Cahier de charges
Bigscreen

Sommaire

1. Présentation:

Bigscreen est une entreprise qui développe une application VR, afin que les utilisateurs puissent regarder en VR des films, émissions TV et des jeux vidéo.
c'est un projet final que nous avons développé en binôme Sodi et Mick

2. Analyse du client:

Le client désire collecter des informations sur des sondés afin d'afficher les réponses de chaque utilisateur, permettre à l'administrateur de bigscreen via son adresse mail et son mot de passe de s'authentifier pour accéder à la partie privée, visualiser les différents graphes, voir la liste des questions et voir les réponses des sondés.

3. Choix technologiques:

Le choix technologique de notre projet est subdivisé en deux parties :

- Pour le back-end nous avons utilisé laravel
- Pour le front-end nous avons utilisé utilisé blade le moteur de template de laravel pour afficher la vue, tailwind css qui est un framework css et

qui met à disposition des composants afin de faciliter la création de l'interface

4. Adresse Github:

Le lien github de notre projet est le suivant :
https://github.com/souleymane-diallo/Bigscreen

5. Déploiement:

bigscreen :

- .env:

changer les variables d'environnements :
DB_DATABASE
DB_USERNAME
DB_PASSWORD

- Installation des dépendances:

cd bigscreen && composer install && npm install

- Lancer les migrations et les seeders:

php artisan migrate:fresh –seed

- Lancement du serveur:

php artisan serve && npm run dev

- Identifiants:

email: admin@gmail.com mot de passe: password

6. Méthode de travail:

Dans le cadre de notre travail nous avons choisi une méthode agile . Cette approche nous a permis de traiter une fonctionnalité après une autre jusqu'à atteindre l'objectif final.Elle a favorisé la communication au centre pour lister les tâches à exécuter.
Nous avons utilisé gitflow pour gérer le fonctionnement de nos branches de Git.

Voici le principe de base :

- Notre projet sera basé sur deux branches : main et test.
- La branche main est le miroir de notre production. Il est donc logique que l'on ne puisse y pousser nos modifications directement.
- La branche test centralise toutes les nouvelles fonctionnalités qui seront livrées dans la prochaine version. Ici, il va falloir se forcer à ne pas y faire de modifications directement.

Deux autres types de branches vont ensuite nous permettre de travailler :

- dev-sodi
- dev-mick

7. Outils utilisés:

   Nous avons utilisés quelques outils pour la réalisation de notre travail:

   Logiciels

   - Trello : Trello est l'outil visuel qui nous permet de gérer notre projet ou de suivre les tâches.
   - github : GitHub nous permet de stocker et de partager publiquement le code

     .

   Outils et services de communication

   - Google Meet : Google nous a permis de créer de visioconférence permet de créer une visioconférence
   - Whatsapp : Whatsapp nous a permis à faire des appels pour pouvoir se communiquer

   Outils de développement

   - MySQL Workbench : MySQL Workbench est le logiciel qui nous a permis notre diagramme de modèle de base de données
   - Visual studio code : Visual studio code est l'éditeur de code qui nous a permis d'implémenter notre projet.

8. Evaluations du temps de travail:

Entrée	Sortie
Modélisation des tables	1 jour
création de seeder pour les questions	1 jour
Création de seeder pour les réponses	1 jour
wireframe et zoning	1 jour
Authentification avec breeze	1 jour
enregistrement des réponses du songé	1 semaine
affichage des graphes	1 semaine

