



# Projet final

# bigscreen

Réalisé Par :

GASSAMA Adama  
CONTE Kankou

Octobre 2022

## **Plan de travail :**

- 1- Description de projet
- 2- Technologies utilisées
- 3- Déploiement
- 4- Méthodologie de travail
- 5- Outils utilisés
- 6- Fonctionnalités
- 7- Temps de travail
- 8- Recettage
- 9- Documentation API
- 10- Modélisation
- 11- Wireframes

# I. Description de projet :

**Bigscreen** est une entreprise qui met à la disposition de ses clients une application de **réalité virtuelle** leur permettant de regarder des films, émissions TV et de jouer à des jeux vidéo sur un écran géant virtuel, qu'ils soient seuls, en famille ou entre amis.

Dans le souci d'apporter plus d'amélioration à son application, l'entreprise **Bigscreen** souhaiterait mettre en place un sondage pour recueillir l'avis de ses utilisateurs.

C'est dans ce sens que l'entreprise nous a confié la responsabilité de développer une application de sondage destinée à leurs utilisateurs pour permettre de connaître leur avis.

# II. Technologies Utilisées :

Pour la réalisation de ce projet, nous avons principalement développé deux grandes parties, à savoir :

## - Le côté **Back-end** :

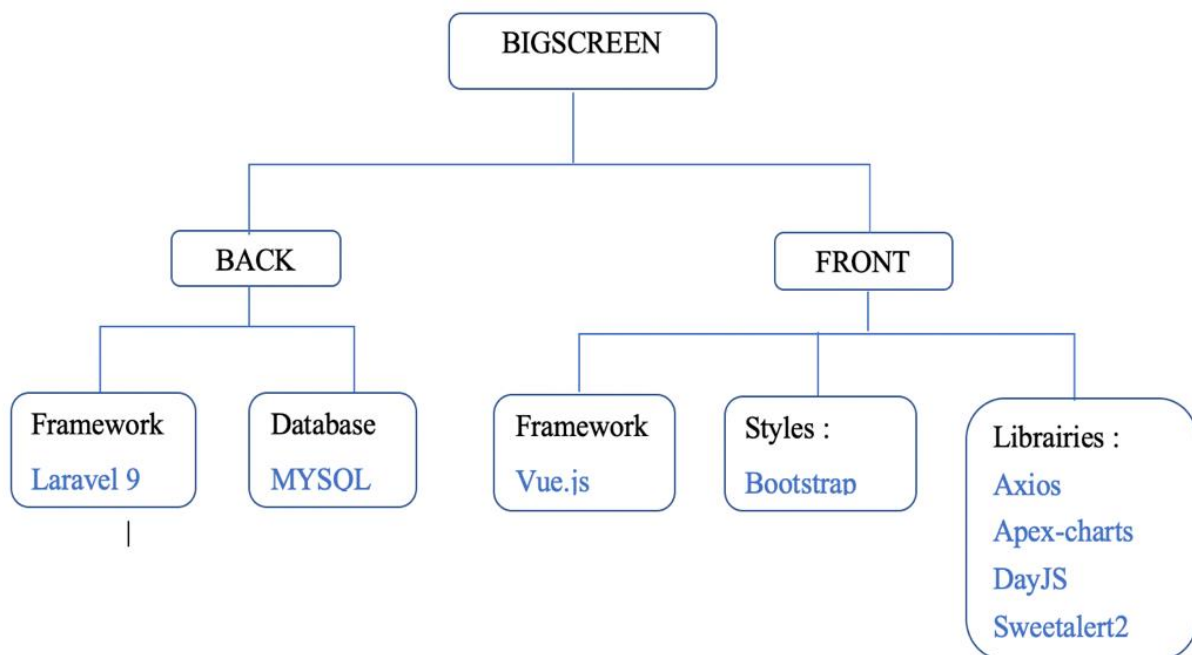
- Nous avons conçu une **API Rest** avec le framework **Laravel** qui possède l'ORM eloquent.
- Nous avons utilisé **Mysql Server** pour la gestion de notre base de données.

## - Le côté **Front-end** :

- Nous avons utilisé le framework **Vues.js** qui nous donne la possibilité de concevoir des composants et de faciliter l'intégration sur les différentes pages.
- Nous avons utilisé la librairie **Axios** pour faire les requêtes sur le serveur. Elle est pratique à mettre en place et les réponses sont facilement exploitables sur le front.

