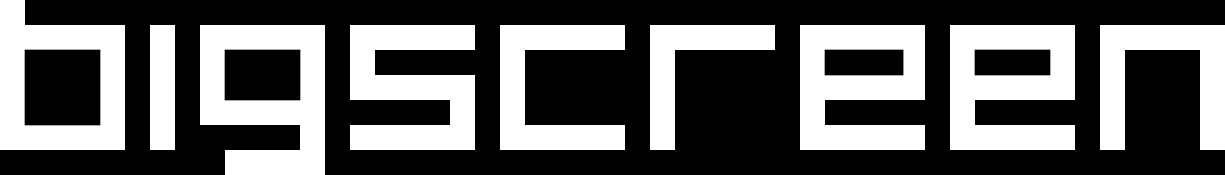
****

**Présentation**

Projet “Big\_screen”, 12/09/2022, GAVREL Tomas et REGUIGUI Yassine.

**Analyse client**

Big Screen souhaiterait mettre en place une application servant dans le cadre de **réalisation de sondages** auprès de ses utilisateurs.

L’application doit être en mesure de lister les questions préalablement choisies pour l’usage utilisateur et pouvoir ensuite être **consultable via un lien spécifique** qui sera fourni à la soumission du formulaire de sondage.  
  
L’administrateur de l’application doit avoir un **espace privé** dédié auquel il a accès via un formulaire de login qui lui permettra de regarder en temps réel les statistiques des différents sondages réalisés.  
  
L’application se compose donc en **deux parties principales**, un interface pour les utilisateurs ainsi qu’un espace admin privé.  
  
Côté utilisateur y a donc une page principale servant à afficher le formulaire de sondage pour les utilisateurs, ainsi qu’une page de résultat consultable par l’utilisateur après la réalisation du sondage et **identifiée par un token**.  
  
Côté administrateur, il y a une page d’identification qui doit servir à accéder aux trois pages principales de l’interface admin qui ne doivent pas être accessibles sans **identification au préalable**.

**Choix technologiques**

Au niveau des choix technologiques, l’application côté **back-end** sera basée sur un **API REST** utilisant le **framework Laravel**.  
Laravel propose une solution de conception d’api pouvant être réutilisée via des appels de requêtes côté back dans le cadre d’une application ou d’un interface séparé pour le front.  
Le stockage des données se fera sur **MySQL** à partir de tables et communiquera avec l’ORM de Laravel **Eloquent**.

Avec cela l’application sera couplée à un interface **front-end** séparé. Le choix technologique de cette interface sera basé sur le **framework javascript Vue.js**.

Vue.js intègre des outils efficaces afin de réaliser une **single-page application** qui améliore la fluidité de l’application et l’expérience utilisateur.  
Des appels vers l’API pourront donc être réalisés au travers de cette interface dans le cadre de la réalisation du projet.

**Adresse Github**

<https://github.com/tomas-gl/Big_screen>

**Déploiement**

Afin de déployer le projet en local; il faudra en premier lieu naviguer vers le répertoire “api” à partir de la racine du projet et correspondant à l’application Laravel, puis entrer les commandes suivantes en ligne de commande:

* ***composer install***

→ Installe les dépendances nécessaires au fonctionnement de l’application.

Pour les commandes suivantes il faudra d’abord s’assurer qu’une nouvelle base de donnée en serveur local ait été créé et porte le même nom que celui défini dans le fichier .env à la racine du projet Laravel (ici “big\_screen”)

* ***php artisan migrate***

→ Permet la migration de tables nécessaires au fonctionnement de l’application dans la nouvelle base de données en local.

* ***php artisan migrate:fresh –seed***

→ Rempli les tables de données et/ou fausses données qui seront utilisables dès le lancement de l’application.

* ***php artisan serve***

→ Démarre le projet Laravel qui servira de serveur à l’application.

Dans un second temps il faudra naviguer vers le répertoire “admin”, toujours à partir de la racine du projet et qui correspond lui à l’application Vue.js, puis entrer les commandes suivantes en ligne de commande:

* ***npm instal****l*

→ Installe les dépendances nécessaires à l’application

* ***npm run serve***

→ Lance le projet dans une adresse en local consultable dans un navigateur

**Méthode de travail**

La méthode de travail privilégiée dans le cadre du projet sera le **modèle en cascade** (ou plus communément appelé “**waterfall**”).