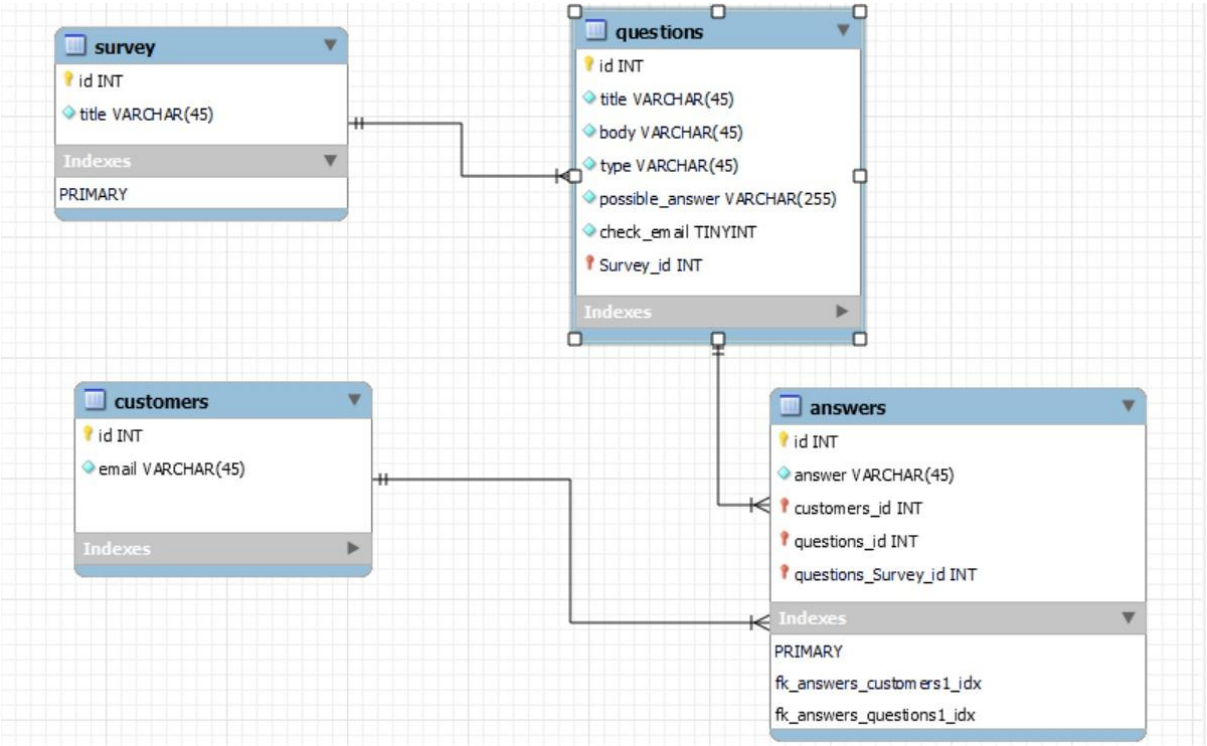
9. Liste fonctionnelle:

Public	Administration
Formulaire de sondage	Authentification
Consulter la réponse d’un sondé	Visualiser les statistiques des réponses
message de confirmation	Visualiser la liste des questions
	Visualiser les réponses des utilisateurs

