**CAHIER DES CHARGES**

**Présentation**

* **Nom :** Sondage Bigscreen
* **Date :** 12 Avril
* **Nom et Prénoms de l’étudiant :** Houssou Elvis Jean-Marc André

**Analyse Client**

Le projet "Sondage Bigscreen" consiste en la création d'une plateforme de sondage en ligne appelée Bigscreen. Cette plateforme permettra aux utilisateurs de participer à des sondages sur différentes questions. Les utilisateurs pourront créer des sondages personnalisés, partager le lien avec leurs contacts et recueillir leurs réponses. Les résultats des sondages seront ensuite affichés de manière agrégée pour une analyse facile.

**Choix Technologiques**

1. **Coté FrontEnd**

* **VueJs :** Le Framework Javascript sera utilisé pour développer l’interface utilisateur interactive et réactive.
* **Vue Router :** La librairie Vue Router sera utilisée pour gérer les routes et la navigation dans l’application frontend.
* **Bootstrap :** La bibliothèque

Bootstrap sera utilisée pour faciliter la conception de l’interface utilisateur et assurer un aspect visuel et attrayant.

1. **Coté BackEnd**

* **FrameworK PHP : Laravel** sera utilisé pour développer le backend et fournir une API RESTful pour l’application frontend.
* **Base de données : MySQL** sera utilisé pour stocker les sondages, les réponses et les données utilisateur en raison de sa fiabilité et de sa prise en charges de requête complexes.

**Adresse GitHub**

L’adresse Github du projet sera :

[Elvis-Houssou/Bigscreen\_sondage (github.com)](https://github.com/Elvis-Houssou/Bigscreen_sondage)

**Déploiement**

Pour déployer le projet veuillez suivre les étapes suivantes :

1. **Technologie à vérifier :**

Assurez-vous que Node.js, PHP, MySQL et les dépendances du projet sont bien installées.

1. **Actions à effectuer dans le terminal**

* Cloner le projet à partir du référentiel github.
* Naviguer dans le répertoire racine du projet.
* Exécuter la commande « npm install » pour installer les dépendances frontend.
* Exécuter la commande « composer install » pour installer les dépendances backend.
* Configurer les informations de connexion à la base de données dans le fichier « .env » de l’api.
* Exécuter la commande « npm run build » pour compiler les fichiers frontend.
* Exécuter la commande « php artisan serve » pour démarrer le serveur backend.

1. **Autres actions nécessaires pour arriver au rendu final**

* Vérifier la configuration de la base de données et s’assurer que toutes les tables requises sont créées.
* Configurer les autorisations et les mesures de sécurités appropriées pour le serveur.

**Méthode de Travail**

Nous adopterons une approche Agile pour le développement du projet "Sondage Bigscreen". Les itérations courtes et incrémentales permettront de répondre rapidement aux besoins changeants du client et d'assurer une livraison plus rapide et régulière des fonctionnalités. Chaque commit sera associé à une tâche ou une fonctionnalité spécifique, ce qui facilitera le suivi des changements et des contributions de chaque membre de l'équipe.

**Outils utilisés**

* **Logiciel :** Visual Studio Code, WampServer, Navigateur Web.
* **Outils de développement :** Npm, Composer, Git, Node js, EsLint pour le contrôle qualité du code.
* **Service en ligne :** Github pour le contrôle de version et le partage du code et Dbdiagram pour le diagramme de la base de données.

**Evaluation du Temps de Travail**

Voici une estimation approximative du temps de travail pour chaque poste de développement :

| **Poste de développement** | **Temps de travail (en jours)** |
| --- | --- |
| Conception et wireframes | 5 |
| Frontend | 7 |
| Backend | 10 |
| Base de données | 5 |
| Tests et recettage | 3 |
| Documentation du code | 5 |
| Total | 30 |

**Liste Fonctionnelle**

1. **FrontEnd :**

* Afficher un sondage
* Répondre à un sondage
* Affichage des résultats des sondages.