- Nous avons utilisé le framework **Bootstrap** pour faire les mises en forme CSS de nos pages.
- Nous avons utilisé la librairie **Apex-chart.js** pour l’affichage des PieChart et RadarChart. Elle est plus facile à intégrer avec VueJs et donne un design bien joli sur la page Admin.
- Nous avons utilisé la librairie **DayJS** pour faire le formatage de la date d’enregistrement de chaque sondage réalisé par un client.
- Nous avons utilisé la librairie **Sweetalert2** pour la gestion du modal qui affiche le lien permettant au visiteur de consulter son résultat.

Voici un schéma explicatif de toutes les technologies que nous avons utilisées.



### III. Déploiement :

Récupération de notre code source sur notre repos github en suivant le lien :

<https://github.com/adagassama/BigScreen>

Déploiement de notre application pour qu'elle fonctionne sur votre machine, veuillez suivre les instructions ci-dessous :

### A faire coté server :

- Créer une Base de données appelé "survey"
- Créer le fichier.env et associer à la BD

### Pour installer les dépendances du projet coté client

- \$cd Back && composer install
- \$cd Front && npm install

### Pour installer les migrations et seeders du projet coté server

- php artisan migrate:fresh --seed

### Ensuite pour lancer le coté server

- \$cd Back / php artisan serve

### Ensuite pour lancer le coté client

- \$cd Front / npm run dev

### Pour la connexion de l'utilisateur coté client :

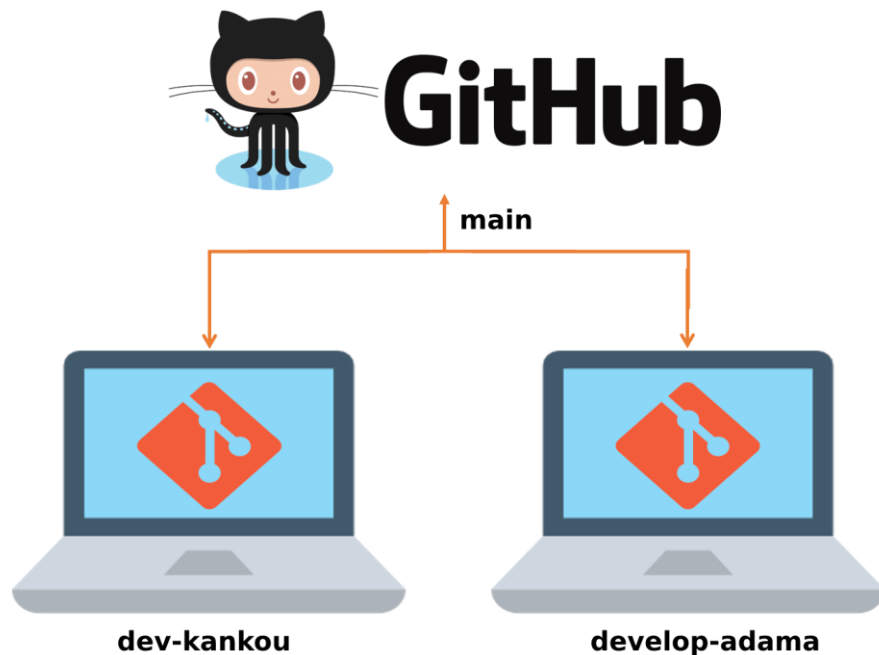
- User: [admin@gmail.com](mailto:admin@gmail.com)
- Mdp: password

## IV. Méthodologie de Travail :

En équipe de deux développeurs, nous avons conçu **Bigscreen** en utilisant la méthode **SCRUM**, en prenant le soin de respecter toutes les étapes du cahier de charge mis à notre disposition par le client :

### Procédure de commit :

Nous avons créé un GitHub avec 2 contributeurs et chaque développeur avait sa branche locale.



Comme illustré sur l'image ci-dessus, une fois que mon collègue finit de coder une fonctionnalité sur sa branche locale il fait son commit et push sur la branche main, ensuite moi je récupère les modifications et pull sur ma branche locale, et vice versa.