L’idée dans le choix de cette méthode sera de **définir l’ordre des tâches** à réaliser le plus approprié dans le cadre de l’avancement du projet afin de **travailler par étapes**.

Une fonctionnalité terminée ou une modification de styles ou d’interface doit justifier d’un **commit**.

**Outils utilisés**

L’outils de développement principal privilégié est celui de l’éditeur de code **Visual Studio Code** qui propose un interface très rempli que ce soit pour la conception ou le débogage des applications.

Pour la mise en place d’un **serveur Web** en local, le choix se porte vers **XAMPP**, une distribution Apache contenant un service MySQL et efficace dans le cadre du développement.

Afin de mettre en commun l’avancement du travail des membres du groupe, le choix d’une plateforme de visionnage et de contrôle de version du projet se tourne vers **Github**, une plateforme de collaboration et de partage en ligne permettant un travail en commun à distance.

**Evaluation du temps de travail**

| **Poste de développement** | **Temps/j** |
| --- | --- |
| Schématisation de la base de donnée | 1,5 |
| Création de l’application Laravel, de la table MySQL et des fichiers de migrations/seeders | 1 |
| Création de l’interface front avec Vue.js | 0,5 |
| Création de la page de sondage et établissement de la connexion entre l’API et le front pour la récupération des questions | 1 |
| Validation des champs du formulaire pour l’écriture en base | 1,5 |
| Création de la page de réponses au sondage pour un utilisateur identifiée par un token | 1,5 |
| Mise en place du formulaire de login et de la protection des routes vers l’espace admin | 1 |
| Création de la navigation côté administration | 0,5 |
| Ajout des tableaux de données côté administration | 1 |
| Ajout des chartes graphique côté administration | 1 |
| Optimisation et ajout de commentaires dans le code | 0,5 |

**Liste fonctionnelle**

Fonctionnalité côté public/utilisateur:

● Page de formulaire de sondage

● Page de consultation des résultats de sondage

Fonctionnalité côté administrateur:

● Formulaire de login

● Page d’administration avec chartes graphiques

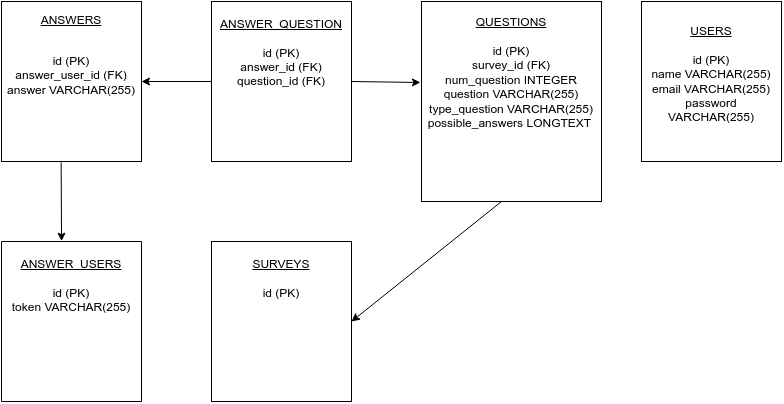
● Page d’administration avec tableaux des sondages réalisés

