

CRÉATION D'UN OUTIL DE SONDAGE EN LIGNE



Auteur : Robustiano LOMBARDO
Chef de projet : Karim BOURAHLA
Experts : M. MALHERBE et M. BERNEY
Version : 5.1
Date de dernière mise à jour : 22.05.2024

Table des matières

1. Presentation du projet	1
2. Analyse / Conception.....	2
2.1 Base de données.....	2
2.2 Stratégie de test.....	3
3. Implémentation	3
3.1 Mise en place de l'environnement	3
3.2 Model	4
3.3 Faker/Seeder.....	5
3.4 CRUD pour les sondages	5
3.5 Graphique.....	9
3.6 Description des tests effectués.....	11
3.7 Problèmes rencontrer.....	14
4. Conclusion	14
5. Annexes	15

1. Presentation du projet

Le projet consiste en la programmation d'une application de sondage en ligne en utilisant Laravel et le Kit Jetstream (Inertia.js/Vue.js/Tailwind CSS). Le projet doit donc aussi prendre en compte la conception et l'implémentation d'une base de données MySQL.



L'application doit inclure des tests avec Pest, une gestion des rôles, des exports vers des fichiers Excel et PDF, la génération et l'affichage d'un graphique et finalement pouvoir appliquer les fonctionnalités CRUD sur tous les modèles.

L'identité visuelle étant secondaire, l'application est donc basé sur les différents rendus disponibles sur le site web <https://tailblocks.cc/>.

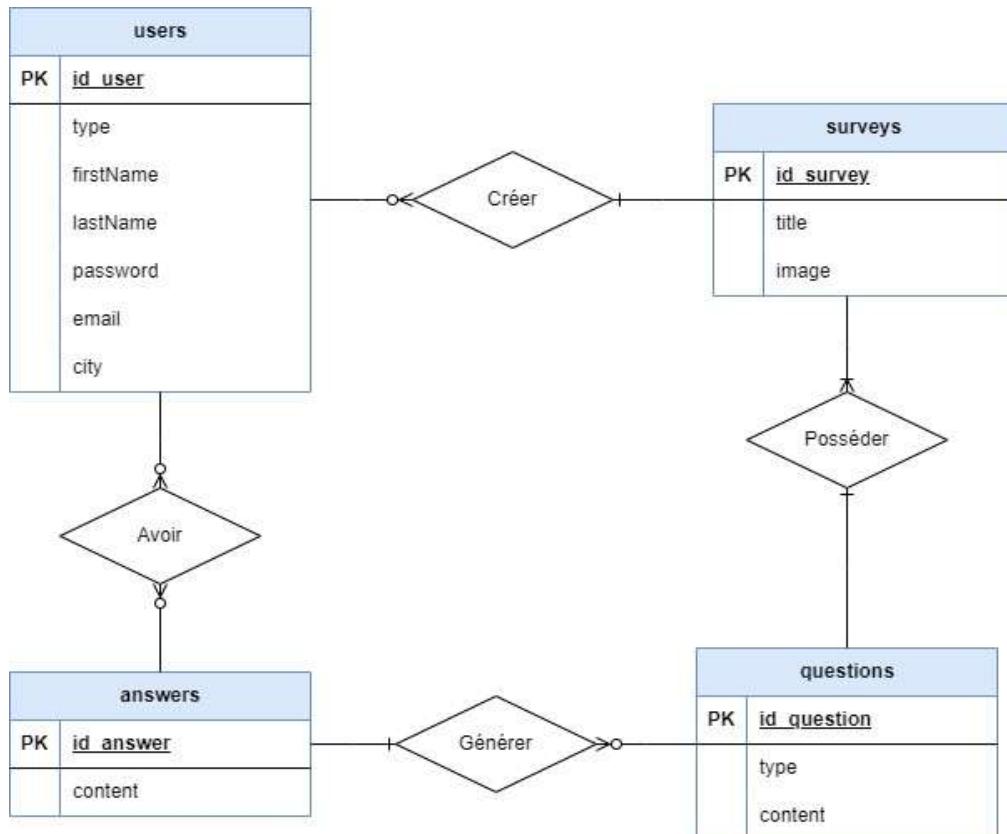
Le client évalue les points suivant :

1. Les liens vers les sondages sont uniques et sécurisés.
2. L'administrateur de la plateforme peut exécuter un CRUD de tous les modèles.
3. L'administrateur peut exporter la liste des créateurs au format Excel avec choix des attributs.
4. L'administrateur peut générer un rapport pdf avec la liste des sondages et choix des attributs.
5. L'implémentation d'une stratégie permettant la sécurisation de la base de données.
6. La mise en place de test PEST.
7. La mise en place de la méthode de suivi des versions avec GitFlow.