1. **BackEnd :**

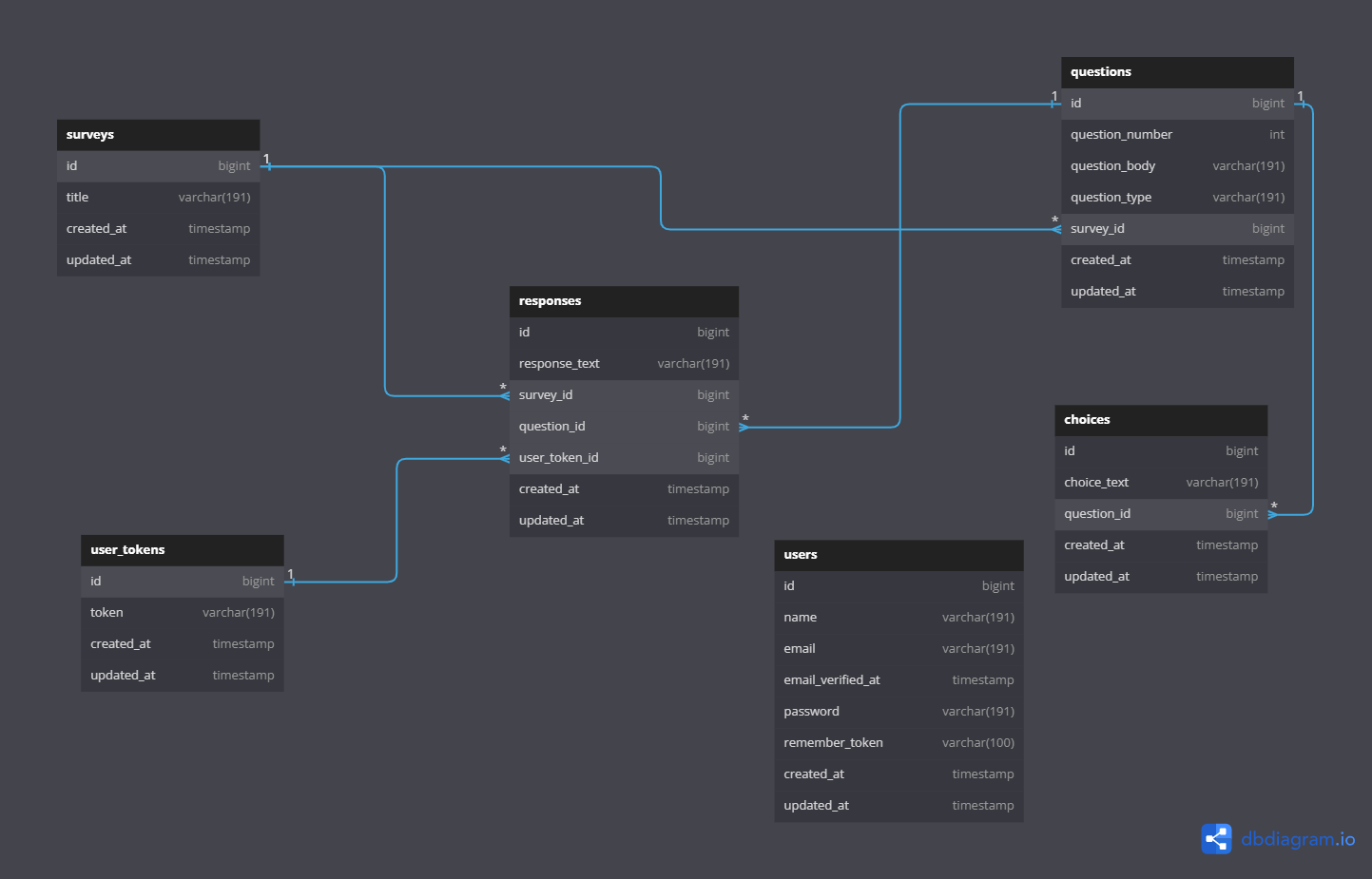
* Gestion des administrateurs (authentification, profil)
* Création via seeder des sondages.
* Gestions des réponses utilisateurs.
* Affichage des réponses de chaque utilisateur.

**Recettage**

Voici un tableau indiquant si chaque fonctionnalité est opérationnelle ou présentes des problèmes.