## V. Outils Utilisés :

Pour mener à bien nos différentes tâches, nous avons utilisé plusieurs outils à savoir :

- **VSCode** : qui nous a servi d'éditeur de code pour la conception
- **MAMP** : pour la gestion de la base de données MySQL
- **Taskade** : pour la gestion de nos tâches
- **GitHub** : pour le versionning du projet
- **Mobile First** : pour les tests de Responsivités
- **Creately** : pour la modélisation de notre base de données
- **Google Slide** : pour la présentation
- **Word** : pour la rédaction du cahier de charge

## VI. Fonctionnalités :

En tant que développeur, nous avons implémenté plusieurs fonctionnalités pour notre application, ci-dessous, vous trouverez chaque fonctionnalité et leurs importances.

### 1) BACK :

#### 1.1) **QuestionController : index()**

Nous avons développé cette fonction pour récupérer toutes les **questions du Sondage**.

#### 1.2) **AnswerController : index()**

Nous avons implémenté cette méthode pour récupérer **toutes les réponses** des différents utilisateurs et leurs questions respectives.

#### 1.3) **AnswerController : store()**

Cette fonction permet **d'enregistrer** les réponses d'un sondage, nous avons mis en place un système de **Validation** pour voir si toutes les règles que nous avons établies sont respectées, si tel n'est pas le cas des messages d'erreurs seront générés, sinon le sondage sera bien validé et les réponses de cet utilisateur seront stockées dans la table **Answer**.

#### **1.4) AnswerController : getVisitorResponse()**

Permet d'obtenir les réponses de chaque visiteur via un **token**.

#### **1.5) VisitorController : index()**

Permet de voir la liste de tous les **visiteurs ayant participé** au sondage.

#### **1.6) BackController : getPieChart()**

Permet de retourner les **statistiques** des questions 6, 7, et 10 dans les charts graphiques de type **Pie chart**.

#### **1.7) BackController : getRadarChart()**

Permet de fournir les données nécessaires à la conception du **Radar chart** pour les questions (11, 12, 13, 14, 15).

#### **1.8) AuthController : login()**

Nous avons géré l'**authentification** de l'administrateur dans cette fonction.

#### **1.9) AuthController : logout()**

Fonction de gestion de la **déconnexion** de l'administrateur.



## 2) FRONT :

### 2.1) Interface de connexion :

En tant qu'Admin, je souhaite accéder au tableau de bord de l'administration, donc je connecte via le formulaire de login que nous avons mis en place.

### 2.2) Page de Sondage :

Représenter par le composant **FormSurvey** dans vue.js, cette page liste l'ensemble des questions dans un formulaire stylisé en Bootstrap afin de permettre aux visiteurs de participer au sondage.

### 2.3) Page Réponse :

Représenter par le composant **ResponseSurvey** dans vue.js, permet à un visiteur via une **URL** que nous lui fournissons à la fin de son sondage, de pouvoir visionner toutes les réponses qu'il a apportées aux différentes questions.

### 2.4) Page d'Accueil du Dashboard :

Une fois que nous sommes connectés en tant qu'administrateur, nous sommes dirigés vers la page d'accueil du tableau de bord, cette page d'accueil comportera les **statistiques des questions (6, 7, 10, 11,12, 13,14 et 15)** représenter sous forme de **chart graphiques**. Sur le côté gauche un **menu dynamique** qui nous permettra d'accéder aux autres fonctionnalités du tableau de bord.

### 2.5) Page Questionnaire du Dashboard :

Connecté en tant qu'administrateur, nous pouvons accéder à un tableau listant toutes les **questions** du sondage.

### 2.6) Page Reponse du Dashboard :

Connecté en tant qu'administrateur, nous pouvons accéder à un tableau listant toutes les **questions et leurs réponses**

selon l'utilisateur et ce tableau évolue selon le nombre de sondage.