● Page d’administration avec tableau des questions du sondage

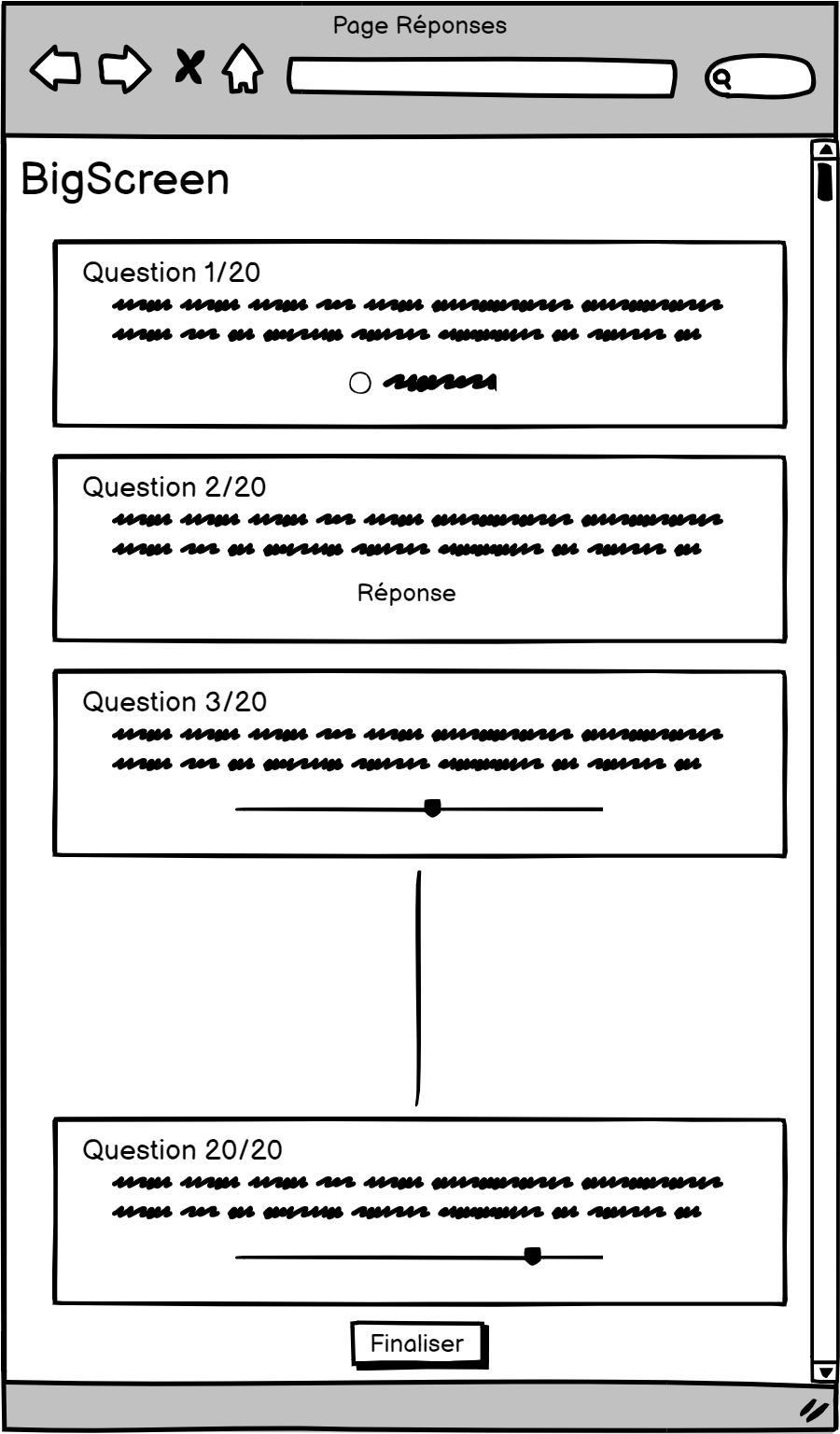
