**Bigscreen**

*Cahier des charges*

Nom du projet: Bigscreen

Date : 12/09/2022 au 10/10/2022

Noms: FUSTIER Julien et MARCEAU Cyril

*Analyse Client*

Bigscreen est une entreprise du numérique spécialisée dans la **réalité virtuelle**. Elle met à disposition une plateforme virtuelle de partage et d’échange entre amis (visionnage de films et d’émissions TV, jeux vidéos, organisation de conférences etc…)

La réalité virtuelle est un **secteur moderne**, qui évolue rapidement et en forte expansion. La principale problématique de notre client est de pouvoir suivre ces changements rapides en s’adaptant en permanence aux **attentes du marché**.

Pour répondre à ces besoins notre entreprise a réalisé une **application de sondage** leur permettant de mieux connaître les retours de leurs clients.

*Choix technologiques*

**Back-end**

* Concernant le back-end nous avons développé une **API rest** avec **Laravel** avec l’ORM eloquent.
* Pour le système d'authentification nous avons utilisé **laravel sanctum**. Un système natif à Laravel qui est compatible avec les **Single page application** (SPA).

**Front-end**

* Pour le front nous avons fait le choix d’utiliser [**Next.js**](https://nextjs.org/). Ce framework front basé sur React propose un ensemble d’outils permettant notamment l’intégration facilité des différentes routes de l’application, l’utilisation des middlewares ainsi que des gains de performance et de sécurité.
* Pour les appels serveur côté front nous avons utilisé la librairie [**Axios**](https://axios-http.com/fr/docs/intro), un client HTTP basé sur les promesses. C’est une librairie simple à utiliser et qui permet de simplifier grandement le système des appels serveur.
* Pour l’intégration visuelle nous avons utilisé la librairie graphique [**Ant Design**](https://ant.design/) créé pour react avec une bonne qualité d'expérience utilisateur et un design sobre et efficace.
* Pour l’affichage des graphiques côté front nous avons utilisé la librairie [**Chart.js**](https://www.chartjs.org/), celle-ci est simple d’utilisation et se base sur un design clair et efficace

*Github*

<https://github.com/cyrilmarceau/Bigscreen>

*Déploiement*

**Changer les variables d’environnements (api/.env):**

* (DB\_DATABASE, DB\_USERNAME et DB\_PASSWORD)

**Installation des dépendances:**

* $ cd api && composer install
* $ cd admin && npm i

**Lancer les migrations:**

* $ php artisan migrate:fresh –seed

**Lancer le serveur de développement**

* $ cd api / php artisan serve
* $ cd front / npm run dev

*Identifiant de connexion*

* Email: [admin@admin.fr](mailto:admin@admin.fr)
* Mot de passe: password

*Méthode de travail*

Nous utiliserons la **méthodologie Agile** car elle permet de s’adapter à la charge de travail en délivrant un travail fonctionnel en continue et permet de mieux diviser le travail étape par étape.

Pour gérer le versionning nous utilisons la méthodologie **Git Flow**. Nous partons d’une branche main qui contient le code finalisé. A partir de celle-ci nous créons une **branche** par tâche de travail. Lorsque nous avons terminé la fonctionnalité du code nous faisons un merge dans la branche main. Ceci nous permet de travailler en équipe de façon efficace et structurée.

Nous utiliserons la **convention de commit** (détail [ici](https://www.conventionalcommits.org/en/v1.0.0/)) qui permet d’avoir une vue globale et simple des différents commits réalisés durant ce projet.

*Outils utilisés*

Pour l’écriture du code nous avons utilisé [**Visual Studio Code**](https://code.visualstudio.com/), il s’agit d’un des éditeurs de texte les plus utilisés pour la programmation, spécialement adapté pour l’écriture de code et très riche grâce à son vaste système d'extension.

Afin de tester les différents appels API nous avons utilisé l’outil [**Insomnia**](https://insomnia.rest/) afin de pouvoir vérifier que nos requêtes fonctionnent correctement et renvoient les bonnes données.

Pour gérer le versioning du code nous avons utilisé le logiciel [**Git**](https://git-scm.com/), très complet et puissant grâce à son vaste ensemble de commandes.

Pour gérer le partage du code à distance dans le cadre du travail en équipe nous utilisons la plateforme [**Github**](https://github.com/cyrilmarceau/Bigscreen).

Pour la planification et le partage des tâches nous avons utilisé l’outil [**Jira Software**](https://www.atlassian.com/fr/software/jira) spécialement adapté à la méthode agile notamment grâce à son tableau kanban:

[**https://cyrilmarceau.atlassian.net/jira/software/projects/BIG/boards/2**](https://cyrilmarceau.atlassian.net/jira/software/projects/BIG/boards/2)

Pour la mise en forme du schéma de la base de données nous utiliserons [**DB diagram**](https://dbdiagram.io/home). Un outil spécialement adapté pour les développeurs qui permet de relier le code avec la représentation visuelle des tables.

