Projet final





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
12. **Description de projet** :

**Bigscreen** est une entreprise qui met à la disposition de ses clients une application de **réalité virtuelle** leurpermettant 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** souhaiterais mettre en place un sondage pour recueillir l’avis de ses utilisateurs.

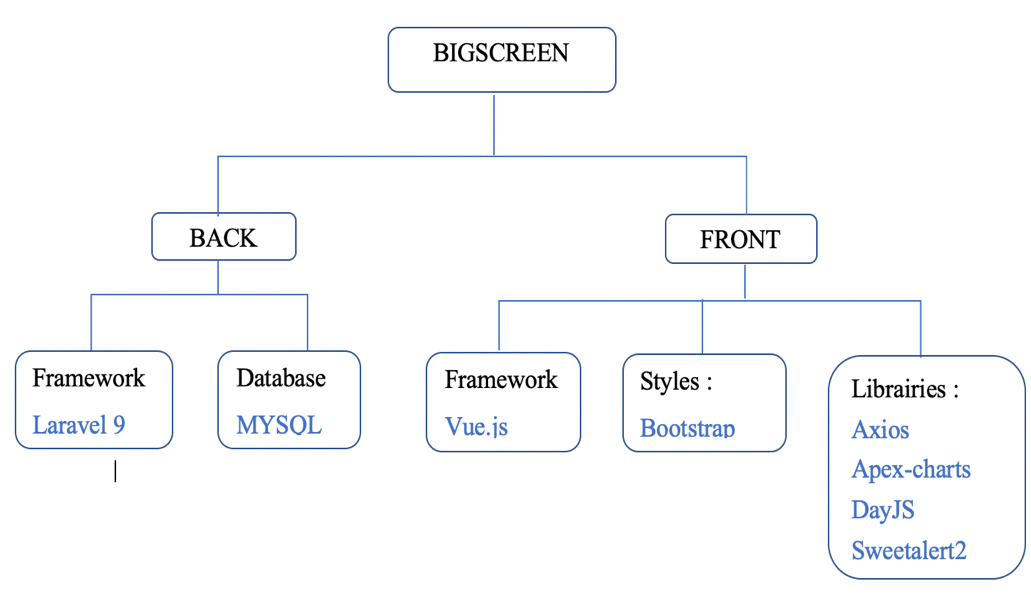
C’est dans ce sens que l’entreprise nous a confié la responsabilité de développer une application de sondage destinée à leurs utilisateurs pour permettre de connaitre leur avis.

1. **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 components 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 **Boostrap** 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 formattage 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.





1. **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 :