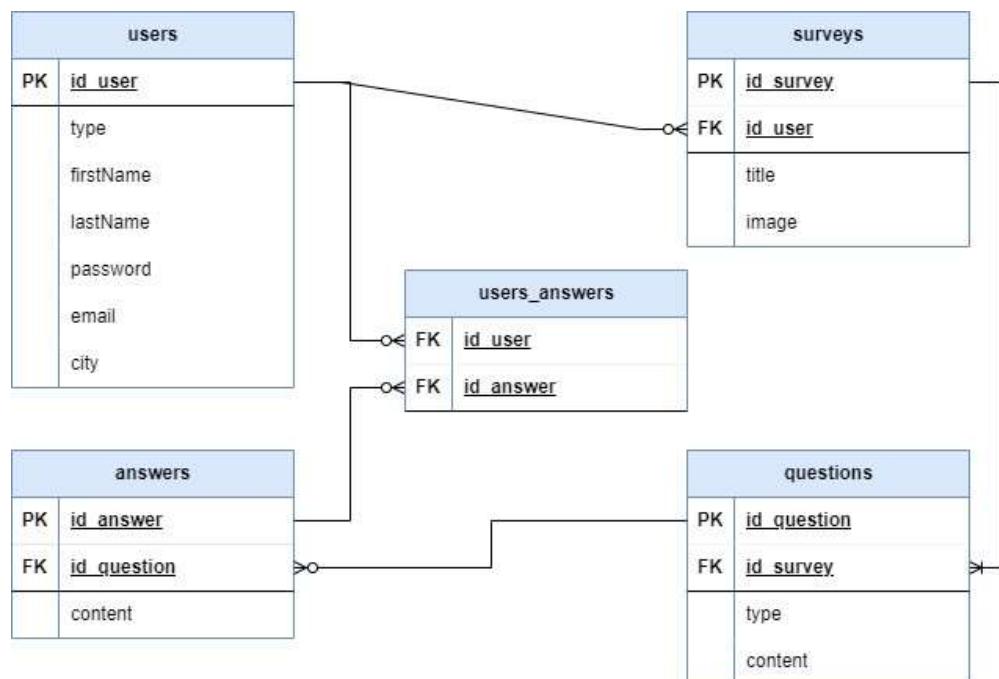
2. Analyse / Conception

2.1 Base de données

MCD :



MLD :



2.2 Stratégie de test

Les fonctionnalités de l'application sont tester grâce à Pest, celui-ci est un framework de tests unitaires. Son utilisation a été demandé par le client pour la réalisation des tests. Les tests unitaires servent à vérifier les fonctionnalités CRUD, la génération des PDFs, la génération des fichiers d'export Excel, la création du graphique et la gestion des droits d'accès pour les différents utilisateurs.

L'application s'occupent de générer de fausse données pour que les tests puissent s'exécuter sans problème. Les tests couvrent environ 90% des fonctionnalités, par exemple les tests unitaires ne peuvent pas vérifier le bon affichage du graphique, mais plutôt uniquement les données que celui-ci affiche.

3. Implémentation

3.1 Mise en place de l'environnement

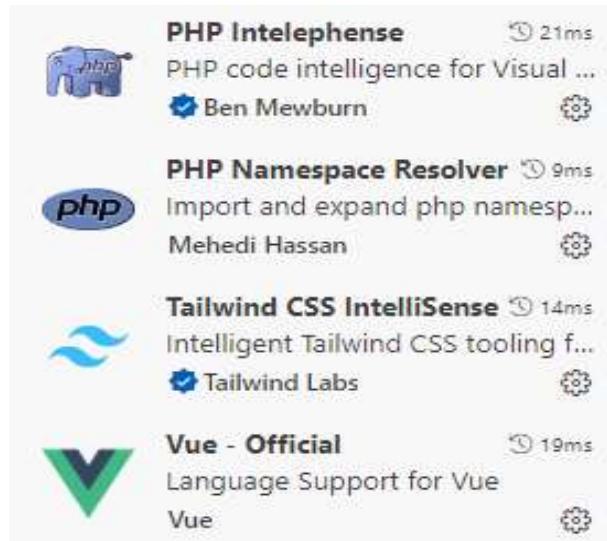
Pour commencer, le projet nécessite d'avoir une version de PHP supérieur ou égal à la 8.2, la dernière version de Composer, Node.js et Xampp installé.

Pour la création du projet, exécuter la commande de création d'un nouveau projet Laravel qui est `laravel new survey` ensuite, voici les paramètres à choisir pour mettre en place l'environnement :

```
Would you like to install a starter kit? [No starter kit]:  
[none] No starter kit  
[breeze] Laravel Breeze  
[jetstream] Laravel Jetstream  
> jetstream  
  
Which Jetstream stack would you like to install? [Livewire]:  
[livewire] Livewire  
[inertia] Vue with Inertia  
> inertia  
  
Would you like any optional features? [None]:  
[none] None  
[api] API support  
[dark] Dark mode  
[verification] Email verification  
[teams] Team support  
[ssr] Inertia SSR  
> none  
  
Which testing framework do you prefer? [Pest]:  
[0] Pest  
[1] PHPUnit  
> 0
```

Lorsque de la création du projet, installé dans celui-ci les 4 packages suivants : pour la gestion `composer require spatie/laravel-permission` des rôles et des permissions, pour la `npm install chart.js` génération de graphique, pour la génération de fichier PDF utiliser `composer require barryvdh/laravel-dompdf` et pour finir qui permet `composer require maatwebsite/excel` l'exportation et l'importation de données via un fichier Excel.

Pour faciliter le développement sous cette environnement voici des extensions utiles sur Visual Studio code :



L'environnement est donc désormais propice au bon développement de cette application Laravel avec un kit Jetstream.

3.2 Model

La création des modèles s'effectue grâce à la commande `php artisan make:model Survey -a` le « -a » sers à créer tous les éléments en rapport avec le modèle comme le controller, le seeder et encore d'autres fichiers. La création et la configuration de ces éléments est donc obligatoire pour que des données soient retournées de la base de données.

Jetsteam ayant déjà un système de gestion des utilisateurs, la modification du fichier model et du fichier de migration de la table users pour appliquer les modifications apportées à cette table.

```
protected $fillable = [
    'firstName',
    'lastName',
    'email',
    'password',
    'city',
];
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('firstName');
    $table->string('lastName');
    $table->string('city')->nullable();
    $table->string('email')->unique()->nullable();
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password')->nullable();
    $table->rememberToken();
    $table->foreignId('current_team_id')->nullable();
    $table->string('profile_photo_path', 2048)->nullable();
    $table->timestamps();
});
```

La modification des fichiers Vue ou PHP contenant des références au champ « name » est obligatoire, celui-ci n'existant plus la valeur de ce champ retourne une erreur.