Pour l’édition des documents de travail (cahiers des charges et slides de présentation orale) nous avons utilisé les outils [**Google Docs**](https://www.google.fr/intl/fr/docs/about/) et [**Google Slides**](https://www.google.fr/intl/fr/slides/about/). Ils ont l’avantage d’être gratuits tout en étant complet et adapté au travail en équipe.

Pour le travail à distance nous avons utilisé [**Discord**](https://discord.com/), un logiciel de communication gratuit et très complet notamment par son système de partage d’écran sophistiqué.

*Evaluation du temps de travail*

| **Entrée** | **Sortie (jours)** |
| --- | --- |
| API - Migrations + Seeders + factories | 1 |
| API - Récupérer la liste des sondages + récupérer 1 sondage + Création d’un sondage | 1 |
| API - Récupérer la liste des questions + formulaire des questions | 0.5 |
| API - Authentification | 0.2 |
| API - Graphique | 0.3 |
| FRONT - Layout + Route protégé | 0.5 |
| FRONT - Authentification | 0.2 |
| FRONT - API Factory | 0.2 |
| FRONT - Afficher les questions + Afficher les réponses d’un sondé | 0.3 |

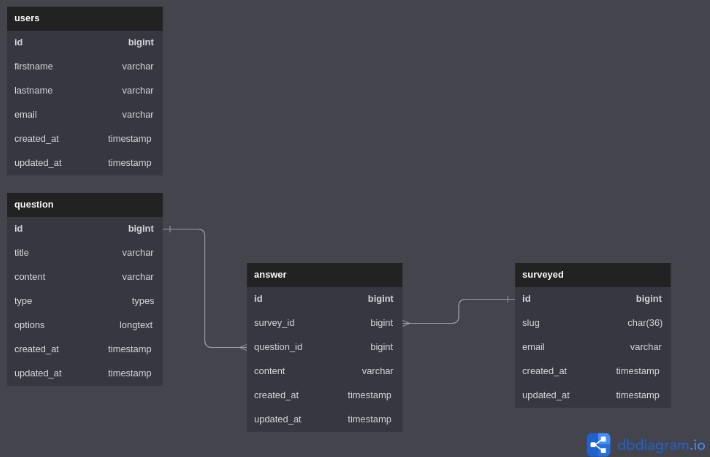
*Liste fonctionnelle*

| **FRONT** | **API** |
| --- | --- |
| En tant que client je peux répondre à un sondage | Migrations |
| En tant que client que je peux accéder à mon sondage | Seeders + factories |
| En tant qu’admin je peux me connecter | Récupérer la liste des sondages |
| En tant qu’admin je peux afficher les questions | Récupérer un sondage |
| En tant qu’admin je peux afficher les sondages | Récupérer la liste des questions |
| En tant qu’admin je peux afficher les réponses d’un sondé | Authentification |
| En tant qu’admin je peux afficher les graphiques | Création d’un sondage |
| En tant qu’admin je peux afficher la liste des sondages | Graphique |

*Recettage*

| **Fonctionnalité** | **Status** |
| --- | --- |
| API - Migrations | Fonctionnelle |
| API - Seeders + factories | Fonctionnelle |
| API - Récupérer la liste des sondages | Fonctionnelle |
| API - Récupérer 1 sondage | Fonctionnelle |
| API - Récupérer la liste des questions | Fonctionnelle |
| API - Authentification | Fonctionnelle |
| API - Création d’un sondage | Fonctionnelle |
| API - Graphique | Fonctionnelle |
| FRONT - Layout | Fonctionnelle |
| FRONT - Page authentification | Fonctionnelle |
| FRONT - API Factory | Fonctionnelle |
| FRONT - Formulaire des questions | Fonctionnelle |
| FRONT - Afficher les questions | Fonctionnelle |
| FRONT - Afficher les réponses d’un sondé | Fonctionnelle |
| FRONT - Affichage graphique | Fonctionnelle |
| FRONT - Message confirmation sondage | Fonctionnelle |
| FRONT - Affichage liste sondage | Fonctionnelle |
| FRONT - Route protégé | Fonctionnelle |