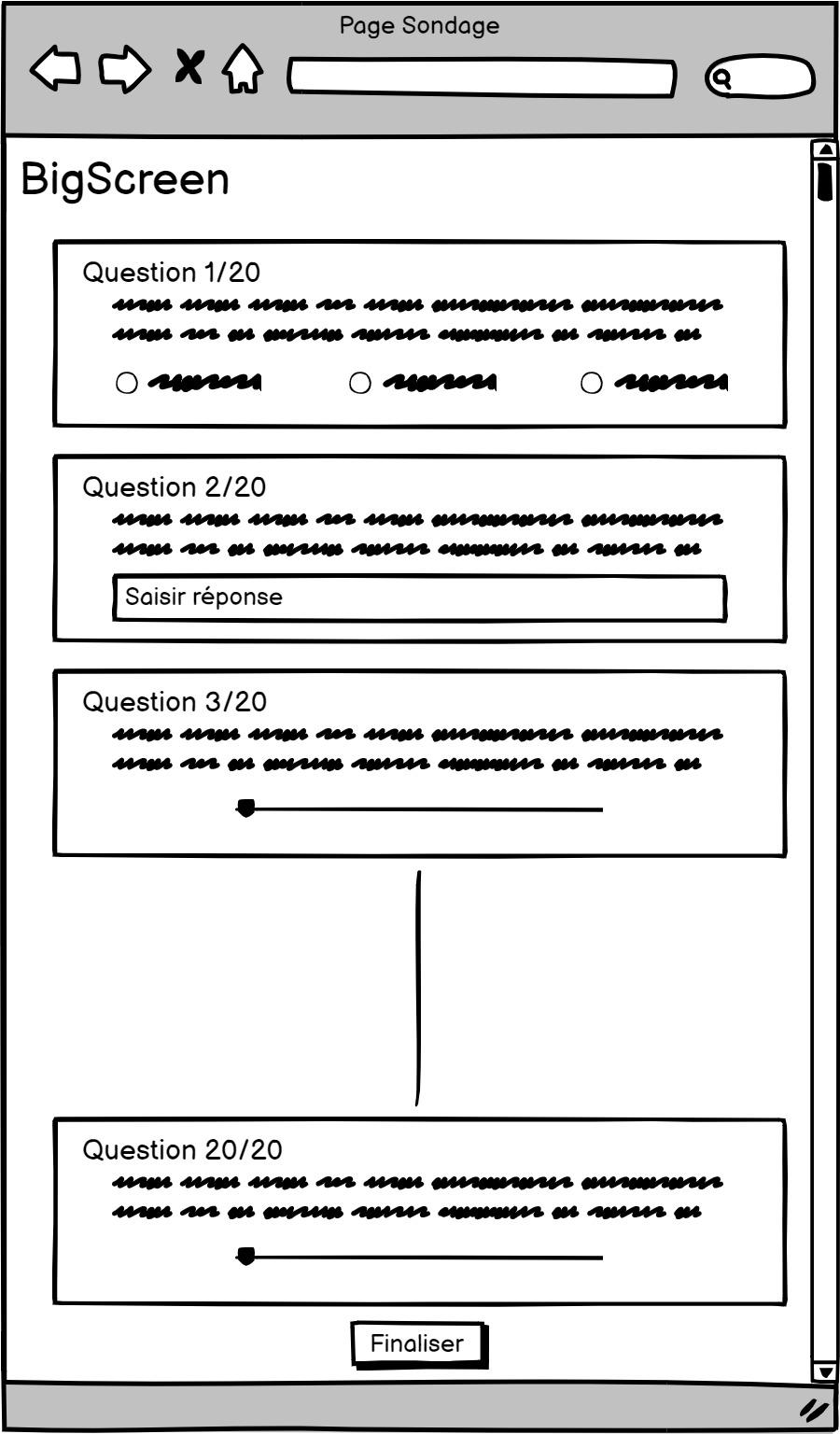
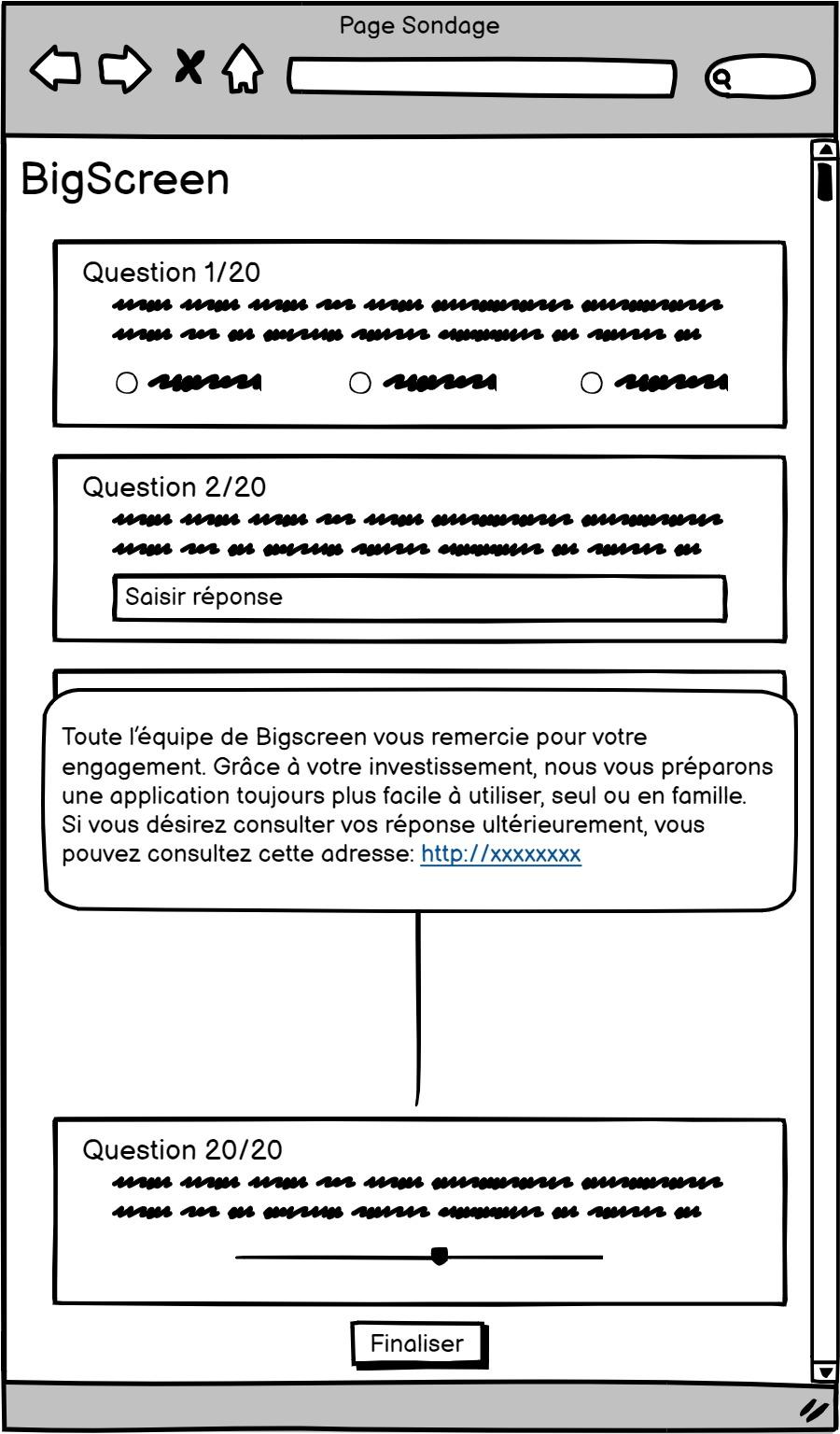
**Recettage**