3.3 Faker/Seeder

Les fakers et seeders sont utiliser pour la génération de données de tests. Ces fichiers ont été générer au préalable grâce à la commande de création des modèles. Les fichiers fakers servent de modèle pour la génération des données. La création des valeurs de chaque élément de la table choisi ce fait ici, voici un exemple :

```
return [
    'title' => $this->faker->unique()->lexify('SurTitle-???'),
    'image' => $this->faker->unique()->lexify('SurImage-???'),
    'id_user' => User::inRandomOrder()->first(),
];
```

Pour les fichiers seeders appelle uniquement la classe correspondante à son faker, utilisé `ClassDeVotreChoix::factory(5)->create();` dans le seeder pour générer le nombre de données de prédéfini. Et dans le fichier « DatabaSeeder.php » appellé les tous les autres seeders, comme ceux-ci :

```
$this->call([
    UserSeeder::class,
    SurveySeeder::class,
    QuestionSeeder::class,
    AnswerSeeder::class,
    UserAnswerSeeder::class,
]);
```

3.4 CRUD pour les sondages

Pour illustrer l'implémentation des fonctionnalités CRUD, celles de la table **surveys** semble être la plus approprié vu le niveau de complexité de celle-ci. Car lorsqu'une modification à lieu sur un de ces éléments, les éléments de la table **questions** et **answers**.

Comme citées précédemment, la fonctionnalité **CREATION** doit permettre la création d'élément **surveys**. Dans l'affichage, la création de variable est nécessaire pour récupérer données entrée par l'utilisateur.

```
const form = reactive({
    title: null,
    image: null,
});
```

Ici la variable **form** contient un élément pour chaque **input** du formulaire, ajouter en autant que nécessaire, le but et récupérer toutes les données.

Ensuite le formulaire est envoyé au controller en utilisant la méthode **POST**.

```
function submit (){
    router.post('/surveys', form);
};
```

En plus d'une génération standard de données, une image doit être stocké dans l'application et le chemin de celle-ci doit être transmis au **survey** permettant plus tard d'afficher l'image qui lui est attribué.

```
public function store(StoreSurveyRequest $request)
{
    if($request->hasFile('image')){
        $path = Storage::disk('public')->putFile('image', new File($request->file('image')));
    }
    $survey = Survey::create(array_merge($request->validated(), [
        'title' => $request->title,
        'image' => $path ?? null,
        'id_user' => Auth::id(),
    ]));
    return redirect('/surveys/'.$survey->id);
}
```

Tout d'abord, une vérification est effectuée sur la présence de l'image fournit par l'utilisateur. Ensuite si celle-ci existe, l'application va la stocker et fournir le chemin d'accès à cette image. Puis une validation des informations envoyées par l'utilisateur est effectuée. Les conditions de validation se trouvent dans le fichier **StoreQuestionRequest.php** et sont gérées avec la fonction **rules**.

```
public function rules(): array
{
    return [
        'title' => ['required', 'min:5', 'max:100'],
        'image' => ['required'],
    ];
}
```

En spécifiant les éléments du formulaire et les conditions que ceux-ci doivent respecter. Si le formulaire passe la validation, le **survey** peut être créé, sinon un message d'erreur est retourné.

La fonctionnalité **READ** est la plus simple, celle-ci doit juste retourner le sondage choisi avec les questions et les réponses qui lui sont liées. Donc voici à quoi ressemble le code dans le controller :

```
public function show($id)
{
    $survey = Survey::where('id', $id)->get();
    $survey['user'] = User::where('id', $survey[0]->id_user)->get();
    $survey['questions'] = Question::where('id_survey', $id)->get();
    foreach($survey['questions'] as $question){
        $question['answer'] = Answer::where('id_question', $question->id)->get();
    }
    return Inertia::render('Survey/Survey', ['survey' => $survey]);
}
```

La fonctionnalité **UPDATE** ressemble beaucoup à la fonction **CREATE** du moins pour le frontend. Car le controller doit ici récupérer l'élément qui doit être modifié et lui appliquer ces modifications.

Pour le front, l'utilisation de **onBeforeMount** fourni par VueJs semble être parfaite, car celle-ci permet de définir des valeurs à certaines variables avant même que l'affichage soit chargé. Par exemple voici comment est prédéfini le titre du formulaire :

```
onBeforeMount(() => {
  form.title = props.survey[0].title;
});
```

Un point très important est que les méthodes **PUT** et **PATCH** ne fonctionne pas pour l'envoie d'image vers le controller. Donc la seule méthode permettant d'envoyer des données restante est la méthode **POST**.

```
function submit (){
  router.post('/surveys/' + props.survey[0].id, form);
};
```

Ensuite après avoir récupérer toutes les valeurs du formulaire (le fonctionnement est le même que pour la fonctionnalité **CREATE**). Le controller doit s'occuper des différentes modifications. Appliquer les modifications aux champs se retrouvant dans le formulaire.

```
public function update(UpdateSurveyRequest $request, $id)
{
  $survey = Survey::find($id);
  $request->validated();
  $survey->title = $request->title;
  if($request->hasFile('image')){
    $path = Storage::disk('public')->putFile('image', new File($request->file('image')));
    if($survey->image != null)
      Storage::disk('public')->delete($survey->image);
    $survey->image = $path;
  }
  $survey->update();

  return redirect('/surveys/'.$survey->id);
}
```

Le controller va vérifier si l'utilisateur à bien remplis tous les champs en procédant de la même manière que pour la fonctionnalités **CREATE** mais dans le fichier **UpdateSurveyRequest.php**.

```
public function rules(): array
{
  return [
    'title' => ['required', 'min:5', 'max:100'],
  ];
}
```

Ici uniquement le champ **title** est vérifier car le champ **image** peut se retrouver au pire vide, l'image existante n'est pas supprimée tant que l'utilisateur ne décide pas de mettre une autre image dans le formulaire.

La fonctionnalité **DELETE** peut être abordé de plusieurs façon, soit en supprimant tous les éléments 1 par 1 manuellement dans le controller, soit en ayant paramétrier une suppression des relations en cascade. La meilleure option dans cette situation semble être la 2ème.