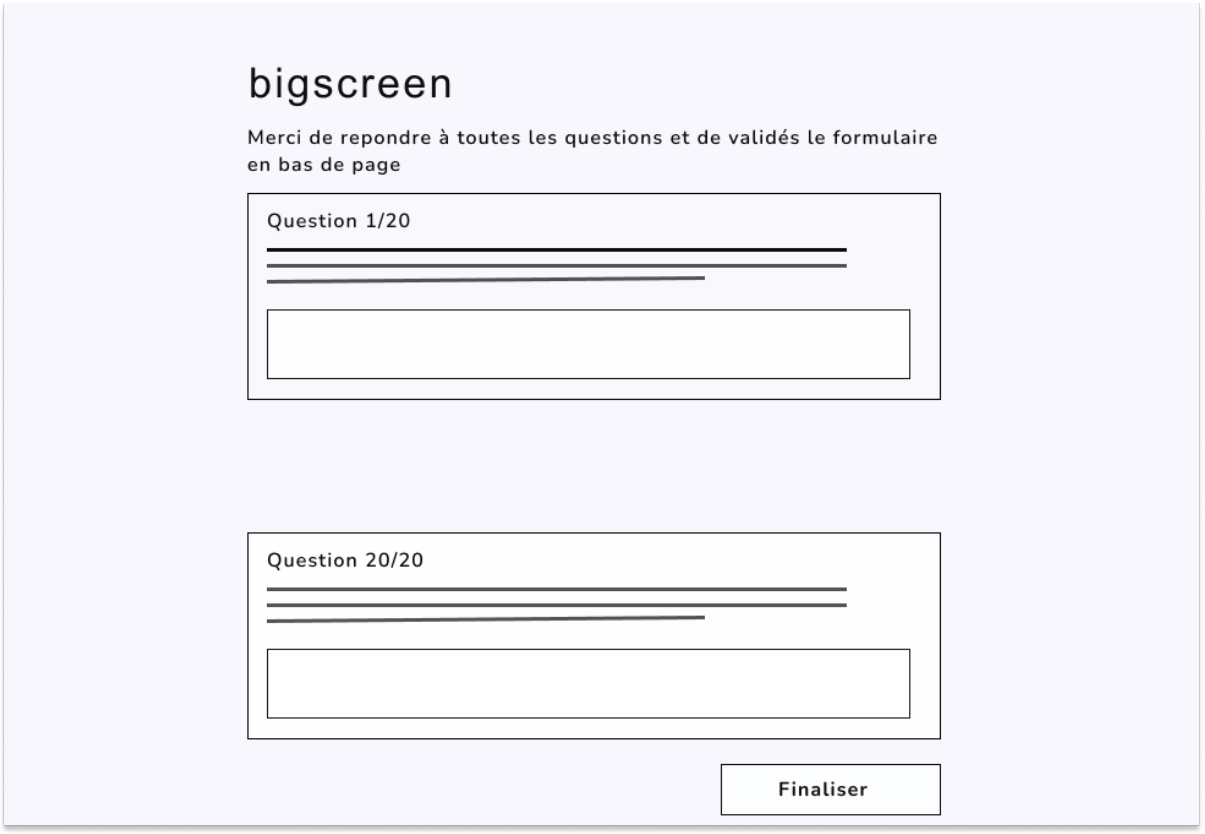
10. Recettage:

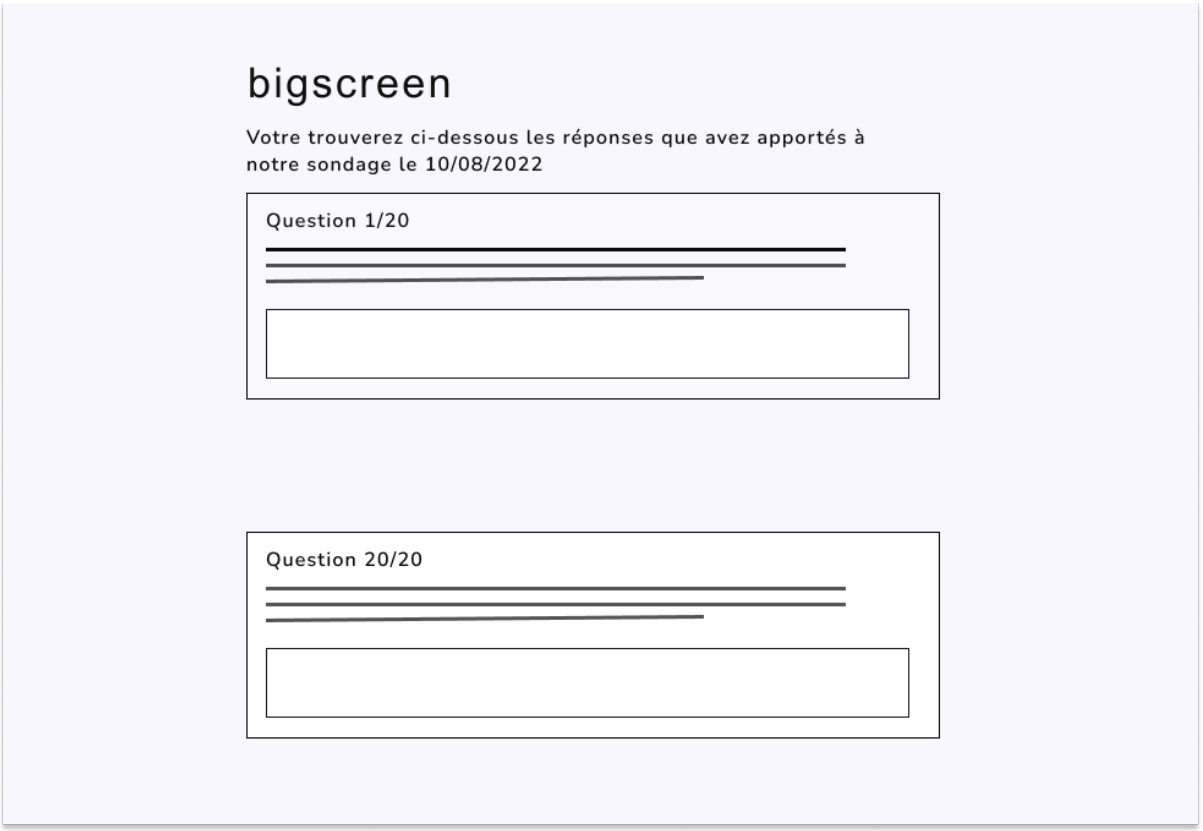
Fonctionnalités	opérationnelle	possède un bug	possède une limitation
Récupérer les 20 questions	oui		
Enregistrer un sondage	oui		
Authentification	oui		
Consulter les statistiques des réponses	oui		
Afficher les questions	oui		
Afficher les réponses des sondés	oui		
Pagination	oui		