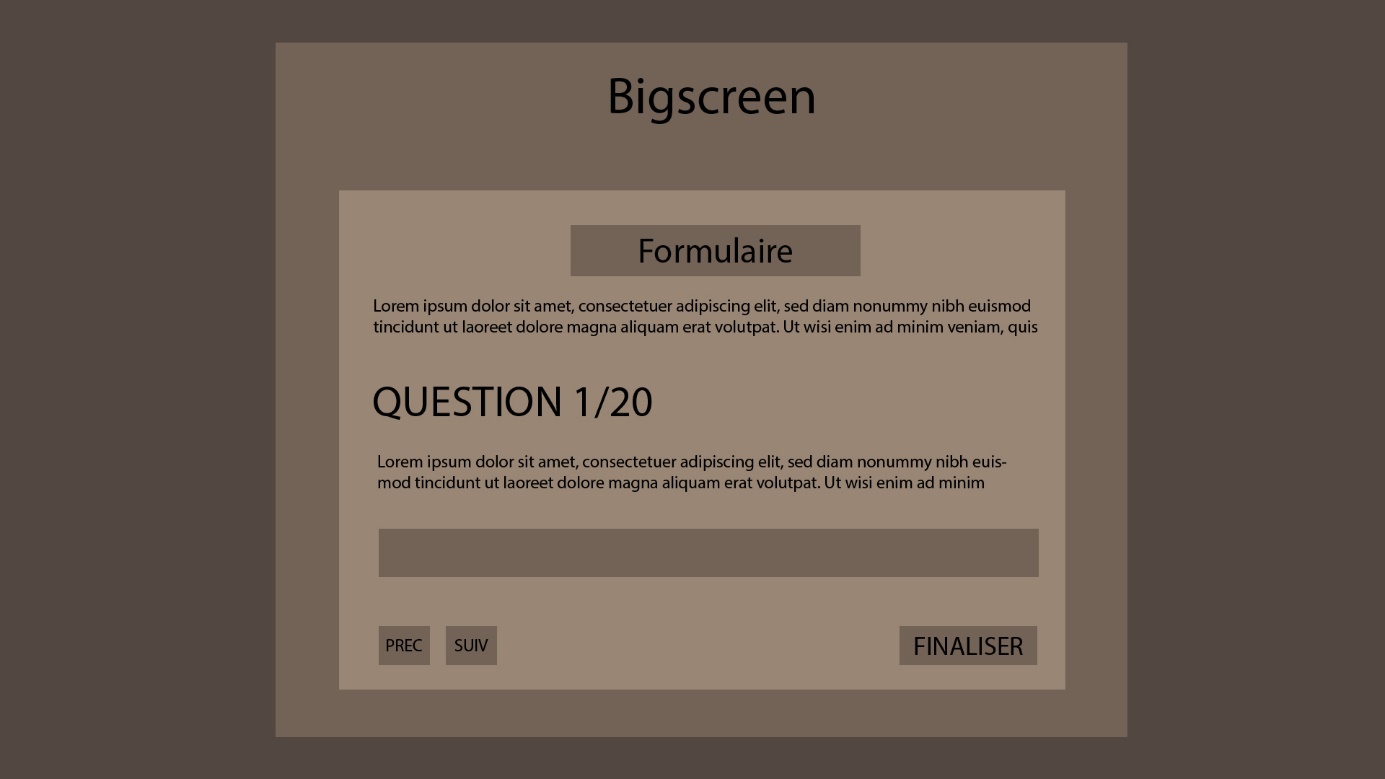
| **Fonctionnalité** | **État** |
| --- | --- |
| Connexion/déconnexion | Opérationnelle |
| Partage de liens de sondages | Opérationnelle |
| Répondre à un sondage | Opérationnelle |
| Affichage des résultats | Opérationnelle |