Cette opération doit être effectuer lors de la création de la DB. Donc dans les fichiers de migration qui vont être impacté par la suppression. Dans ce cas se sont les fichiers **questions_tables.php**, **answers_tables.php** et **user_answers_tables.php**. La ligne permettant relation grâce à une FK doit ressembler à ceci :

```
$table->foreignId('id_survey')->constrained(table: 'surveys')->onUpdate('cascade')->onDelete('cascade');
```

Le premier paramètre est le nom du champ, le deuxième est le nom de table pour effectuer la relation. Ensuite ce sont les actions à effectuer en cas de mise à jour ou de suppression de l'éléments en lien.

Dans la fonction **destroy** ajouter ceci pour supprimer le sondage et l'image qui lui est liée dans l'application.

```
$survey = Survey::find($id);
if($survey->image != null){
    Storage::disk('public')->delete($survey->image);
}
$survey->delete();
```

De plus d'une suppression de classique, cette fonctionnalité doit avoir une double validation et lorsque la suppression a été effectuer la position du scroll doit être préserver.

Pour la validation de suppression, en utilisant le kit de démarrage Jetstream tous les éléments nécessaires sont déjà fourni. Donc pour éviter de faire le travail en double utiliser ces composants. Les composants utiles sont « **ConfirmationModal**, **DangerButton** et **SecondaryButton** ».

```
<ConfirmationModal :show="showConfirmDeleteQuestionModal" @close="closeModal">
    <template #title>
        <h2 class="text-lg font-semibold text-slate-800">Supprimer ce sondage ?</h2>
    </template>
    <template #content>
        <p class="text-lg font-semibold text-slate-600">{{ CurrentElement.title }}</p>
    </template>
    <template #footer>
        <div class="mt-6 flex space-x-4">
            <DangerButton @click="deleteQuestion(CurrentElement.id)">Supprimer</DangerButton>
            <SecondaryButton @click="closeModal">Annuler</SecondaryButton>
        </div>
    </template>
</ConfirmationModal>
<Pagination :meta="questions.meta"/>
```

Pour pouvoir utiliser correctement ces composants, ceux-ci ont besoin de variables ou fonctions à leur déclenchement. Et pour récupérer les informations de l'objet à supprimer, c'est transmis lorsque l'utilisateur appuie sur le bouton de suppression.

```
<button @click="openModal(item)" type="button">
    <Trash></Trash>
</button>
```

Finalement, pour préserver le scroll de la page. Celui-ci s'applique à l'envoie de la requête de suppression sur le front. En plus d'envoyer l'id de l'élément à supprimer, rajouter la variable **preserveScroll** en paramètre.

```
const deleteQuestion = (id) =>{
  router.delete('/questions/' + id, {preserveScroll: true});
  closeModal();
};
```

3.5 Graphique

Pour l'implémentation du graphique, une méthode assez simple pour générer des graphiques de qualités et de manière dynamique est d'utiliser la librairie **Chart.js**. Son installation a déjà été présenté dans **3.1 Mise en place de l'environnement**.

Le graphique s'affiche sur la **Homepage** de l'application. Donc pour y afficher un graphique importer sur la page ces éléments :

```
import Chart from 'chart.js/auto';
import { onBeforeMount, onMounted } from 'vue';
```

Le premier permet d'utiliser la classe **Chart** qui sert à la génération et l'affichage du graphique. **onBeforeMount** permet de charger des données avant la génération de l'affichage et **onMounted** va afficher le graphique.

```
const dataAnswers = {
  labels: [],
  datasets: [
    {
      type: 'bar',
      label: 'Nombre de personne ayant répondu au sondage',
      data: [],
      borderWidth: 1
    }
]
```

Ensuite la création d'une variable permettant par la suite de transmettre au graphique les données à afficher.

```
onBeforeMount(() => {
  if(props.surveys.data[0] != null){
    props.surveys.data.forEach(element => {
      dataAnswers.labels.push(element.title);
      dataAnswers.datasets[0].data.push(element.nbAnswer);
    });
  }
})
```

Pour permettre au graphique d'avoir accès aux données, la méthode **onBeforeMount** se charge de transmettre les données à la variable.

```

onMounted(() => {
  if(props.surveys.data[0] != null){
    const ctx = document.getElementById('myChart');
    const myChart = new Chart(ctx, {
      data: dataAnswers,
      options: {
        scales: {
          y: {
            beginAtZero: true
          }
        }
      }
    });
  }
})

```

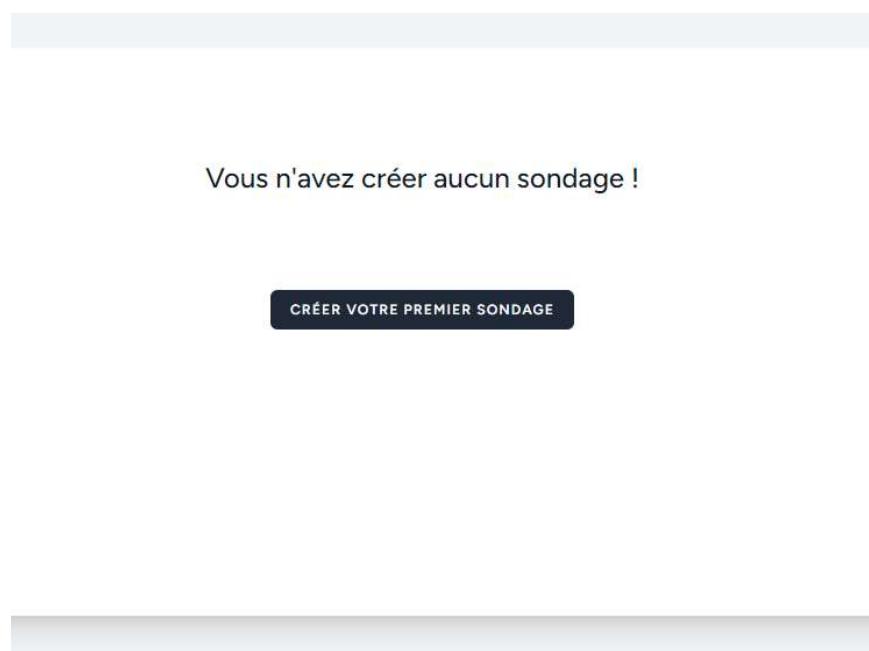
Ici, le graphique est créé avec les données qui viennent d'être récupérer. Cette méthode récupère aussi un élément html avec grâce à l'id **myChart**. Cette élément doit être de type **canvas** pour pouvoir afficher le graph.

```

<div v-if="surveys.data[0] == null">
  <h2 class="text-center text-2xl font-medium title-font mb-4 text-gray-900">
    Vous n'avez créer aucun sondage !
  </h2>
  <div class="mt-20 text-center">
    <PrimaryButton @click="createSurvey">Créer votre premier sondage</PrimaryButton>
  </div>
</div>
<canvas id="myChart"></canvas>