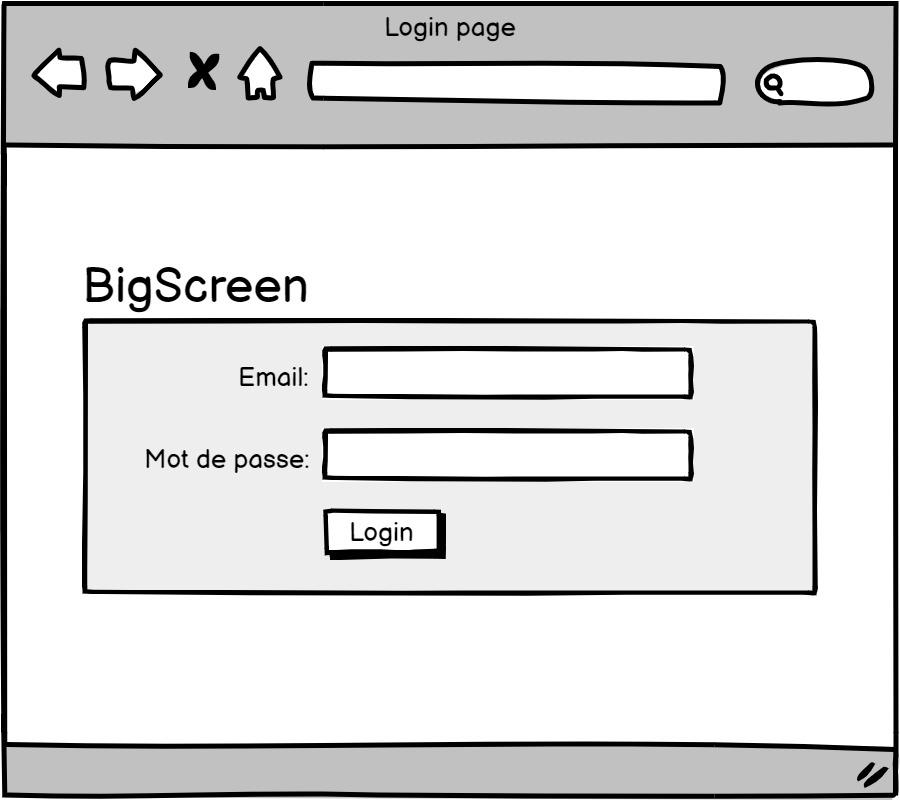
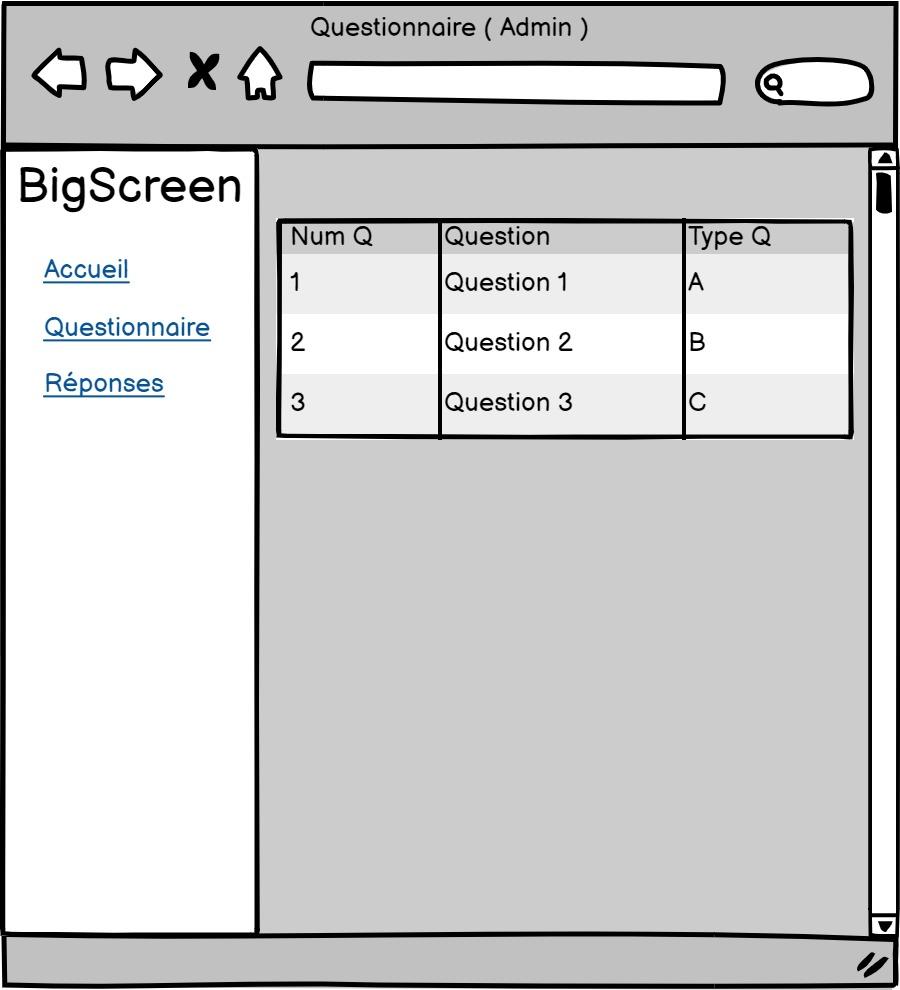
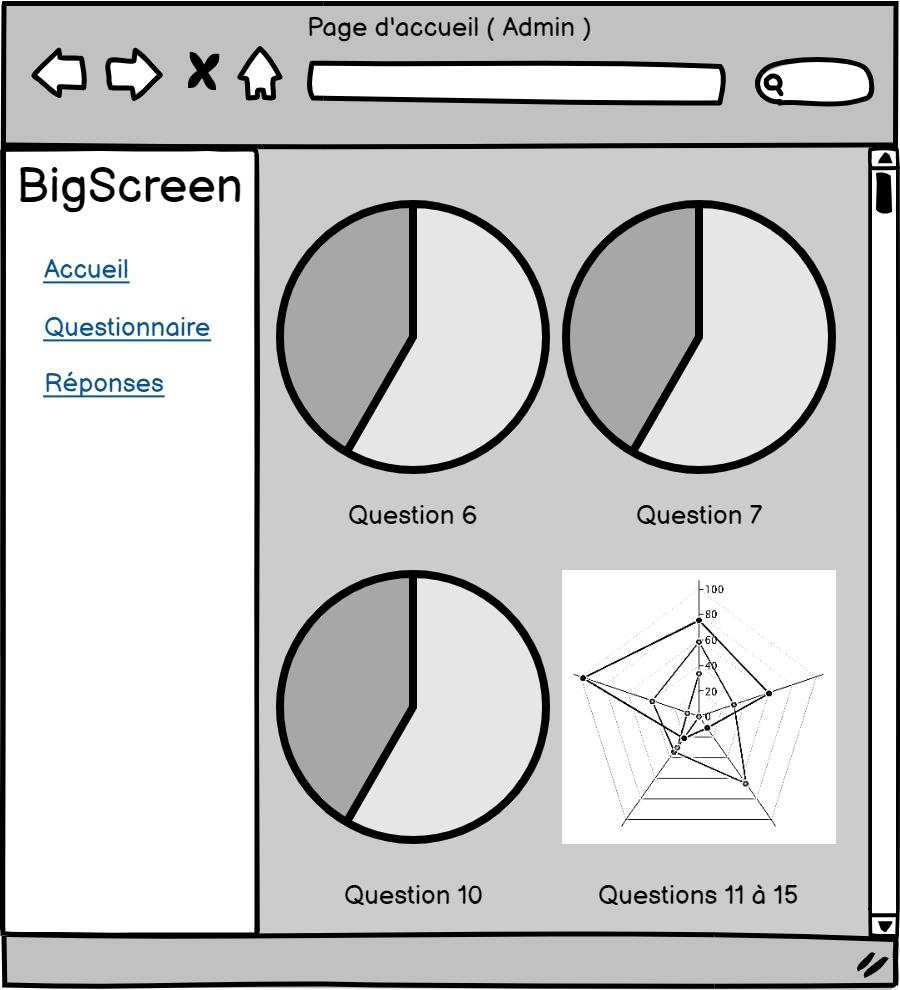
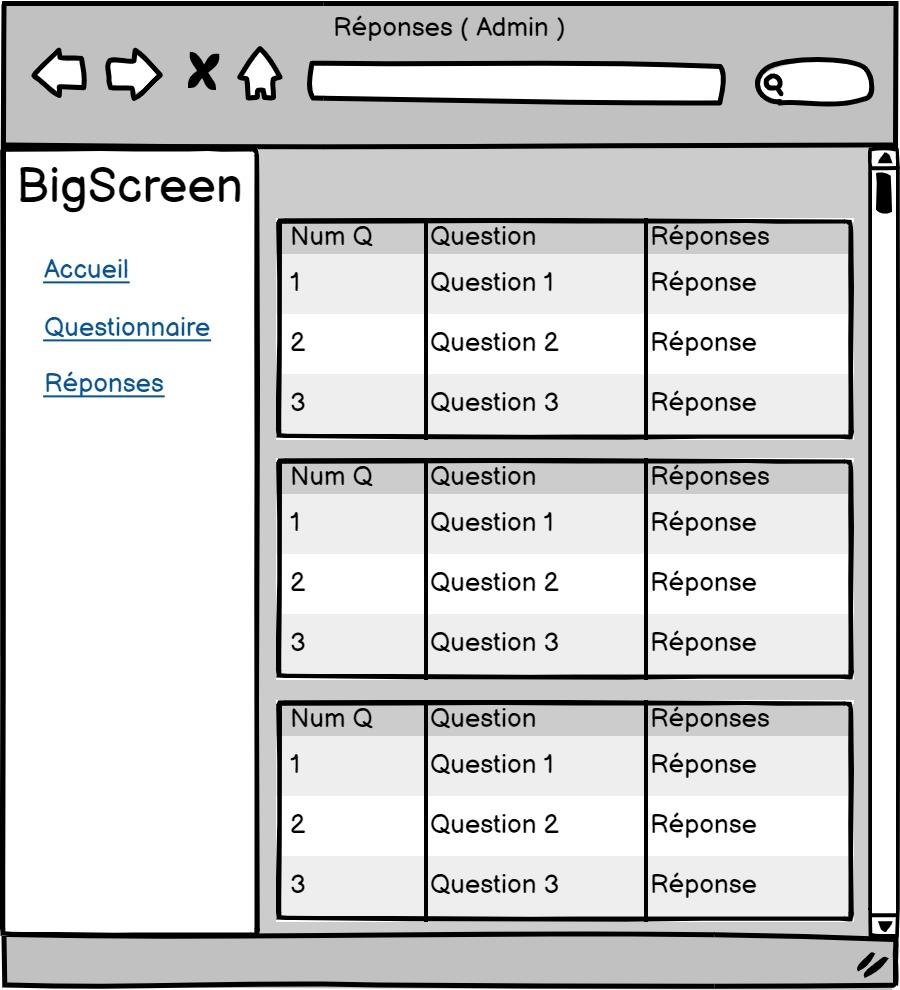
| **Fonctionnalité** | **État** |
| --- | --- |
| Page de formulaire de sondage | **Opérationnelle** |
| Page de consultation des résultats de sondage | **Opérationnelle** |
| Formulaire de login | **Opérationnelle** |
| Page d’administration avec chartes graphiques | **Opérationnelle** |
| Page d’administration avec tableaux des sondages réalisés | **Opérationnelle** |
| Page d’administration avec tableau des questions du sondage | **Opérationnelle** |