*Base de données*

****

**Table User:**

Informations concernant l’utilisateur admin

**Table Question:**

Contient les différentes questions définies au préalable

**Table Answer:**

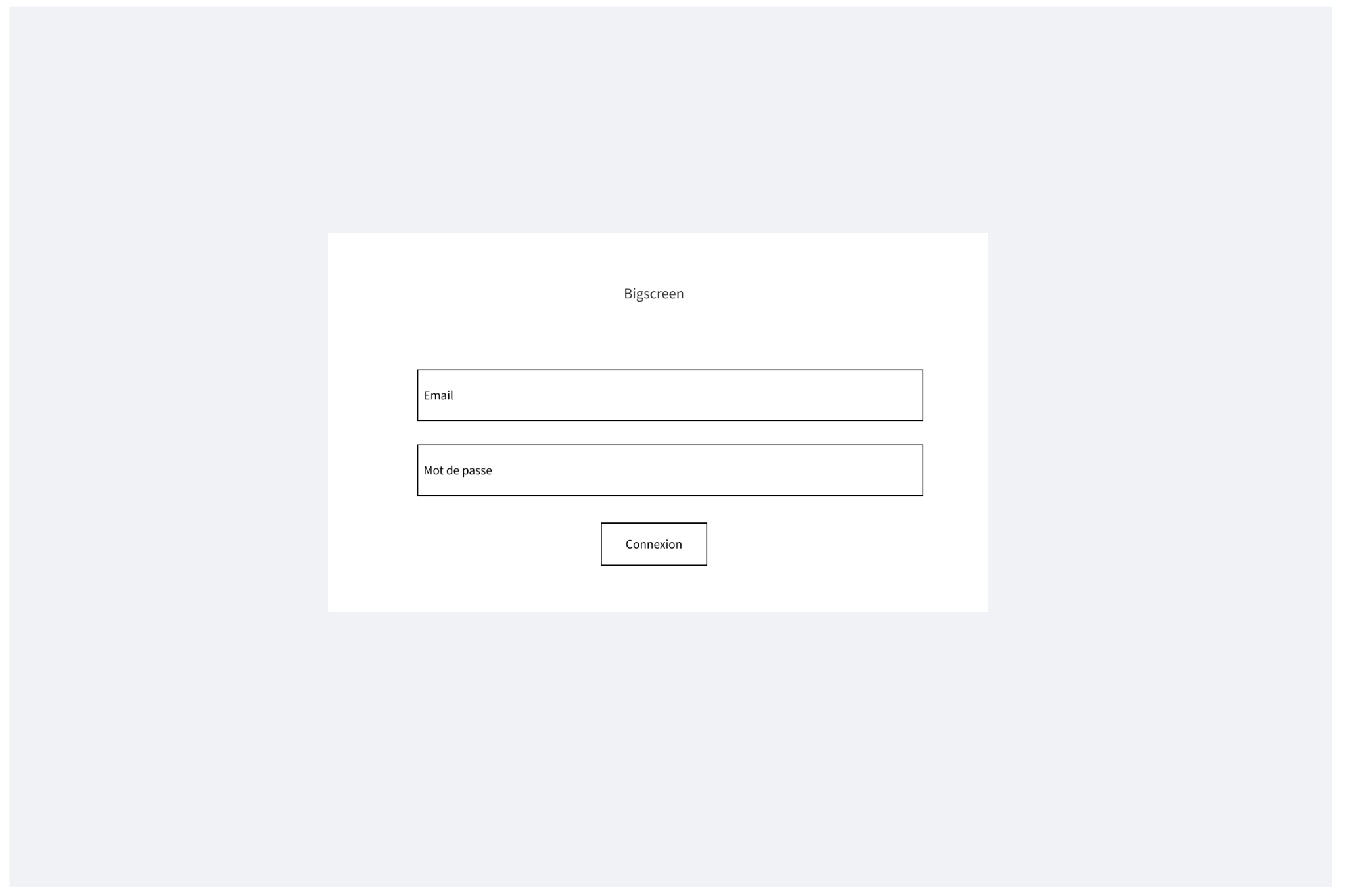
Contient les réponses et la relation vers la question