**Diagramme de la base de données**

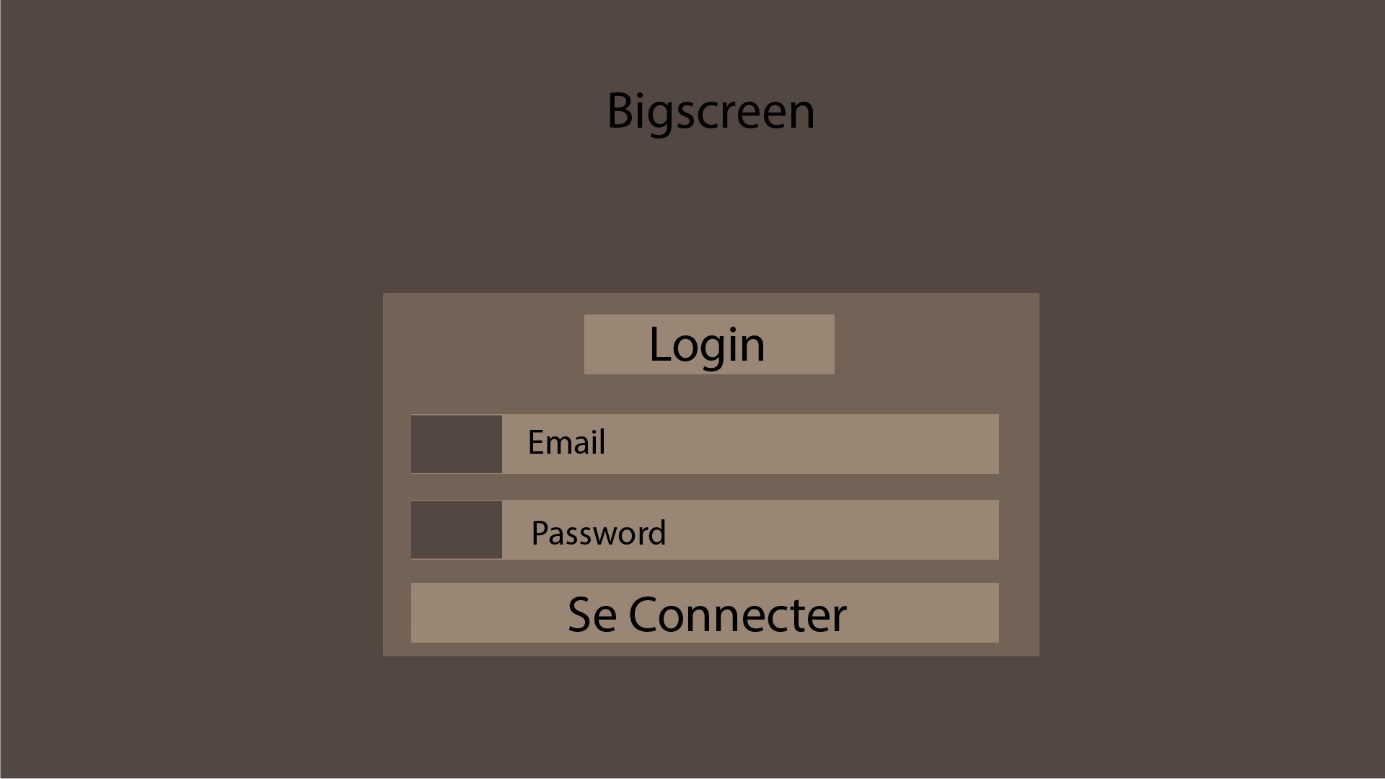


**Wireframes**

**Page sondage**

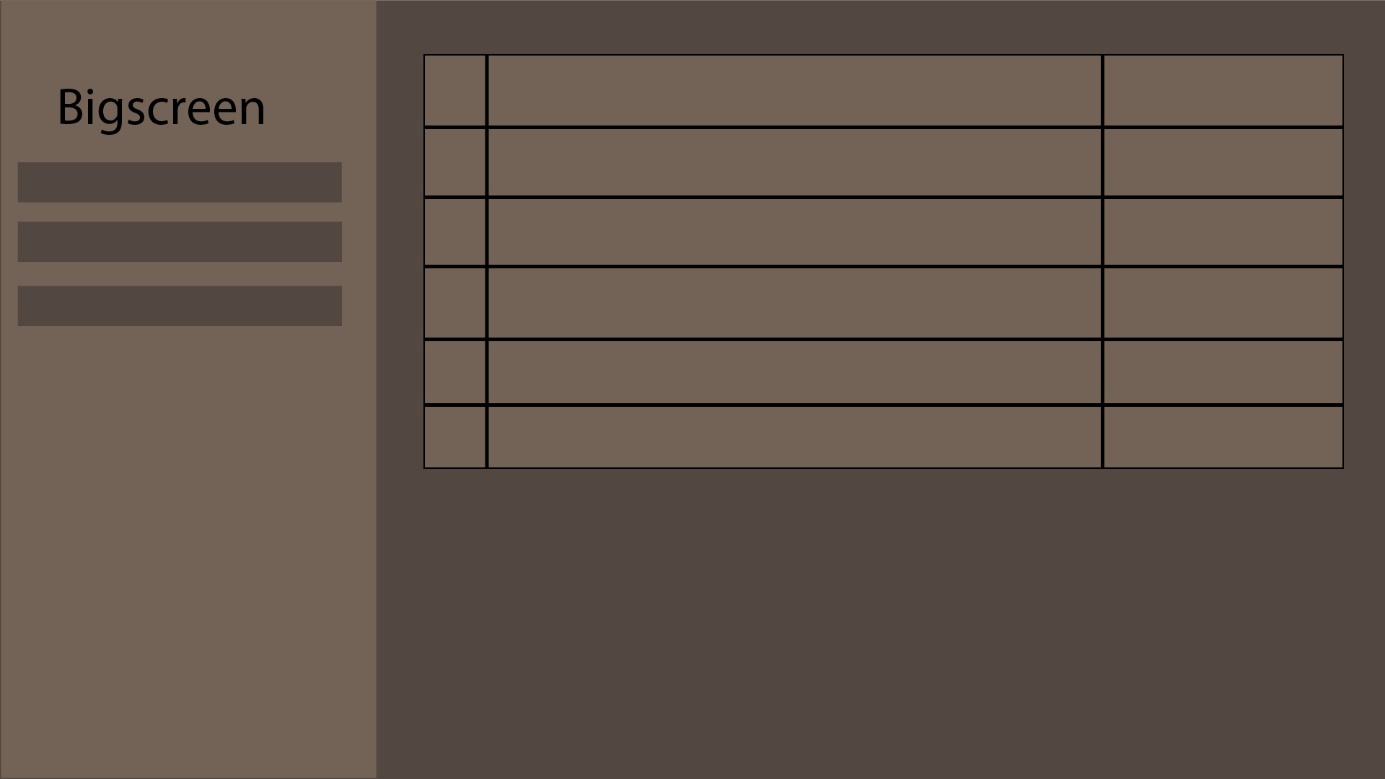
****

**Page Réponses**

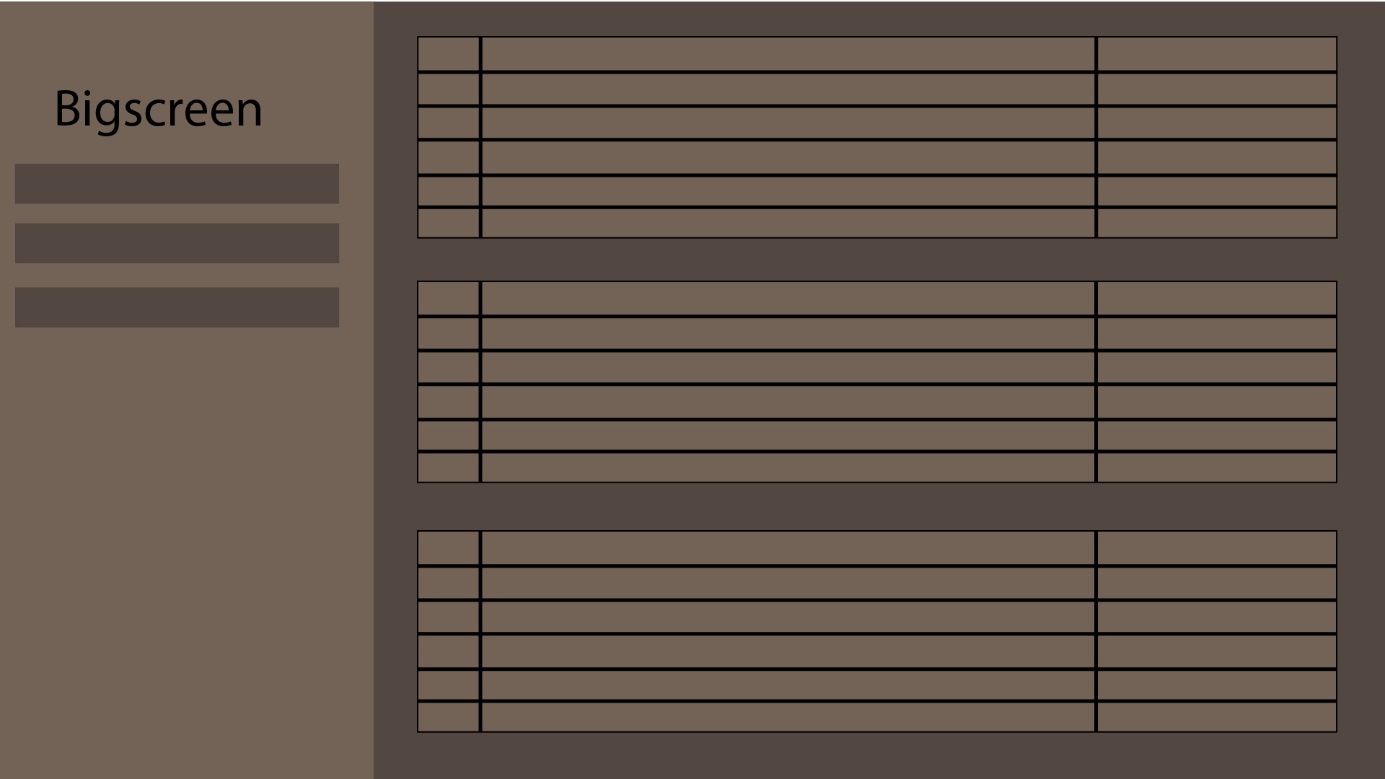
**Page Login**

**Administration**

**Page Accueil**

**Page Questions**

**Page Réponses**

****

**Documentation du code**

**Partie Frontend :**

* **Nom du composant** : **TheSurvey**
* **Type du composant :** Vue
* **Paramètres du composant** :

**Data**:

**. questions** : Tableau des questions récupérées du serveur.

**. currentQuestionIndex** : Index de la question actuellement affichée.

**. currentQuestion** : Question actuelle.

**. responses** : Réponses fournies par l'utilisateur.

**. showError** : pour afficher les erreurs de saisie.

**. mailError** : Indicateur pour afficher les erreurs de saisie d'adresse e-mail.

**. FormError** : Indicateur pour afficher les erreurs générales du formulaire.

**. showPopup** : Indicateur pour afficher ou masquer la fenêtre pop-up.