```

Voici à quoi ressemble la partie html permettant l'affichage, de plus si l'utilisateur n'a pas encore créer de sondage, le graph ne s'affiche pas mais à la place, du texte lui est afficher. Celui-ci lui dit qu'aucun sondage n'a été créée par l'utilisateur et un bouton est aussi afficher lui proposant de créer son premier sondage.



Jusqu'ici ce n'était que la partie front. Pour le backend, une information à savoir est qu'une des **goods practices** de l'utilisation de **Chart.js** est la création de son propre controller. Donc la création de **ChartController.php** est nécessaire. A l'intérieur de celui-ci, une seule fonction est nécessaire car l'application n'utilise qu'un seul graphique.

```
public function homepage()
{
    $surveys = Survey::where('id_user',Auth::id())->get();
    for($x = 0; $x < count($surveys);$x++){
        $surveys[$x]->user = count(User::where('id_survey',$surveys[$x]->id)->get());
    }
    $surveys = ChartResource::collection($surveys);
    return Inertia::render('HomePage', ['surveys' => $surveys]);
}
```

Cette fonction doit retourner uniquement les sondages créé par l'utilisateur et le nombre de personne ayant répondu au sondage. Dès que ces informations sont récupérées, celle-ci sont transmises au fichier **ChartResource.php** qui a été créer au préalable. Ce fichier sert à séparer les données inutiles dans certain cas pour éviter de surcharger le front avec des informations que celle-ci n'utilise pas comme par exemple ici : la date de création, la date de dernière modification ou encore l'image dans ce cas.

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'title' => $this->title,
        'nbAnswer' => $this->user,
    ];
}
```

Dans ce fichier la fonction **toArray** est celle qui va filtrer les données et retourner uniquement les valeurs choisies.

3.6 Rôles et permissions

L'application doit pouvoir gérer les accès de certain utilisateur grâce au rôle et aux permissions qui leur sont accordés. Donc comme pour les graphiques la librairie **Laravel-permission** permettant cette gestion est déjà installé.

Tout d'abord, récupérer les fichiers permettant la configuration et la création de nouveau rôles et de permission grâce à la commande :

```
php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"
```

Créer ensuite un fichier **Seeder** permettant la création des 2 rôles que vont contenir l'application.

```
public function run(): void
{
    $role = Role::create(['name' => 'user']);
    $permissions = Permission::create(['name' => 'user']);
    $role->syncPermissions($permissions);

    $role = Role::create(['name' => 'admin']);
    $permissions = Permission::create(['name' => 'admin']);
    $role->syncPermissions($permissions);
}
```

Ensuite pour que les utilisateurs puissent se voir attribuer un rôle, la modification du modèle de ceux-ci est nécessaire. Ajouter donc `use HasRoles;` cette élément permet de faire comprendre à la librairie que le modèle peut recevoir un rôle. N'oublier pas d'attribuer les rôles aux utilisateurs de tests dans le fichier **Seeder** des utilisateurs.

```
public function run(): void
{
    $users = User::factory(9)->create();
    foreach ($users as $user){
        $user->assignRole('user');
    }
    $users = User::factory(1)->create([
        'firstName'=>'admin',
        'lastName'=>'admin',
        'email'=>'admin@admin.ch'
    ]);
    foreach ($users as $user){
        $user->assignRole('admin');
    }
}
```

La prochaine étape est la vérification des accès permettant de restreindre les accès d'un certain rôle. Celle-ci peut s'effectuer à plusieurs endroits dans l'application. En premier, c'est dans le fichier de **route web.php**. Ce fichier permet d'attribuer les différentes requêtes **http** à une fonction d'un **controller** permettant par exemple l'affichage ou la création de données.

La vérification ici ce fait aux travers du **middleware**, en ajoutant par exemple cette élément qui va vérifier si l'utilisateur qui souhaite accéder à l'**url** « **/survey/create** » possède le rôle « **user** ». Si celui-ci dispose belle et bien de ce rôle, l'accès lui est autoriser, sinon l'application lui renvoie un statut 403 lui expliquant que l'utilisateur ne dispose pas du rôle nécessaire pour accéder à cette page.

```
Route::middleware(['auth:sanctum', config('jetstream.auth_session'), 'verified', 'role:user'],
]->group(function () {
    Route::get('/surveys/create', [SurveyController::class, 'create'])->name('surveys.create');
```

Mais pour pouvoir vérifier le rôle de l'utilisateur, la modification du **middleware** est nécessaire. Celle-ci s'effectue en ajoutant l'alias « **role** » dans le fichier **app.php**.

```
->withMiddleware(function (Middleware $middleware) {
    $middleware->web(append: [
        \App\Http\Middleware\HandleInertiaRequests::class,
        \Illuminate\Http\Middleware\AddLinkHeadersForPreloadedAssets::class,
    ]);
    $middleware->alias([
        'role' => \Spatie\Permission\Middleware\RoleMiddleware::class,
    ]);
})
```

Pour utiliser les rôles dans le **controller** c'est beaucoup plus simple. En ajoutant la référence `use Illuminate\Support\Facades\Auth;` dans le fichier désirer. Cela permet à l'application de récupérer les informations de l'utilisateur qui est actuellement connecté donc savoir son rôle. Ensuite vérifier son rôle grâce à l'objet « **Auth** ». Voici un exemple dans le fichier **SurveyController.php**.