**Table Surveyed:**

Contient les différents sondé et la relations vers les réponses

*Wireframe - administration*

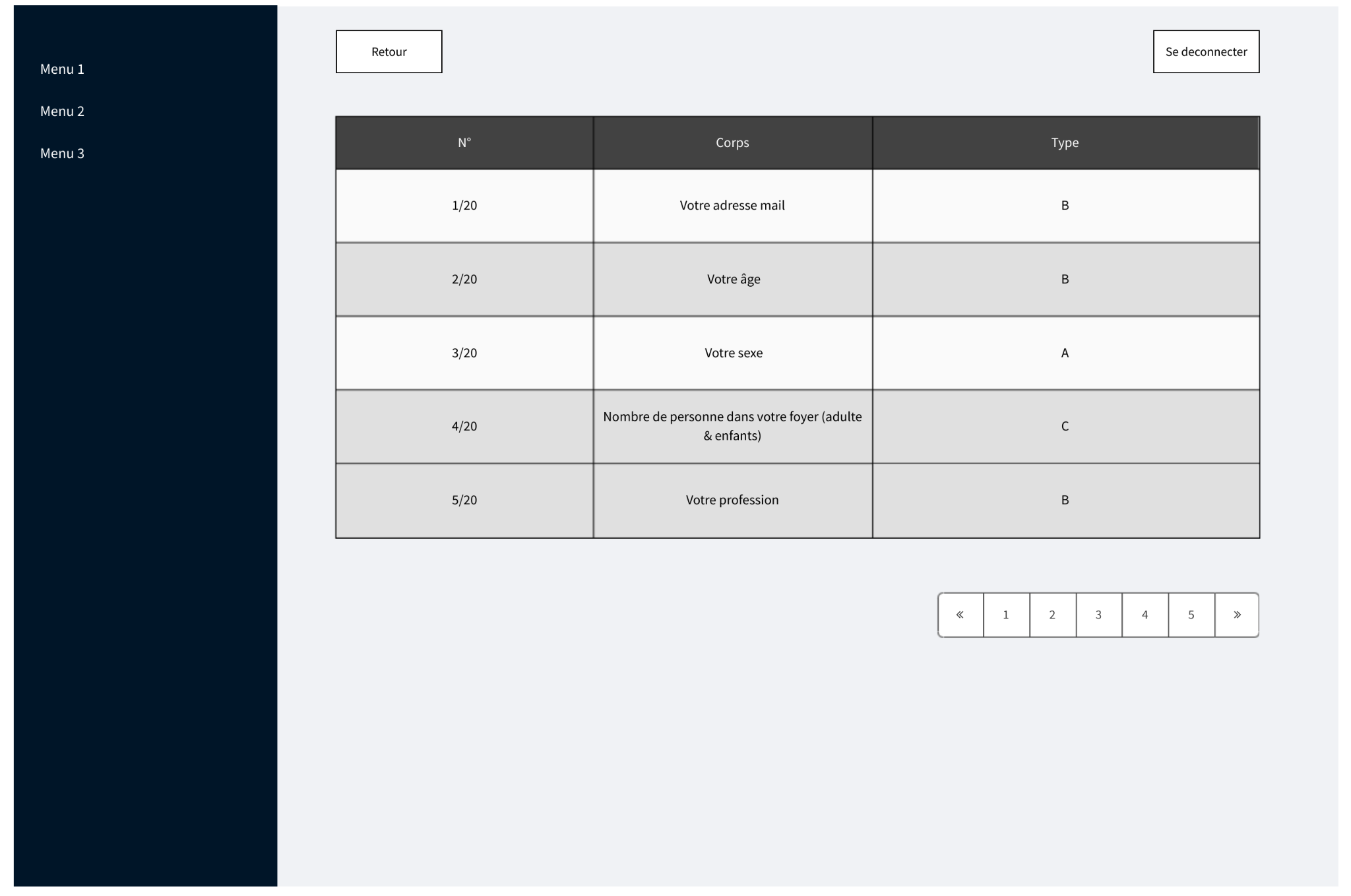
***Page d’authentification***

****

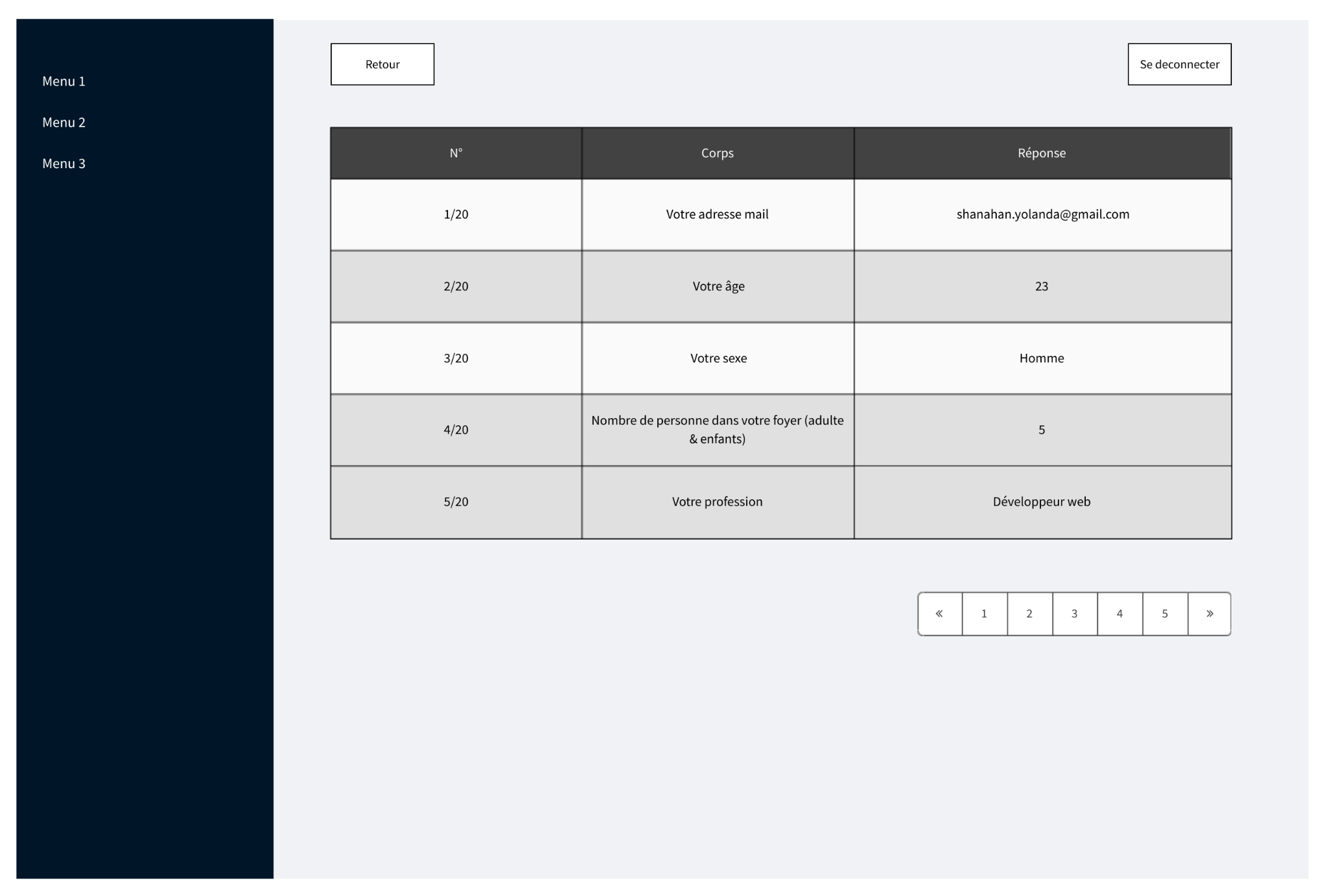