**. link** : Lien de redirection vers les réponses utilisateur.

**Methode**:

**. getData** (méthode asynchrone): Récupère les questions du sondage depuis le serveur.

**. submitAllResponses** (méthode asynchrone): Permet la soumission de toutes les reponses a l’API et stocke dans une propriété le lien de redirection vers les réponses de l’utilisateur.

**. previousQuestion** (méthode): Affiche la question précédente dans le formulaire.

**. nextQuestion** (méthode): Permet de passer à la question suivante.

**. copyToClipboard** (méthode): méthode pour copier le lien de redirection dans le presse papier.

**. closePopup** (méthode) : méthode pour fermer la fenêtre de pop-up et recharger la page.

* **Nom du composant** : **TheResponse**
* **Type du composant :** Vue
* **Paramètres du composant** :

**Data**:

**. results** : Tableau pour stocker les réponses récupérées du serveur.

**Methode**:

**. getData** (méthode asynchrone): Récupère les données des réponses à partir du serveur.

**. formDate** (méthode): Permet de formater la date dans un format lisible.

**. scrollToTop** (méthode): Fait défiler la page vers le haut de façon fluide.

**. handleScroll** (méthode): Gère l’affichage du bouton de retour en haut de page.

* **Nom du composant** : **TheLogin**
* **Type du composant :** Vue
* **Paramètres du composant** :

**Data**:

**. email** : Champ de saisie de l’adresse e-mail.

**. password** : Champ de saisie du mot de passe.

**. userText** : Indicateur d'affichage du message de bienvenue.

**. adminName** : Nom de l'administrateur connecté.

**. mailError** : Indicateur d'erreur de saisie de l'e-mail.

**. passwordError** : Indicateur d'erreur de saisie du mot de passe.

**. formError** : Indicateur d'erreur de connexion.

**Methode**:

**. Login** (méthode asynchrone) : Effectue là connexion en envoyant les données au serveur.

* **Nom du composant** : **TheAdmin**
* **Type du composant :** Vue
* **Paramètres du composant** :