11. Diagramme des bases de données:



12. Wireframes:

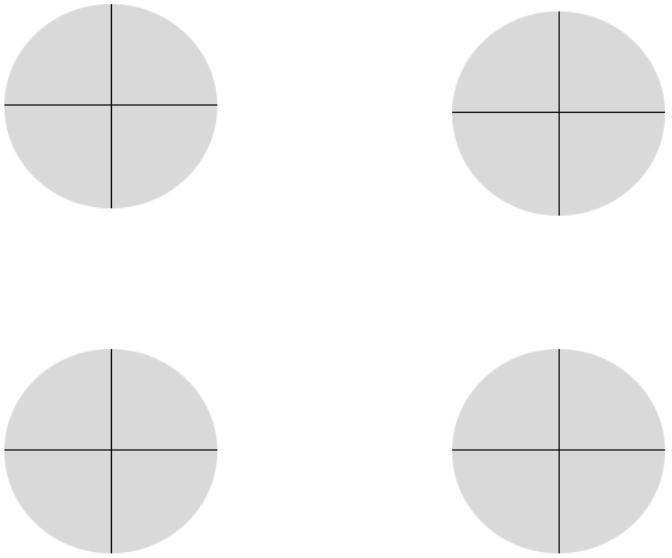



bigscreen
Votre trouverez ci-dessous les réponses que avez apportés à notre sondage le 10/08/2022
Question 1/20
Question 20/20