***Page d’accueil***

****

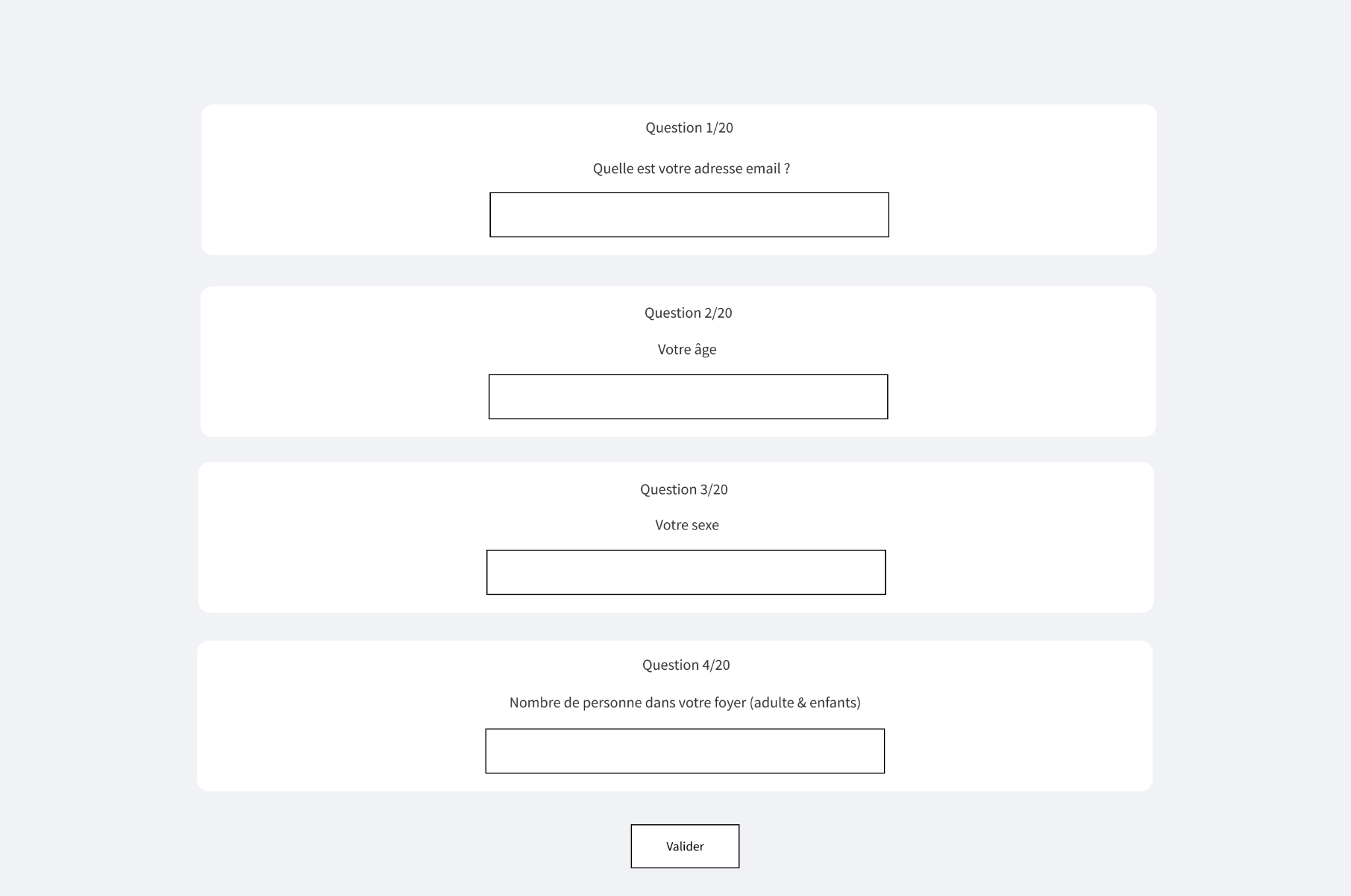
***Liste des questions***

**

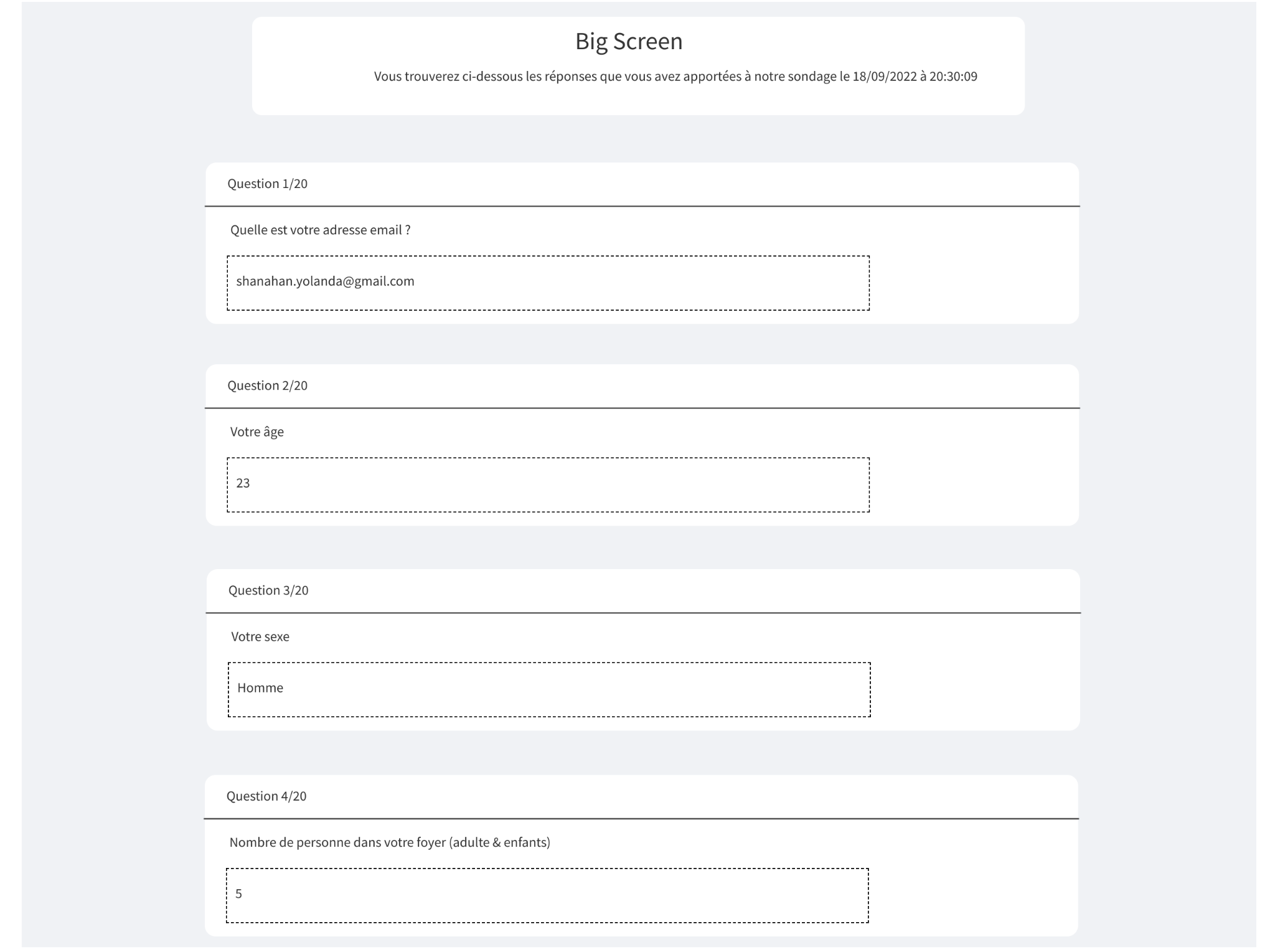
***Liste des réponses***

**

*Wireframe - client*

**

***Questionnaire***

***Réponses d’un sondé***

*Point de développement*

Durant ce projet nous avons eu comme principale difficulté la construction de l’**API pour les graphiques**. Il a fallu créer des fonctions assez génériques afin de générer les différentes réponses concernant les questions en gérant les différents cas possibles de chacune.