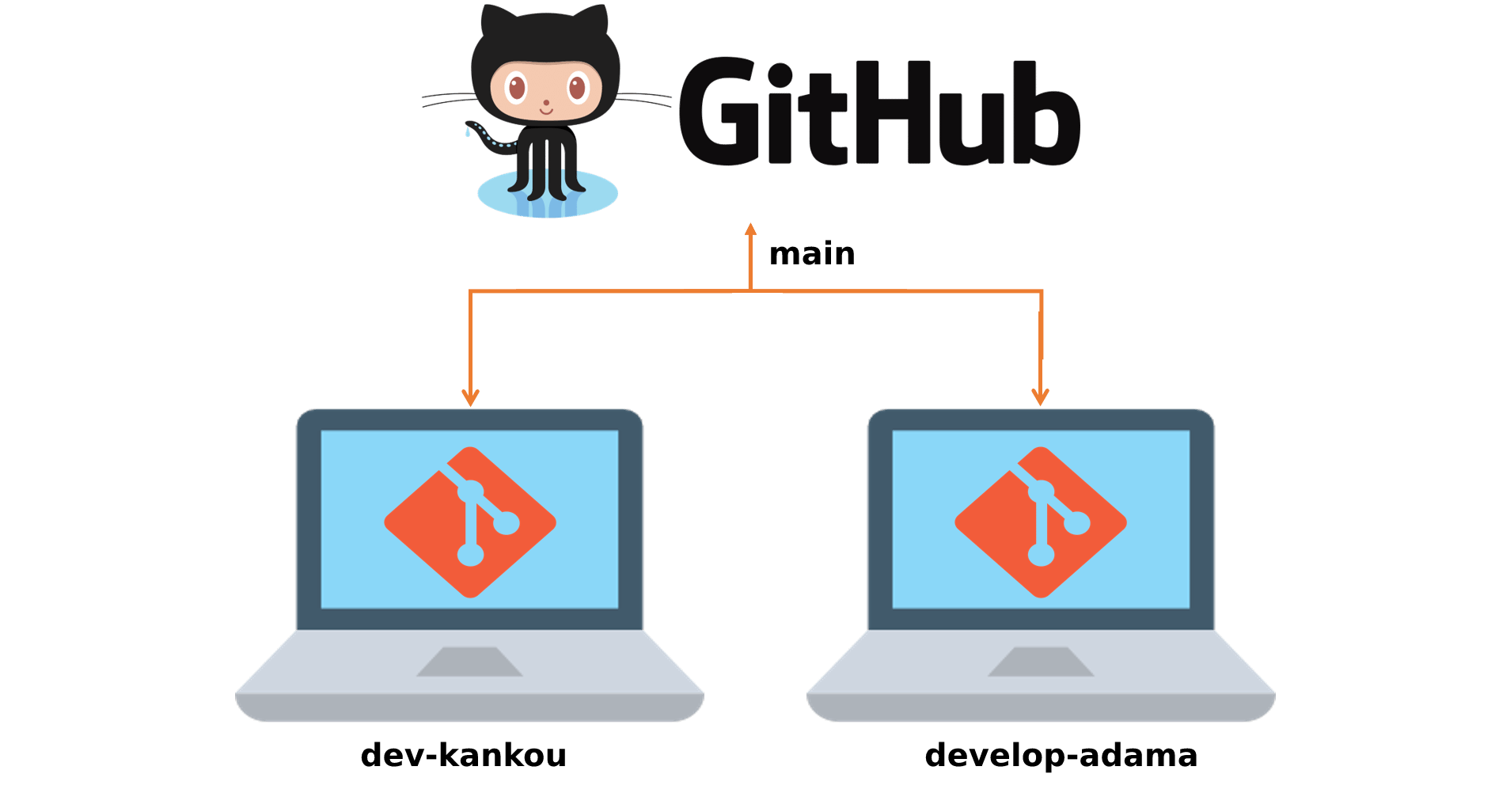
* User: [admin@gmail.com](mailto:admin@gmail.com)
* Mdp: password

1. **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.

1. **Outils Utilisés :**

Pour mener à bien nos différentes taches, 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 taches
* **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

1. **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. **QuestionController : index()**

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

* 1. **AnswerController : index()**

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

* 1. **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. **AnswerController : getVisitorResponse()**

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

* 1. **VisitorController : index()**

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

* 1. **BackController : getPieChart()**

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

* 1. **BackController : getRadarChart()**

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

* 1. **AuthController : login()**

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

* 1. **AuthController : logout()**

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

1. **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.

1. **Temps de Travail :**

|  |  |
| --- | --- |
| **BACK :** | 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 |
|  |  |
| **FRONT :** |  |
| 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 |

1. **Recettage:**

|  |  |
| --- | --- |
| **BACK :** | 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 |
| **FRONT :** | 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 |

1. **Documentation API :**

Liste des questions

Méthode : **GET**

Paramètres : Aucun

**URL** : http://127.0.0.1:8000/api/questions

Exemple de Réponse :



Liste de toutes les réponses

Méthode : **GET**

Paramètres : Aucun

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

Exemple de Réponse :



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 :