```
public function index()
{
    if(Auth::user()->can('admin'))
    {
        $surveys = SurveyResource::collection(Survey::paginate(12));
        return Inertia::render('Admin/SurveyDashboard', ['surveys' => $surveys]);
    }else{
        $surveys = SurveyResource::collection(Survey::where('id_user',Auth::id())->paginate(12));
        return Inertia::render('Survey/SurveyDashboard', ['surveys' => $surveys]);
    }
}
```

Le premier cas retourne toutes les données de la table **surveys** vu que l'utilisateur actuellement connecté est un administrateur. Dans l'autre cas, les données retourner sont uniquement celles que l'utilisateur à créer.

Le dernier endroit où utiliser ces fichiers est dans le **frontend** avec **Vuejs**. Ici, c'est assez simple de vérifier le rôle d'un utilisateur. En rajoutant un simple **if** qui a comme condition que l'utilisateur actuel doit avoir le rôle cité. Presque toutes les affichages dispose de cette information grâce à « **\$page.props.auth.user** » qui contient les informations de l'utilisateur. Voici comment c'est appliqué dans le fichier **AppLayout.vue** qui sert de **Header** :

```
<div class="ms-3 relative" v-if="$page.props.auth.user.roles[0].name == 'admin'">
```

Voici l'affichage à quoi ressemble le **Header** d'un utilisateur ayant le rôle « **admin** » :



Et voici celui d'un utilisateur ne disposant pas de ce rôle :



3.7 Description des tests effectués

On sait depuis longtemps que travailler avec du texte lisible et contenant du sens est source de distractions, et empêche de se concentrer sur la mise en page elle-même. L'avantage du Lorem Ipsum sur un texte générique comme 'Du texte. Du texte. Du texte.' est qu'il possède une distribution de lettres plus ou moins normale,

3.8 Problèmes rencontrer

On sait depuis longtemps que travailler avec du texte lisible et contenant du sens est source de distractions, et empêche de se concentrer sur la mise en page elle-même. L'avantage du Lorem Ipsum sur un texte générique comme 'Du texte. Du texte. Du texte.' est qu'il possède une distribution de lettres plus ou moins normale,

4. Conclusion

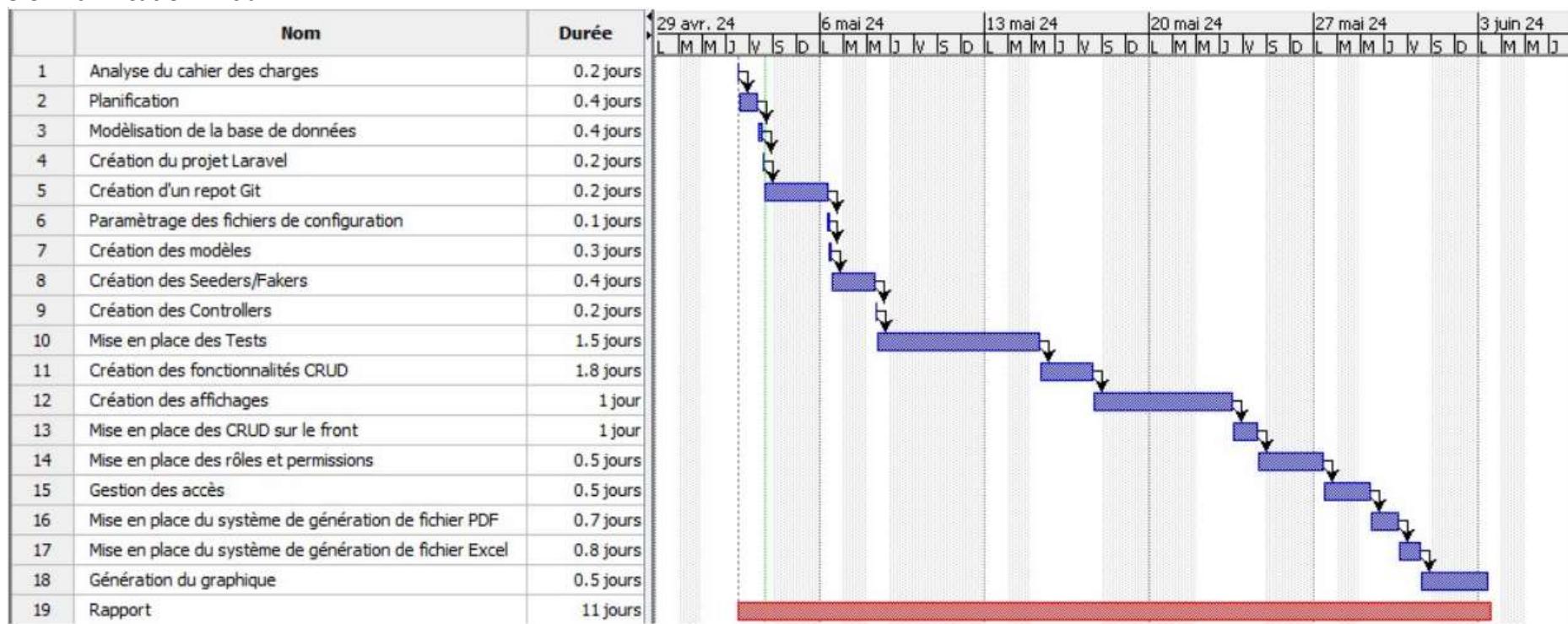
5. Annexes

5.1. Résumé du rapport du TPI

On sait depuis longtemps que travailler avec du texte lisible et contenant du sens est source de distractions, et empêche de se concentrer sur la mise en page elle-même. L'avantage du Lorem Ipsum sur un texte générique comme 'Du texte. Du texte. Du texte.' est qu'il possède une distribution de lettres plus ou moins normale,

5.2. Sources

5.3. Planification Initial



5.4. Mockup

The image shows a wireframe mockup of a login interface. At the top center, the word "Connexion" is displayed. Below it, there are two input fields: one labeled "Email" and another labeled "Mot de passe". Underneath the password field is a checkbox labeled "Se souvenir de moi". At the bottom right of the form, there are two buttons: a white button with the text "Créer un compte" and a blue button with the text "Connexion".

Connexion

Email

Mot de passe

Se souvenir de moi

[Créer un compte](#) **Connexion**

Créer un compte

Prénom

Nom

Email

Mot de passe

Confirmer le mot de passe

[Vous avez déjà un compte ?](#)

Créer

[Se connecter](#)

Nom d'utilisateur

Votre inscription à été envoyé et est en attente de validation.

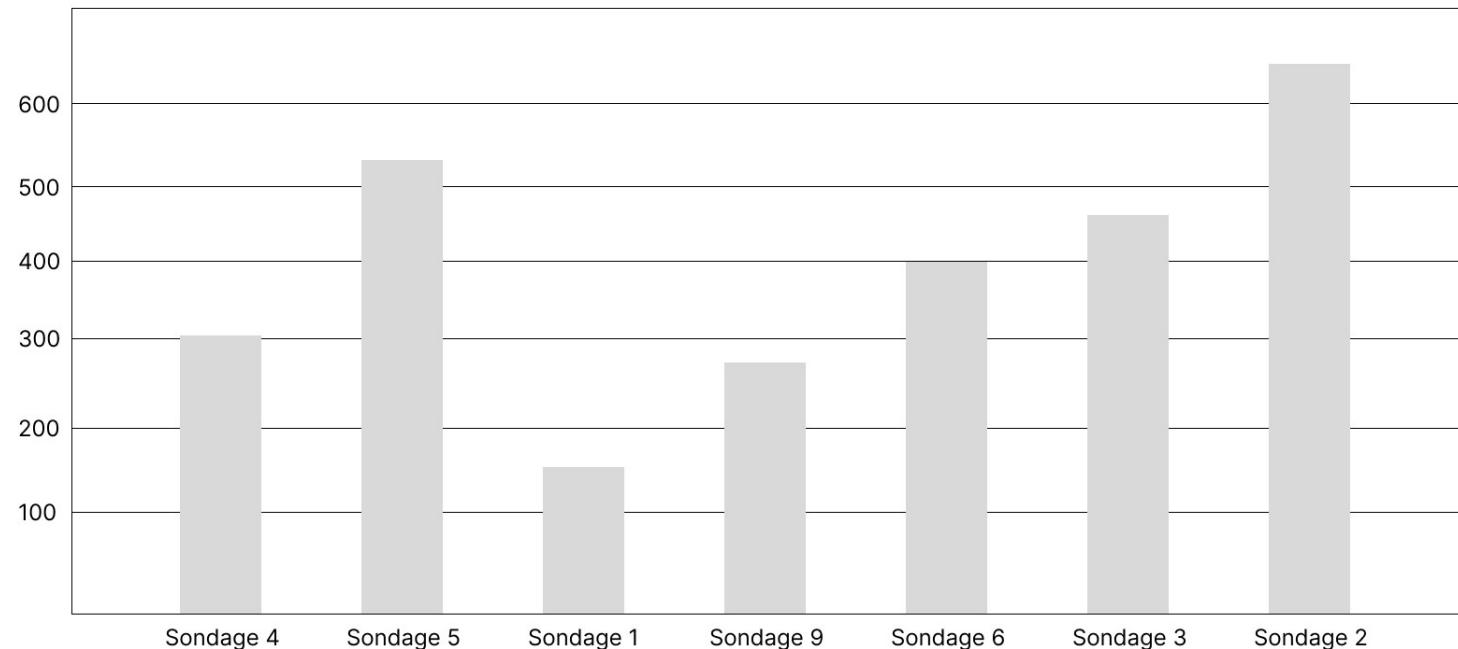


Vos sondages

Sondages

Nom utilisateur

Vos sondages





Vos sondages

Sondages

Nom utilisateur

Gestion des sondages