## VII. Temps de Travail :

<b>BACK :</b>	Temps de travail
Création de projet & base de données	1 jour
Migration des tables	2 jours
Factory, seeders et Model	1jour
Fonction login (AuthController)	0,5 jour
Fonction logout (AuthController)	0,5 jour
Fonction index (QuestionController)	0,5 jour
Fonction index (AnswerController)	1 jour
Fonction store (AnswerController)	4 jours
Fonction getVistorResponse (AnswerController)	2 jours
Fonction index (VisitorController)	0,5 jour
Fonction getPieChart (BackController)	1 jour
Fonction getRadarChart (BackController)	1 jour
<b>FRONT :</b>	
Formulaire d'authentification & sa méthode	2 jours
Page Sondage	4 jours
Page de Réponses	2jours
Page d'Accueil du Dashboard	3 jours
Page Questionnaire du Dashboard	1 jour
Page de Réponse du Dashboard	2 jours
Pop-up de validation	0,5 jour
Responsives	2 jours
Mise en forme des pages	2 jours

## VIII. Recettage:

<b>BACK :</b>	Niveau
Création de projet & base de données	Terminé
Migration des tables	Terminé
Factories, seeders et Models	Terminé
Fonction login (AuthController)	Terminé
Fonction logout (AuthController)	Terminé
Fonction index (QuestionController)	Terminé
Fonction index (AnswerController)	Terminé
Fonction store (AnswerController)	Terminé
Fonction getVistorResponse (AnswerController)	Terminé
Fonction index (VisitorController)	Terminé
Fonction getPieChart (BackController)	Terminé
Fonction getRadarChart (BackController)	Terminé
	Terminé
<b>FRONT :</b>	Terminé
Formulaire d'authentification & sa méthode	Terminé
Page Sondage	Terminé
Page de Réponses	Terminé
Page d'Accueil du Dashboard	Terminé
Page Questionnaire du Dashboard	Terminé
Page de Réponse du Dashboard	Terminé
Pop-up de validation	Terminé
Responsives	Terminé
Mise en forme des pages	Terminé
Gestions des erreurs	Terminé
Commentaires	Terminé

## IX. Documentation API :

### Liste des questions

Méthode : **GET**

Paramètres : Aucun

URL : <http://127.0.0.1:8000/api/questions>

Exemple de Réponse :



```
{
  success: true,
  message: "List des questions",
  data: [
    {
      id: 1,
      title: "Question 1/20",
      content: "Votre adresse mail",
      type: "B",
      possible_answer: "",
      created_at: "2022-09-23T11:23:19.000000Z",
      updated_at: "2022-09-23T11:23:19.000000Z"
    },
    {
      id: 2,
      title: "Question 2/20",
      content: "Votre age",
      type: "B",
      possible_answer: "",
      created_at: "2022-09-23T11:23:19.000000Z",
      updated_at: "2022-09-23T11:23:19.000000Z"
    },
    {
      id: 3,
      title: "Question 3/20",
      content: "Votre sexe",
      type: "A",
      possible_answer: "Homme,Femme,Préfère ne pas répondre",
      created_at: "2022-09-23T11:23:19.000000Z",
      updated_at: "2022-09-23T11:23:19.000000Z"
    },
    {
      id: 4,
      title: "Question 4/20",
      content: "Nombre de personne dans votre foyer (adulte & enfants)",
      type: "C",
      possible_answer: "",
      created_at: "2022-09-23T11:23:19.000000Z",
      updated_at: "2022-09-23T11:23:19.000000Z"
    }
  ]
}
```

# Liste de toutes les réponses

Méthode : **GET**

Paramètres : Aucun

URL : <http://127.0.0.1:8000/api/answers>

Exemple de Réponse :



```
{
  success: true,
  message: "Liste de toutes les réponses",
  data: [
    - {
      id: 1,
      answer: "aaaaggg@gmail.com",
      question_id: 1,
      visitor_id: 1,
      created_at: "2022-09-23T11:26:05.000000Z",
      updated_at: "2022-09-23T11:26:05.000000Z",
      - question: {
        id: 1,
        title: "Question 1/20",
        content: "Votre adresse mail",
        type: "B",
        possible_answer: "",
        created_at: "2022-09-23T11:23:19.000000Z",
        updated_at: "2022-09-23T11:23:19.000000Z"
      }
    },
    - {
      id: 2,
      answer: "33",
      question_id: 2,
      visitor_id: 1,
      created_at: "2022-09-23T11:26:05.000000Z",
      updated_at: "2022-09-23T11:26:05.000000Z",
      - question: {
        id: 2,
        title: "Question 2/20",
        content: "Votre age",
        type: "B",
        possible_answer: "",
        created_at: "2022-09-23T11:23:19.000000Z",
        updated_at: "2022-09-23T11:23:19.000000Z"
      }
    },
    - {
      id: 3,
```