*Propositions d’amélioration / évolutions futures*

* Editer des sondages
* Supprimer des sondages
* Sauvegarder la progression de son sondage

*Création des fonctions côté back-end*

App/Models/Surveyed

getAllWithRelations(array $relationships)

$relationships: Tableau contenant les relations à récupérer

*Récupère tous les champs d’une table et les relations associées*

*App/Models/Question*

getAllWithExcludedColumns(array $columnToExclude)

$columnToExclude: Tableau contenant les colonnes à exclure

*Récupère tous les champs d’une table et les relations associées*

*App/Http/Controller/Controller*

sendResponse(mixed $result, string $message)

$result: Objet ou tableau d’objet contenant le résultat de la requête

$message: Le message à renvoyer

*Renvoie un JSON avec un status 200, un message et un objet “data” qui contiendra le contenu de la variable*

sendError(string $error, int $code = 404)

$error: Message d’erreur

$code: Code erreur de la requête

*Renvoie un JSON avec un message d’erreur et le code erreur de la requête*

*App/Http/Controller/Admin/AdminChartController*

removeDupplicateValue(Collection $answer)

$answers: Tableau contenant la collection

*Supprime les doublons et renvoie un nouveau tableau*

getAnswerStats(array $answers, Question $question)

$answers: Tableau de réponse après avoir supprimer les doublons

$question: Question en relation avec la réponse

*Renvoie un tableau contenant les statistiques de chaque réponse par rapport à une question*

*Création des composants côté front-end*

*Component/SurveyedForm*

SurveyedForm(questions, submitSurveyed, submitFailed)

questions: Liste des questions

submitSurveyed: Fonction lors de la validation

submitFailed: Fonction appelée si la validation du formulaire a échoué

*Component/CardQuestion*

CardQuestion(title, content, span, children)

title: Titre de la question

content: Corps de la question

span: Nombre de colonne en taille XS

children: Enfant appelé dans le composant parent

*Component/Layout/DefaultLayout*

DefaultLayout(children)

children: Enfant appelé dans le composant parent

Ce composant est utilisé pour le layout de la partie client

*Component/Layout/AdminLayout*

AdminLayout(children)

children: Enfant appelé dans le composant parent

Ce composant est utilisé pour le layout de la partie admin

*Helper/index*

Helper()

Cette classe contient plusieurs méthodes concernant le localStorage, format de date, parser du JSON

*api/index*

APIFactory(publicApiUrl, prefix)

publicApiUrl: URL de l’api

prefix: Prefix de l’api

Cette classe contient plusieurs méthodes concernant les appels API

getAxiosInstance(authenticatedRoute = false, extraParams = {})

authenticatedRoute: Permet de savoir si le header d’authentification doit être ajouté

extraParams: header supplémentaire à rajouter

Création d’une instance d’axios

handleResponse(status, data, message)

status: Code de retour de l’appel API

data: Données provenant de l’API

message: Message d’erreur

Cette méthode formate les résultats des appels API

*Documentation - API - ADMIN*

Se connecter

***url***: <http://127.0.0.1:8000/api/login>

***type:*** POST

***Paramètres***:

{

"email": string,

"password": string,

}

200

{

"success": true,

"data": {

"token": "17|X4vufTR9YdlFgvY8jyqy3MrWLJ5M1laIkNZH4EyU",

"name": "Oliver Greenholt"

},

"message": "Bienvenue Oliver Greenholt"

}

404

{

"success": false,

"message": "Identifiant incorrect",

"data": {

"error": "unauthorised"

}

}

Se déconnecter

***url***: [http://127.0.0.1:8000/api/logout](http://127.0.0.1:8000/api/login)

***type:*** POST

***header***:

* Authorization: Bearer token
* Accept: application/json

200

{

"success": true,

"data": {

"data": null

},

"message": "Utilisateur déconnecté"

}

401

{

"message": "Unauthenticated."

}

Récupérer la liste des sondés

***url***: <http://127.0.0.1:8000/api/admin/surveyeds>

***type:*** GET

***header***:

* Authorization: Bearer token
* Accept: application/json

200

{

"success": true,

"data": [

{

"id": 1,

"slug": "444aa4e1-78a8-38c3-8535-43fcba8a3694",

"email": "kimberly59@gmail.com",

"created\_at": "2022-09-18T18:30:09.000000Z",

"updated\_at": "2022-09-18T18:30:09.000000Z",

"answers": [

{

"id": 1,

"content": "shanahan.yolanda@gmail.com",

"created\_at": "2022-09-18T18:30:09.000000Z",

"updated\_at": "2022-09-18T18:30:09.000000Z",

"question": {

"id": 1,

"title": "1/20",

"content": "Votre adresse mail",

"type": "B",

"options": null,

"created\_at": "2022-09-18T18:30:09.000000Z",

"updated\_at": "2022-09-18T18:30:09.000000Z"

}

},

],

"message": "Liste des sondés récupérés avec succès"

}

401

{

"message": "Unauthenticated."