Créer un sondage

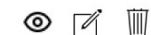
Sondage numéro 1



Sondage numéro 2



Sondage numéro 3



Sondage numéro 4



<< Précédent 1 2 3 4 Suivant >>

The screenshot shows a user interface for managing surveys. At the top, there is a navigation bar with a logo, the text "Vos sondages" and "Sondages" (underlined), and a "Nom utilisateur". Below the navigation, there is a section titled "Gestion des sondages" with a "Créer" button. A modal dialog box is centered on the screen, asking "Supprimer ce sondage ?" (Delete this survey?). Inside the dialog, there is a text input field labeled "Titre du sondage" containing the placeholder "Titre du sondage". At the bottom of the dialog are two buttons: a red "Supprimer" (Delete) button and a grey "Annuler" (Cancel) button. Below the dialog, four survey entries are listed in a grid format:

Sondage numéro 1			
Sondage numéro 2			
Sondage numéro 3			
Sondage numéro 4			

At the bottom of the page, there is a navigation bar with links: "<< Précédent", "1", "2", "3", "4", "Suivant >>".



Vos sondages

Sondages

Nom utilisateur

Modifier un sondage

Modifier un sondage

Titre

Titre sondage 1

Image

Image sondage 1

Modifier



Vos sondages

Sondages

Nom utilisateur

Créer un nouveau sondage

Création d'un nouveau sondage

Titre

Image

Créer



Vos sondages

[Sondages](#)

Nom utilisateur

Contenu du sondage

[Voir les résultats](#)

[Gérer les questions](#)

Nom du sondage

Image du sondage

Questions

Aucun questions n'existent



Vos sondages

Sondages

Nom utilisateur

Résultat du sondage

Voir le sondage

Voir les questions

Réponse du sondage Nom du sondage

Question numéro 1

Réponse A	Réponse B	Réponse C	Réponse D
Nombre de réponse	Nombre de réponse	Nombre de réponse	Nombre de réponse

Question numéro 2

Réponse numéro 1

Réponse numéro 2

Réponse numéro 3

Réponse numéro 4

<< Précédent 1 2 3 4 Suivant >>

Question numéro 3

Réponse 1

Nombre de réponse

Réponse 2

Nombre de réponse

Réponse 3

Nombre de réponse

Répondre



Vos sondages

Sondages

Nom utilisateur

Gestions des questions

Titre du sondage

[Créer une question](#)

[Voir le sondage](#)

Question numéro 1



Question numéro 2



Question numéro 3



Question numéro 4



<< Précédent 1 2 3 4 Suivant >>



Vos sondages

Sondages

Nom utilisateur

Créer une nouvelle question

Création d'une nouvelle question

Titre de la question

Type de la question 1

Text

Select box

Select Multiple Box

Contenu de la question 1

Créer



Vos sondages

Sondages

Nom utilisateur

Créer une nouvelle question

Création d'une nouvelle question

Titre de la question

Type de la question 1

Text

Select box

Select Multiple Box

Nombre de réponse

▼

Réponse 1

Contenu de la question 1

Créer



Vos sondages

[Sondages](#)

Nom utilisateur

Contenu du sondage

[Voir les résultats](#)

[Gérer les questions](#)

Nom du sondage

Image du sondage

Questions

Question numéro 1

Question numéro 2

Question numéro 3

Réponse 1

Réponse 2

Réponse 3



Vos sondages Sondages

Nom utilisateur

Modifier votre profile

Modifier votre profile

Nom

Nom de l'utilisateur

Prénom

Adresse email

Modifier

Modifier votre MDP

Mot de passe actuelle

Nouveau mot de passe

Confirmer le mot de passe

Modifier

Vue Admin



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Gestion des utilisateurs

Gérer les inscriptions

[Créer un utilisateur](#)

[Exporter les utilisateurs](#)

Utilisateur numéro 1



Utilisateur numéro 2



Utilisateur numéro 3