# Affichage de la réponse d'un seul visiteur

Méthode : **GET**

Paramètres : url (unique à chaque visiteur)

URL : <http://127.0.0.1:8000/api/results/hWpmfrFCfac2sQjdVIbQ>

Exemple de Réponse :



The screenshot shows a web browser window with the address bar displaying the URL `127.0.0.1:8000/api/results/hWpmfrFCfac2sQjdVIbQ`. The browser's developer tools are open, showing a JSON response. The JSON is a nested object with a `success` field set to `true`, a `message` field set to `"Réponses Visitor"`, and a `data` array containing two objects. Each object represents a visitor's response, including fields for `id`, `answer`, `question_id`, `visitor_id`, `created_at`, `updated_at`, and a `question` object. The first visitor (id: 81) answered "testt@gmail.net" for question 1. The second visitor (id: 82) answered "23" for question 2. Both responses were created and updated on 2022-10-03 at 10:14:02.000000Z.

```
{
  success: true,
  message: "Réponses Visitor",
  data: [
    {
      id: 81,
      answer: "testt@gmail.net",
      question_id: 1,
      visitor_id: 5,
      created_at: "2022-10-03T10:14:02.000000Z",
      updated_at: "2022-10-03T10:14:02.000000Z",
      question: {
        id: 1,
        title: "Question 1/20",
        content: "Votre adresse mail",
        type: "B",
        possible_answer: "",
        created_at: "2022-09-23T11:23:19.000000Z",
        updated_at: "2022-09-23T11:23:19.000000Z"
      }
    },
    {
      id: 82,
      answer: "23",
      question_id: 2,
      visitor_id: 5,
      created_at: "2022-10-03T10:14:02.000000Z",
      updated_at: "2022-10-03T10:14:02.000000Z",
      question: {
        id: 2,
        title: "Question 2/20",
        content: "Votre age",
        type: "B",
        possible_answer: "",
        created_at: "2022-09-23T11:23:19.000000Z",
        updated_at: "2022-09-23T11:23:19.000000Z"
      }
    }
  ]
}
```

# Connexion

Méthode : **POST**

Paramètres : email, password et device\_name (browser par défaut)