bigscreen
Email
Mot de passe
Connexion

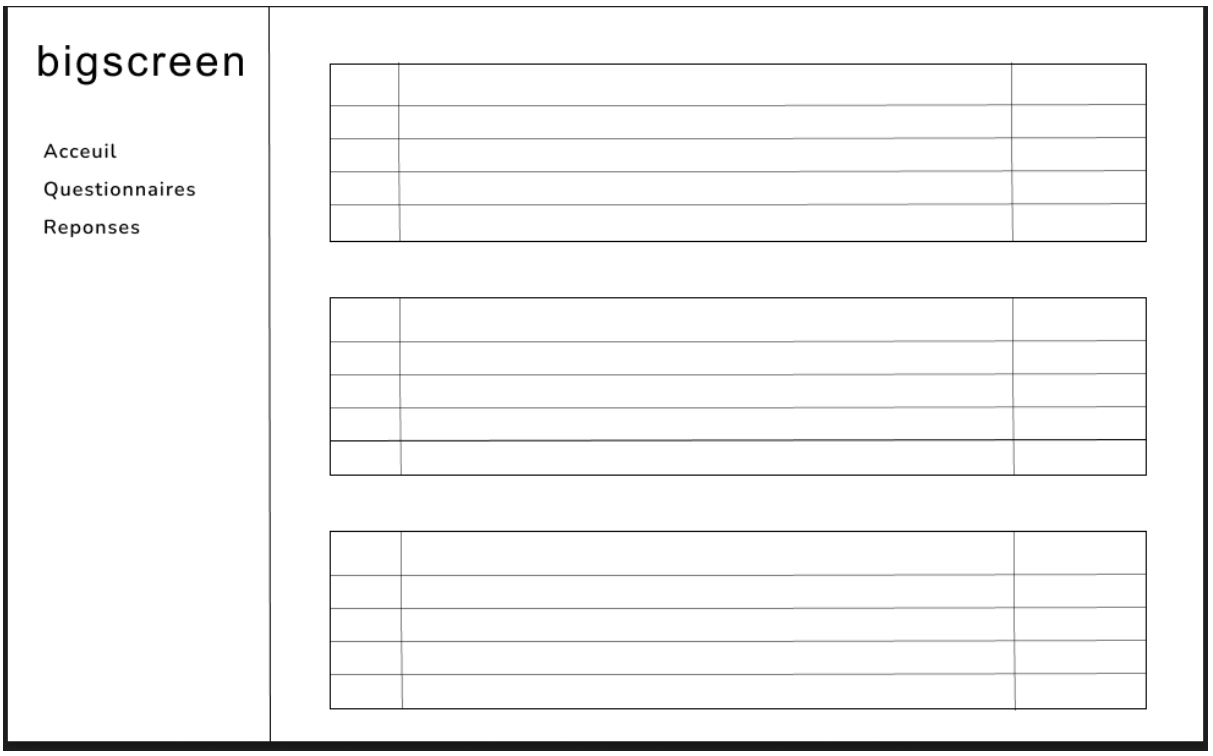
bigscreen

Acceuil
Questionnaires
Reponses



bigscreen

Acceuil
Questionnaires
Reponses



13. Documentation du code:

Backend

Les différentes classes implémentées sont les suivantes :

- AnswerController

Rôle : Cette classe permet d'implémenter les méthodes qui interagissent entre la base de données et la vue index de la partie frontend.
Méthode : store()

- DashboardController

Rôle : Cette classe permet d'implémenter les méthodes qui interagissent entre la base de données et les différentes vues de la partie backend.
Méthodes : index() ,pieChart() , radarChart() ,questionnaires() et answers ()

- FrontController

Rôle : Cette classe permet d'implémenter les méthodes qui interagissent entre la base de données et les différentes vues de la partie frontend.

Méthodes : index() ,answers() et message()

- Answer

Rôle : Cette classe permet de représenter la structure logique et de concrétiser les relations entre les tables . c'est le modèle de la table answers
Méthodes : question() ,customer() et HasPath

- Customer

Rôle : Cette classe permet de représenter la structure logique et de concrétiser les relations entre les tables . c'est le modèle de la table customers

Méthodes : answers() et Email()

- Question

Rôle :Cette classe permet de représenter la structure logique et de concrétiser les relations entre les tables . c'est le modèle de la table questions

Méthodes : answers() et survey()

- Survey

Rôle : Cette classe permet de représenter la structure logique et de concrétiser les relations entre les tables . c'est le modèle de la table surveys

Méthodes : question()

Les différentes Méthodes implantées sont les suivantes :

- AnswerController : store()

Paramètre : cette méthode prend en paramètre une de variable de type AnswerRequest
Rôle : Permet d'insérer toutes les réponses d'un sondé
Valeur de retour : Ne retourne aucune valeur mais nous redirige vers la route message

- DashboardController : index()

Paramètre : Ne prend aucun paramètre
Rôle : Affiche trois pie Charts et radar Chart
Valeur de retour : retourne deux tableaux : un tableau de pieChart et un autre de radar Chart

- DashboardController : pieChart ()

Paramètre : prend une variable en paramètre (un nombre de type string)

Rôle : permet de définir le nombre fois qu'une réponse a été choisi par rapport à une question
Valeur de retour : renvoie un tableau qui contient le numéro de la question la question,un tableau de réponses possibles , un tableau de nombre de fois , un tableau de couleur.