Utilisateur numéro 4



<< Précédent 1 2 3 4 Suivant >>



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Exporter les créateurs

Exporter les créateurs au format Excel

Quel champs souhaitez vous exporter :

Nombre de sondage

Nom

Prénom

Exporter



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Gestion des utilisateurs

Gérer les inscriptions

Nouveau utilisateur numéro 1

Nouveau utilisateur numéro 2

Nouveau utilisateur numéro 3

Nouveau utilisateur numéro 4

<< Précédent 1 2 3 4 Suivant >>

Vos sondages Sondages Utiliseurs

Nom utilisateur

Gestion des utilisateurs

Rejeter cette inscription ?

Nom et prénom de l'utilisateur

Rejeter Annuler

Nouveau utilisateur numéro 1

Nouveau utilisateur numéro 2

Nouveau utilisateur numéro 3

Nouveau utilisateur numéro 4

<< Précédent 1 2 3 4 Suivant >>

Vos sondages Sondages Utilisateurs

Nom utilisateur

Gestion des utilisateurs

Valider cette inscription ?

Nom et prénom de l'utilisateur

Nouveau utilisateur numéro 1

Nouveau utilisateur numéro 2

Nouveau utilisateur numéro 3

Nouveau utilisateur numéro 4

<< Précédent 1 2 3 4 Suivant >>

Valider **Annuler**



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Créer un utilisateur

Créer une nouvelle utilisateur

Nom

Prénom

Email

Mot de passe

Confirmer le mot de passe

Créer



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Modifier l'utilisateur

Modifier l'utilisateur

Nom

Prénom

Adresse email

Modifier

Modifier le MDP de l'utilisateur

Nouveau mot de passe

Confirmer le mot de passe

Modifier



Vos sondages

Sondages

Utilisateurs

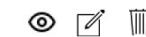
Nom utilisateur

Gestion des sondages

[Créer un sondage](#)

[Exporter les sondages](#)

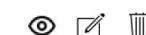
Sondage numéro 1



Sondage numéro 2



Sondage numéro 3



Sondage numéro 4



<< Précédent 1 2 3 4 Suivant >>



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Exporter les sondages

Exporter les sondages au format PDF

Quel champs souhaitez vous exporter :

Titre Image

Exporter



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Résultat du sondage

Voir le sondage

Voir les questions

Réponse du sondage Nom du sondage

Question numéro 1

Réponse A Nombre de réponse	Réponse B Nombre de réponse	Réponse C Nombre de réponse	Réponse D Nombre de réponse
--------------------------------	--------------------------------	--------------------------------	--------------------------------

Question numéro 2

Réponse numéro 1	
Réponse numéro 2	
Réponse numéro 3	
Réponse numéro 4	

<< Précédent 1 2 3 4 Suivant >>

Question numéro 3

Réponse 1 Nombre de réponse	Réponse 2 Nombre de réponse	Réponse 3 Nombre de réponse
--------------------------------	--------------------------------	--------------------------------

Répondre



Vos sondages

Sondages

Utilisateurs

Nom utilisateur

Modifier une réponse

Modification d'une réponse

Contenu de la réponse

Contenu Réponse 1

Modifier

Vue Répondant

Nom du sondage
Image du sondage

Vos informations

Nom Prénom

Ville

Questions

Question numéro 1

Question numéro 2

Question numéro 3

Réponse 1 Réponse 2 Réponse 3

Répondre

5.5. Journal de Travail

On sait depuis longtemps que travailler avec du texte lisible et contenant du sens est source de distractions, et empêche de se concentrer sur la mise en page elle-même. L'avantage du Lorem Ipsum sur un texte générique comme 'Du texte. Du texte. Du texte.' est qu'il possède une distribution de lettres plus ou moins normale,