URL : <http://127.0.0.1:8000/api/login>

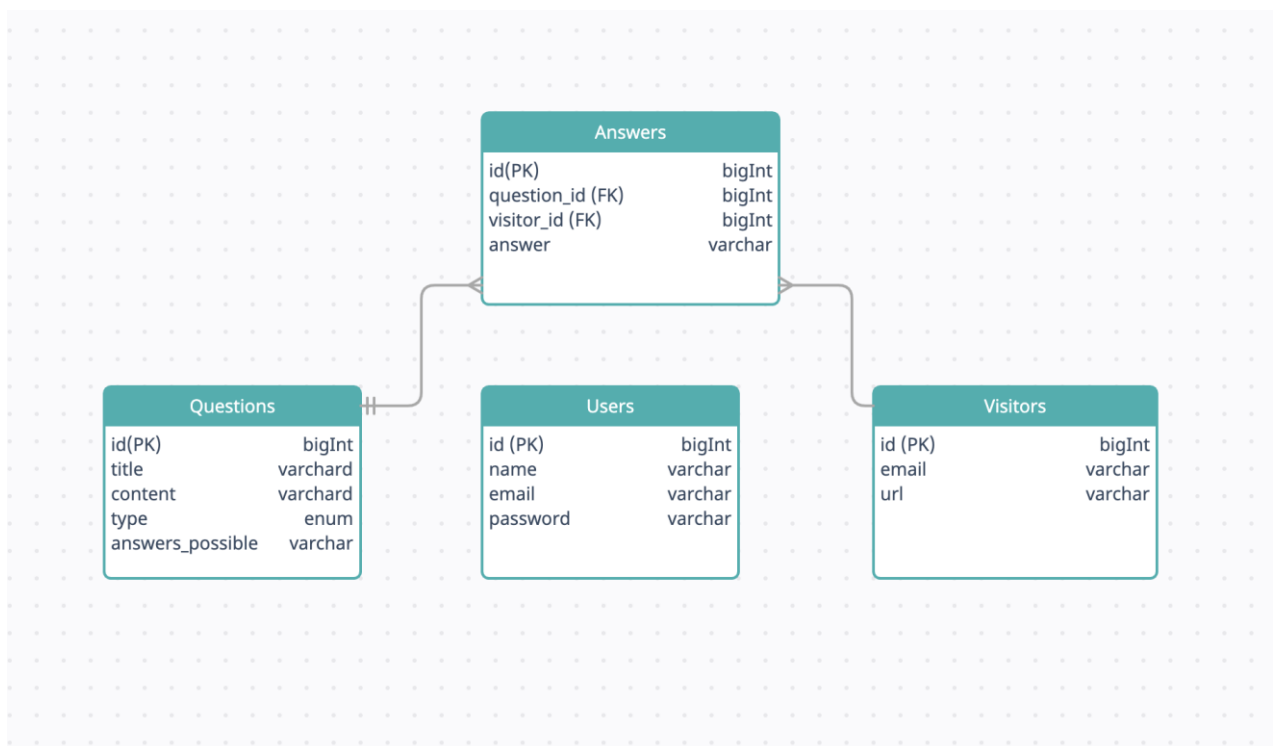
# Déconnexion

Méthode : **POST**

Paramètres : Aucun

URL : <http://127.0.0.1:8000/api/logout>

## X. Modélisation des tables :



# XI. Wireframes :

## Page de Sondage

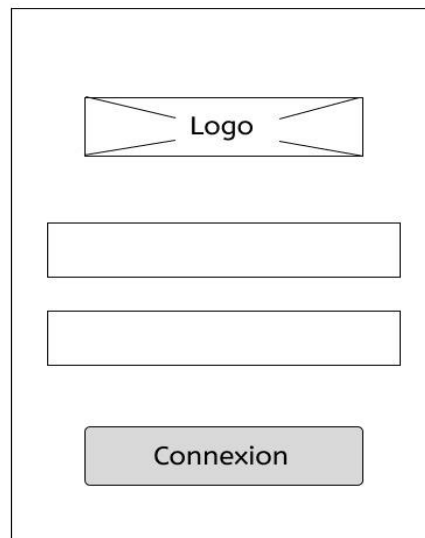
The wireframe for the 'Page de Sondage' (Survey Page) is enclosed in a rectangular border. At the top, there is a header area containing a 'Logo' placeholder, represented by a rectangle with diagonal lines. Below the header, there are two horizontal lines. The main content area contains two question blocks. The first block is labeled 'Question 1/20' and includes three horizontal lines for text input and a large dashed rectangular box for a response. Below this block are four small square checkboxes arranged vertically. The second block is labeled 'Question 20/20' and also includes three horizontal lines for text input and a large dashed rectangular box for a response. At the bottom right of the page, there is a grey button labeled 'Finaliser'.

## Page Réponses

The wireframe for the 'Page Réponses' (Results Page) is enclosed in a rectangular border. It features a header area with a 'Logo' placeholder (a rectangle with diagonal lines) and two horizontal lines below it. The main content area contains two question blocks. The first block is labeled 'Question 1/20' and includes three horizontal lines for text input and a large dashed rectangular box for a response. Below this block are four small square checkboxes arranged vertically. The second block is labeled 'Question 20/20' and also includes three horizontal lines for text input and a large dashed rectangular box for a response.