}

Récupérer la liste des questions

***url***: <http://127.0.0.1:8000/api/admin/questions>

***header***:

* Authorization: Bearer token
* Accept: application/json

***type:*** GET

200

**{**

**"success": true,**

**"data": [**

**{**

**"id": 1,**

**"title": "1/20",**

**"content": "Votre adresse mail",**

**"type": "B",**

**"created\_at": "2022-09-18T18:30:09.000000Z",**

**"updated\_at": "2022-09-18T18:30:09.000000Z"**

**},**

**{**

**"id": 2,**

**"title": "2/20",**

**"content": "Votre âge",**

**"type": "B",**

**"created\_at": "2022-09-18T18:30:09.000000Z",**

**"updated\_at": "2022-09-18T18:30:09.000000Z"**

**}**

**],**

**"message": "Liste des questions récupérer avec succès"**

**}**

401

{

"message": "Unauthenticated."

}

Récupérer les statistiques pour le graphique

***url***: <http://127.0.0.1:8000/api/admin/charts>

***header***:

* Authorization: Bearer token
* Accept: application/json

***type:*** GET

200

**{**

**"success": true,**

**"data": [**

**{**

**"content": "Quel marque de casque VR utilisez vous ?",**

**"stats": {**

**"label": "Windows Mixed Reality",**

**"count": 1**

**},**

**"type": "pie"**

**},**

**{**

**"content": "Sur quel magasin d’application achetez vous des contenus VR ?",**

**"stats": {**

**"label": "Google Play",**

**"count": 1**

**},**

**"type": "pie"**

**},**

**{**

**"stats": [**

**{**

**"label": "Audio",**

**"count": 3.6**

**}**

**],**

**"type": "radar"**

**}**

**],**

**"message": null**

**}**

401

{

"message": "Unauthenticated."

}

Documentation - API - CLIENT

Récupérer la liste des questions

***url***: [http://127.0.0.1:8000/api/client/questions](http://127.0.0.1:8000/api/admin/charts)

***header***:

* Accept: application/json

***type:*** GET

200

**{**

**"success": true,**

**"data": [**

**{**

**"id": 1,**

**"title": "1/20",**

**"content": "Votre adresse mail",**

**"type": "B",**

**"created\_at": "2022-09-18T18:30:09.000000Z",**

**"updated\_at": "2022-09-18T18:30:09.000000Z"**

**},**

**{**

**"id": 2,**

**"title": "2/20",**

**"content": "Votre âge",**

**"type": "B",**

**"created\_at": "2022-09-18T18:30:09.000000Z",**

**"updated\_at": "2022-09-18T18:30:09.000000Z"**

**}**

**],**

**"message": "Liste des questions récupérer avec succès"**

**}**

Récupérer un sondage

***url***: [http://127.0.0.1:8000/api/client/surveyeds/](http://127.0.0.1:8000/api/surveyeds/c9de43b5-09e5-3391-a320-7dfc46d4608c)<uuid>

***header***:

* Accept: application/json

***type:*** GET

200

**{**

**"success": true,**

**"data": {**

**"id": 9,**

**"slug": "c9de43b5-09e5-3391-a320-7dfc46d4608c",**

**"email": "jruegckefr@hotmail.com",**

**"created\_at": "2022-09-16T09:36:25.000000Z",**

**"updated\_at": "2022-09-16T09:36:25.000000Z",**

**"answers": [**

**{**

**"id": 140,**

**"content": "32",**

**"created\_at": "2022-09-16T09:36:25.000000Z",**

**"updated\_at": "2022-09-16T09:36:25.000000Z",**

**"question": {**

**"id": 2,**

**"title": "2/20",**

**"content": "Votre âge",**

**"type": "B",**

**"options": null,**

**"created\_at": "2022-09-15T15:29:25.000000Z",**

**"updated\_at": "2022-09-15T15:29:25.000000Z"**

**}**

**}**

**]**

**},**

**"message": "Sondé récupéré avec succès"**

**}**

404

{

"success": false,

"message": "Aucun sondé n'a été trouvé"

}

Créer un sondage

***url***: <http://127.0.0.1:8000/api/client/surveyeds>

***header***:

* Content-Type: application/json

***type***: POST

***Paramètre***:

{

"email": string,

"questions": [{

"questionId": int<1-20>,

"content": string | int

}]

}

200

**{**

**"success": true,**

**"data": {**

**"slug": "82ab0e7d-157d-380e-b417-a37e965778f7",**

**"email": "test.fr@gmail.com",**

**"updated\_at": "2022-09-25T16:10:32.000000Z",**

**"created\_at": "2022-09-25T16:10:32.000000Z",**

**"id": 6**

**},**

**"message": "Sondé crée avec succès"**

**}**

400

{

"success": false,

"message": "Erreur de validation du formulaire",

"data": {

"email": [

"L'email est obligatoire"

]

}

}

400

{

"success": false,

"message": "Erreur de validation du formulaire",

"data": {

"email": [

"Vous devez correctement formater votre email"

]

}

}

422

**{**

**"success": false,**

**"message": "**Erreur enregistrement: Vous avez déjà répondu au sondage avec une adresse email identique**"**

**}**