**Data**:

**. answerGraphic**: Données pour les graphiques.

**. answerQuestion6**: Initialiser les données pour le graphique de la question 6.

**. answerQuestion7**: Initialiser les données pour le graphique de la question 7.

**. answerQuestion10**: Initialiser les données pour le graphique de la question 10.

**. answerQcm11**: Initialiser les données pour les graphiques des questions 11.

**. answerQcm12**: Initialiser les données pour les graphiques des questions 12.

**. answerQcm13**: Initialiser les données pour les graphiques des questions 13.

**. answerQcm14**: Initialiser les données pour les graphiques des questions 14.

**. answerQcm15**: Initialiser les données pour les graphiques des questions 15.

**Methode**:

**. getData** (méthode asynchrone): Permet de récupérer les données nécessaires pour les graphiques depuis le serveur.

**. logout** (méthode asynchrone): Permet de déconnecter l’utilisateur.

* **Nom du composant** : **TheAdminQuiz**
* **Type du composant :** Vue
* **Paramètres du composant** :

**Data**:

**. questions**: Tableau pour stocker les questions récupérées depuis le serveur.

**Méthode**:

**. getData** (méthode asynchrone): Permet de récupérer les questions depuis le serveur.

**. logout** (méthode asynchrone): Permet de déconnecter l’utilisateur.

* **Nom du composant** : **TheAdminAnsswer**
* **Type du composant :** Vue
* **Paramètres du composant** :

**Data**:

**. answers**: Tableau pour stocker les réponses récupérées depuis le serveur.

**Méthode**:

**. getData** (méthode asynchrone) : Permet de récupérer les réponses depuis le serveur.

**. logout** (méthode asynchrone) : Permet de déconnecter l’utilisateur.

* **Librairie :**

1. **Bootstrap**

* **Paramètres :** mise en page créative.

1. **Vue Router**

* **Paramètres :** Navigation, Passages de paramètres, Middleware.

**Partie Backend :**

1. **Fonction :**

* **Login**
* **Rôle :** Permet à l’administrateur de se connecter.
* **Paramètres :** Request, validate, Hash, createToken.
* **showGraphic**
* **Rôle :** Permet à l’administrateur de visualiser les réponses.
* **Paramètres :**
* **questionList**
* **Rôle :** Permet à l’administrateur de visualiser les questions du sondage.
* **Paramètres :**
* **surveyedResponses**
* **Rôle :** Permet à l’administrateur de visualiser les questions du sondage ainsi que les reponses de chaque sondé.
* **Paramètres :**
* **showQuestions**
* **Rôle :** Permet de récupérer toutes les questions avec leurs choix associés.
* **Paramètres :**
* **responseStore**
* **Rôle :** Permet l’enregistrement des reponses et la création d’un token associé dans la base de données.
* **Paramètres :** Request, validate, uuid.
* **userResponse**
* **Rôle :** Permet de récupérer des reponses associé à un token dans la base de données.
* **Paramètres :** token.

1. **Table**

* **Nom :** users
* **Champs :** id, name, email\_verified\_at, password, remember\_token, created\_at, updated\_at
* **Rôle :** Cette table est celle qui stocke l’identifier de l’administrateur et lui permet de se connecter.
* **Nom :** personal\_acces\_tokens
* **Champs :** id, tokenable\_type, tokenable\_id, name, token, abilities, last\_used\_at, espires\_at, created\_at, updated\_at
* **Rôle :** Cette table est celle qui stocke le token personnel de l’administrateur générer à chaque connexion.
* **Nom :** surveys
* **Champs :** id, title, created\_at, updated\_at
* **Rôle :** Cette table est celle qui stocke le sondage lorsqu’il est créé.
* **Nom :** questions
* **Champs :** id, question\_number, question\_body, question\_type, survey\_id, created\_at, updated\_at
* **Rôle :** Cette table est celle qui stocke les questions de chaque sondage à l’aide d’une clé étrangère.
* **Nom :** responses
* **Champs :** id, response\_text, question\_id, user\_token\_id, survey\_id, created\_at, updated\_at
* **Rôle :** Cette table est celle qui stocke les réponses aux questions de chaque sondé à l’aide d’une clé étrangère.
* **Nom :** choices
* **Champs :** id, choice\_text, question\_id, created\_at, updated\_at
* **Rôle :** Cette table est celle qui stocke les choix des réponses aux questions de types A qui sont des questions avec plusieurs propositions.
* **Nom :** user\_tokens
* **Champs :** id, token, created\_at, updated\_at
* **Rôle :** Cette table est celle qui stocke le token de chaque utilisateur après avoir répondu à un sondage.