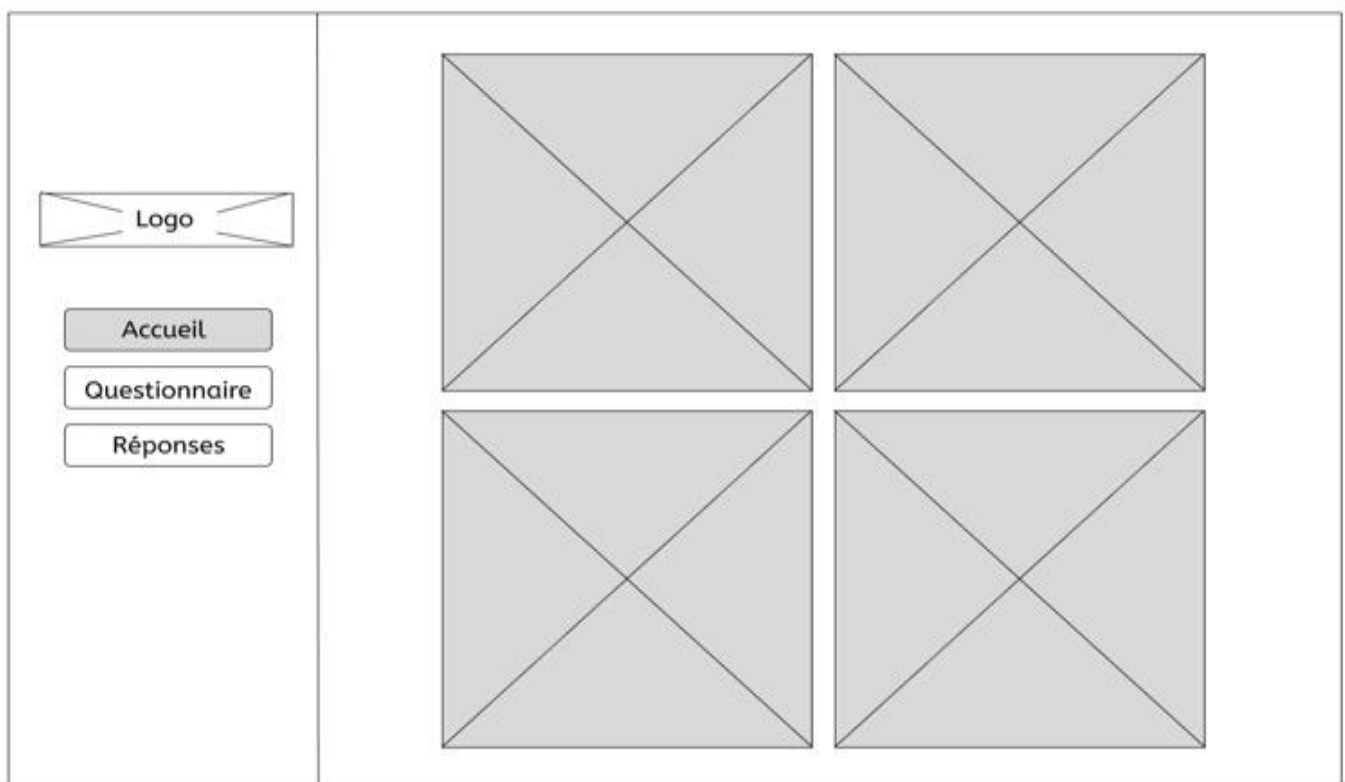


## Formulaire de Connexion



A login form layout within a rectangular container. At the top is a logo placeholder, represented by a rectangle with the word "Logo" in the center and two diagonal lines forming a stylized 'X'. Below the logo are two empty rectangular input fields for text. At the bottom is a gray button with the text "Connexion".

## Accueil Dashboard



A dashboard layout divided into two main sections. The left section is a sidebar containing a logo placeholder (rectangle with "Logo" and a stylized 'X') and three buttons: "Accueil" (gray), "Questionnaire" (white), and "Réponses" (white). The right section is a main content area containing four large square placeholders, each with a gray background and a black 'X' formed by two diagonal lines, representing missing images or charts.

## Tableau de Questionnaire

<div>Logo</div> <div>Accueil</div> <div>Questionnaire</div> <div>Réponses</div>	<table border="1"><thead><tr><th>#</th><th>Corps</th><th>Type</th></tr></thead><tbody><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></tbody></table>			#	Corps	Type																					
	#	Corps	Type																								

## Tableau des Questions et leurs réponses

<div>Logo</div> <div>Accueil</div> <div>Questionnaire</div> <div>Réponses</div>	<table border="1"><thead><tr><th>#</th><th>Corps</th><th>Réponses</th></tr></thead><tbody><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></tbody></table>			#	Corps	Réponses															
	#	Corps	Réponses																		
<table border="1"><thead><tr><th>#</th><th>Corps</th><th>Réponses</th></tr></thead><tbody><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></tbody></table>			#	Corps	Réponses																
#	Corps	Réponses																			
<table border="1"><thead><tr><th>#</th><th>Corps</th><th>Réponses</th></tr></thead><tbody><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></tbody></table>			#	Corps	Réponses																
#	Corps	Réponses																			