**Diagramme de la base de données**

****

**Wireframes**

**  **

**  **

**Documentation du code**

**Front-End Vue.js:**

**Pages:**

**Dossier admin:**

* **AnswersAdmin**

Affiche les résultats sous forme de tableau des sondages réalisés

* **HomeAdmin**

Accueil affichant sous forme de chartes les statistiques des sondages

* **LoginAdmin**

Identification public pour l’accès à l’espace administrateur

* **SurveyAdmin**

Affiche les questions du sondage sous forme de tableau

**Dossier User:**

* **SurveyPage**

Affiche le formulaire de sondage avec la liste des questions et un bouton de validation

* **SurveyResultPage**

Affiche la liste des réponses pour un sondage

**Composants:**

* **HeaderUser**

Header principale de l’application côté utilisateur

* **SideNavBar**

Barre de navigation côté administration

* **SurveyCard**

Affichage d’une question dans le formulaire

**Dossier router:**

* **index.js**

Configuration des routes de l’application

**Dossier store:**

* **index.js**

Stockage centralisé des données de l’application

**Librairies:**

* [Axios](https://axios-http.com/fr/docs/intro), client http établissant la connexion entre le côté client et serveur
* [Vuex](https://vuex.vuejs.org/), librairie de gestionnaire d’état pour l’application
* [ElementPlus](https://element-plus.org/en-US/), librairie de composants et de styles graphiques pour l’application
* [Vue-Chartjs](https://vue-chartjs.org/), librairie de création de chartes graphiques grâce à des données

**Back-end Laravel:**

Controllers Laravel:

* ***Class AdminController***

Stock les méthodes nécessaires au fonctionnement de l’espace administrateur

* ***Class SurveyController***

Stock les méthodes nécessaires au fonctionnement des pages de sondages

Méthodes des controllers:

AdminController

* ***login()***
* Identifie un utilisateur
* prend en paramètre l’objet Request
* Retourne une réponse de validation
* ***getSurveyDatas()***
* Récupères les données des sondages
* Retourne sous format json les résultats

SurveyController

* ***getQuestionsSurvey()***
* Récupère la liste des questions du sondage
* Retourne sous format json les questions
* ***saveQuestionSurvey()***
* Sauvegarde en base un nouveau sondage créé
* Prend en paramètre l’objet Request
* Retourne une réponse de validation
* ***getSurveyResult()***
* Récupère les réponses pour un sondage
* Prend en paramètre le string token
* Retourne sous format json les données

Structure des tables:

**ANSWERS**

Stock les réponses des sondages réalisés

Liste des champs:

* **id** (Primary key)
* **answer\_user\_id** (Foreign key), cible une ligne vers la table ANSWER\_USERS
* **answer** (Varchar(255 caractères)), contient une réponse

**ANSWER\_QUESTION**

Table d’association permettant la récupération des données d’une question via une réponse et inversement

Liste des champs:

* **id** (Primary key)
* **answer\_id** (Foreign key), cible une ligne vers la table ANSWERS
* **question\_id** (Foreign key), cible une ligne vers la table QUESTIONS

**QUESTIONS**

Stock les questions pour un sondage

Liste des champs:

* **id** (Primary key)
* **survey\_id** (Foreign key), cible une ligne vers la table SURVEYS
* **num\_question** (Integer), numéro de la question
* **question** (Varchar(255 caractères)), contenu de la question
* **type\_question** (Varchar(255 caractères)), type de la question
* **possible\_answers** (Longtext), réponses possibles à une question

**ANSWER\_USERS**

Stock un token servant de ciblage à un sondage

Liste des champs:

* **id** (Primary key)
* **token** (Varchar(255 caractères)), token pour un sondage

**SURVEYS**

Stock un ou plusieurs sondages

Liste des champs:

* **id** (Primary key)

**USERS**

Stock les utilisateurs

Liste des champs:

* **id** (Primary key)
* **name** (Varchar(255 caractères)), nom de l’utilisateur
* **name** (Varchar(255 caractères)), mail de l’utilisateur
* **password** (Varchar(255 caractères)), mot de passe de l’utilisateur