- DashboardController : radarChart ()

Paramètre : prend un tableau en paramètre
Rôle : permet de faire la moyenne des réponses des questions passées en paramètres dans le tableau
Valeur de retour : retourne deux tableaux

- DashboardController : questionnaires ()

Paramètre : Prend aucun paramètre
Rôle : Récupère toutes les questions dans la base de données
Valeur de retour : retourne un tableau de question

- DashboardController : answers ()

Paramètre : Prend aucun paramètre
Rôle : Affiche les questions et les réponses d'un sondé
Valeur de retour : renvoie deux tableaux : un tableau des questions et l'autre des réponses d'un sondé

- FrontController : index()

Paramètre : Ne prend aucun paramètres
Rôle : Récupère toutes les questions
Valeur de retour : retourne un tableau

- FrontController : answers ()

Paramètre : Prend une variable de type String en paramètre
Rôle : Montre les réponses d'un sondé

Valeur de retour : renvoie deux tableaux : les tableaux des questions et des réponses

- FrontController : message ()

Paramètre : Ne prend aucun paramètre
Rôle : renvoie vue qui affiche le message au sondé après avoir finaliser
Valeur de retour : Ne retourne aucune valeur

- Answer : question ()

Paramètre : Ne prend aucun paramètre
Rôle : Permet de définir qu'une réponse appartient à une question

Valeur de retour : Ne retourne aucune valeur

- Answer : customer ()

  Paramètre : Ne prend aucun paramètre
  Rôle : Permet de définir qu'une réponse appartient à un sondé
  Valeur de retour : Ne retourne aucune valeur

- Answer : HasPath ()

  Paramètre : prend en paramètre une variable
  Rôle : permet de faire une requête en fonction de la variable passée paramètre
  Valeur de retour : Ne retourne aucune valeur

- Customer : answers ()

  Paramètre : Ne prend aucun paramètre
  Rôle : Permet de définir qu'un sondé a plusieurs réponses
  Valeur de retour : Ne retourne aucune valeur

- Customer : Email ()

  Paramètre :prend en paramètre une variable
  Rôle : permet de faire une requête en fonction de la variable passée paramètre
  Valeur de retour : Ne retourne aucune valeur

- Question : answers ()

  Paramètre : Ne prend aucun paramètre
  Rôle : Permet de définir qu'une question a plusieurs réponses
  Valeur de retour : Ne retourne aucune valeur

- Question : survey ()

  Paramètre : Ne prend aucun paramètre
  Rôle : Permet de définir qu'une question appartient à un sondage
  Valeur de retour : Ne retourne aucune valeur

- Question : AnswerPossible ()

  Paramètre : prend en paramètre une variable
  Rôle : permet de faire une requête en fonction de la variable passée paramètre
  Valeur de retour : Ne retourne aucune valeur

- Survey : question ()

Paramètre : Ne prend aucun paramètre
Rôle : Permet de définir qu'un sondage a plusieurs questions
Valeur de retour : Ne retourne aucune valeur

Table de la base de données

Nom de la table **:** questions

Rôle : La table questions stocke toutes les questions du sondage

Champs	**Type**
id	bigInteger (clé primaire)
title	string
body	string
type	enum
possible_answer	string
check_email	boolean

Nom de la table **:** answers

Rôle : La table answers stocke toutes les réponses des sondés

Champs	**Type**
id	bigInteger (clé primaire)
answer	String

Nom de la table **:** customers
Rôle : La table customers stocke les adresses emails de tous les sondés pour les identifier de manière unique.

Champs	**Type**
id	bigInteger (clé primaire)
email	string

Nom de la table **:** surveys

Rôle : La table surveys stocke les différents sondages

Champs	**Type**
id	bigInteger (clé primaire)
title	string