php  
  
namespace App\\Models;  
  
use Illuminate\\Database\\Eloquent\\Factories\\HasFactory;  
use Illuminate\\Database\\Eloquent\\Model;  
use Illuminate\\Database\\Eloquent\\Relations\\HasMany;  
  
class Submission extends Model  
{  
use HasFactory;

protected $fillable = [

'url\_token',

];

/\*\*

* Une soumission contient plusieurs réponses. \*/ public function answers(): HasMany { return $this->hasMany(Answer::class); }

}

#### 4. Modèle Answer

<br>

app/Models/Answer.php

PHP

<?php  
  
namespace App\\Models;  
  
use Illuminate\\Database\\Eloquent\\Factories\\HasFactory;  
use Illuminate\\Database\\Eloquent\\Model;  
use Illuminate\\Database\\Eloquent\\Relations\\BelongsTo;  
  
class Answer extends Model  
{  
use HasFactory;

protected $fillable = [

'submission\_id',

'question\_id',

'value',

];

/\*\*

* Une réponse appartient à une soumission. \*/ public function submission(): BelongsTo { return $this->belongsTo(Submission::class); }

/\*\*

* Une réponse concerne une question. \*/ public function question(): BelongsTo { return $this->belongsTo(Question::class);5 }

}

**Sources**

1. <https://github.com/armandwipangestu/armandwipangestu.vercel.app>

2. <https://github.com/Adama699/stackoverflow>

3. <https://github.com/41saurab/Task_Management>

4. <https://abstractentropy.com/laravel-notifications-telegram-bot/>

5. <https://stackoverflow.com/questions/64696911/laravel-api-argument-1-passed-to-tokenguard-construct-must-implement-inte>

6. <https://github.com/0x004d4m/sallourasweets>

7. <https://github.com/AnaGapEvi/project-alison-laravel>

8. <https://github.com/zeindevs/laravel-sas-sv-knn>

9. <https://github.com/AhElhefny/fat2fit>