Connexion

Méthode : **POST**

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

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

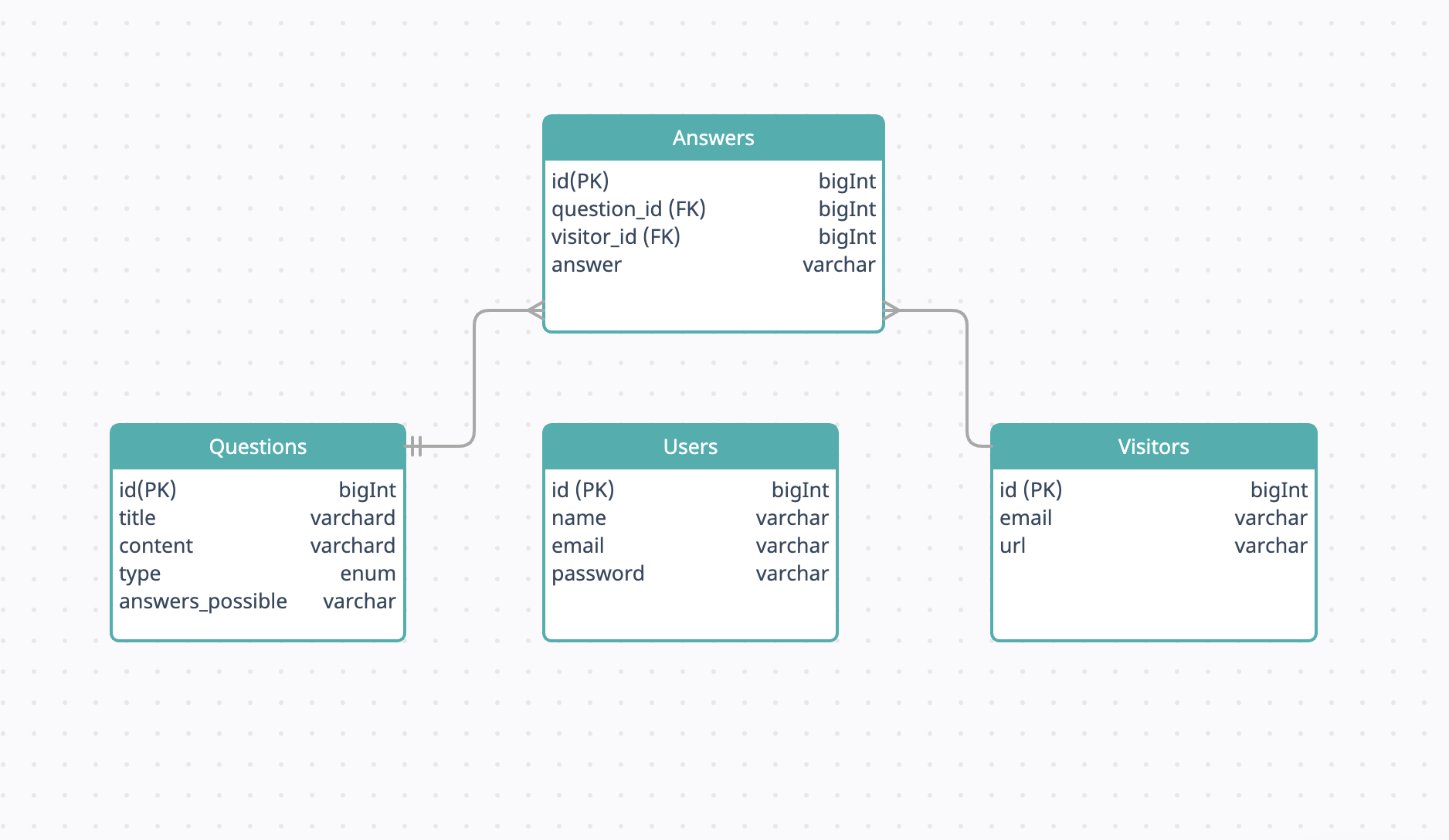
Déconnexion

Méthode : **POST**

Paramètres : Aucun

**URL** : http://127.0.0.1:8000/api/logout

1. **Modélisation des tables :**



1. **Wireframes :**

Page de Sondage

Page Réponses

Formulaire de Connexion

Accueil Dashboard

Tableau de Questionnaire

Tableau des Questions et leurs réponses