REDEMPTION

Rapport final
Projet Semestre 6

Membres de l'équipe :

Martin BOUTEILLER

Joao BRILHANTE

Olivier DOUSSAUD

Valentin ROCCELLI

UNIVERSITÉ CÔTE D'AZUR

POLYTECH NICE-SOPHIA

Table des matières

I. Personas et scénarios

1. Personas

a. Françoise

Françoise, 81 ans et est atteinte d'Alzheimer mais elle sait se servir des technologies tactiles. Cependant, étant donné son âge avancé, sa vision s'est beaucoup dégradée. Elle aimerait que tout soit facile à distinguer sur l'écran. Elle a déjà joué à des quiz sur une tablette et avait beaucoup apprécié l'expérience. Seulement, un jour, Françoise était tombée sur une question qu'elle n'arrivait pas à passer, malgré le fait qu'elle donne "la bonne réponse". Le quiz ne manquait pas alors de lui rappeler avec l'apparition d'une grosse croix rouge. Elle voudrait que le quiz ne la bloque pas pendant son activité.

b. Louise

Louise, 32 ans, est aide médico-psychologique au sein d'un centre d'accueil de jour d'Alzheimer près de chez elle. Elle souhaiterait que les personnes accueillies puissent s'occuper en faisant des quiz pour solliciter leur mémoire. De plus, elle aimerait pouvoir personnaliser les quizz pour s'adapter à tous les accueillis. Enfin, elle voudrait pouvoir visualiser les résultats et la progression de chacun pour pouvoir améliorer les prochains quiz.

c. Roger

Roger, 27 ans, est assistant médical au sein d'un centre d'accueil de jour et a remarqué très vite que les quiz sont un bon outil pour stimuler les accueillis. Pour garantir la participation dans le calme, il crée des profils en fonction des handicaps visuels de chacun. Grâce à cela chaque accueilli a une approche personnalisée des quizz auxquels il répondra.

2. Scénarios

a. Françoise

Françoise est accompagnée par une aide médicale pour sélectionner son quiz, c'est cette dernière qui va cliquer sur "Lancer un quiz", puis sélectionner le thème "Animaux" qui intéresse Françoise, ensuite elle lancera le quiz sur les mammifères et donnera la main à Françoise qui va répondre aux questions une par une (elle se trompera sur certaines et prendra des fois beaucoup de temps pour réfléchir, ce qui aura pour effet de faire apparaître un bouton "Passer à la question suivante" qu'elle utilisera de temps en temps). A la fin du quiz elle sera félicitée et rendra l'outil à l'aide médicale.

b. Louise

Louise va sélectionner "Voir les résultats" et choisir le dernier quiz qui vient d'être fait par Françoise. Après avoir analysé les résultats elle va retourner dans le Menu principal en cliquant sur "Menu principal" en haut à gauche et choisira "Gérer les quiz" pour aller modifier le quiz qui vient d'être fait par Françoise. Elle pourra alors modifier le thème, le quiz ou encore chaque question et les réponses qui pourraient avoir posées un problème.

c. Roger

Roger choisit le menu "Gérer les accueillis" et ajoute un nouvel accueilli en entrant son nom et son prénom puis sélectionnera le profil d'accessibilité "Contraste élevé" (qui correspond à un affichage zoomé et un contraste élevé). Une fois son nouveau profil validé il pourra revenir au menu principal et tenter de jouer le quiz pour ce nouvel accueilli et observer les ajustements d'accessibilité fait en conséquence du profil choisi lors de la création.

II. Présentation de l'architecture client/serveur

Nous avons décidé de concevoir notre projet en utilisant des **conteneurs Docker** pour simplifier le lancement, le déploiement sur n'importe quel serveur mais également l'installation des dépendances et l'exécution des migrations. Notre projet est alors composé de trois conteneurs : le client, le serveur et la base de données.

1. Présentation du client

Notre client se compose de trois parties majeures. La première partie est la plus importante puisqu'il s'agit de la partie permettant à un accueilli de **jouer un quiz**. La deuxième partie est celle permettant la **visualisation des résultats** des accueillis. La troisième partie est celle permettant l'**administration des ressources** du site. Enfin, la dernière partie est celle qui permet la liaison entre le client et le serveur à travers des **services**.

a. Jouer un quiz

La partie du client permettant de jouer un quiz est composée de **cinq composants**, dont **trois pour la sélection** de l'accueilli, du thème et du quiz. Les **deux composants restants** permettent de **jouer au quiz**. Le premier récupère le quiz sélectionné, avec ses questions, ses réponses et ses images. Il se charge également de savoir quelle question est en cours, de l'envoyer au second composant ainsi que d'afficher l'écran de transition entre chaque changement de question. Le second composant s'occupe d'afficher la question que son parent lui a transmise et de gérer les animations survenant quand un accueilli sélectionne une réponse. Enfin, il est également responsable de la collecte d'informations qui seront envoyées au composant parent, qui se chargera d'envoyer ses informations au serveur pour permettre la visualisation des résultats.

b. Visualiser un résultat

Pour la partie permettant la **visualisation des résultats**, nous avons réutilisé le composant de sélection d'un accueilli. Ensuite, nous avons un composant affichant, à la volonté des aides-médicales, les **résultats un à un, triés par date de complétion**, ou les **résultats moyens par quiz**, permettant ainsi aux aides de savoir plus facilement quel quiz pourrait poser un problème.

En réalité, ces deux options sont réalisées par deux composants. En effet, au départ, tout le code était concentré dans le composant parent. Seulement, avec toutes les données requises pour l'affichage et la gestion du changement, le composant avait trop de responsabilités. Nous avons donc fait le choix de décomposer ce dernier en deux composants plus petits, en lui laissant la responsabilité de gérer la transition entre les

deux nouveaux composants, ainsi que de récupérer les résultats d'un accueilli afin de les transmettre à ces derniers.

De plus, nous avons également la possibilité de visualiser, pour un résultat en particulier, le détail des résultats à ce quiz, avec chaque réponse sélectionnée par l'accueilli et si elle a été sautée.

c. Administration des ressources

La partie permettant l'**administration des ressources,** par les aides médico-psychologiques, est la plus complète du client. En effet, cette partie est composée de 25 composants. Parmi ces composants, il y a notamment les **listes de gestion** des ressources (images, thèmes, quiz et accueillis). Ces listes sont elles-mêmes constituées de sous-composants propres à chaque ressource. De plus, nous avons opté pour l'affichage des ressources sous forme de grands éléments identifiables à l'aides d'images, de gros titres, et de boutons de gestions (modifier et supprimer) pour une bonne utilisabilité. Enfin, nous avons ajouté une pagination aux listes pour permettre un meilleur affichage sur des dispositifs tels que des tablettes.

Parmi les composants restants, il y a les **formulaires de gestion** des ressources. Ces-derniers sont développés à l'aide des outils fournis par les *Reactive Forms*. Ces outils nous permettent notamment d'écouter en temps réel et de valider les champs des formulaires. Enfin, ces formulaires sont utilisés à la fois pour la création et la modification des ressources pour éviter au maximum la redondance dans le code.

d. Services

La dernière partie est celle permettant au client d'interagir avec le serveur. En effet, il s'agit des nombreux **services** que nous avons mis en place pour exécuter les différentes requêtes et récupérer les données nécessaires au fonctionnement du client. Pour ces services, nous avons utilisé des observables, fournis par la bibliothèque RxJS, pour **stocker les ressources et notifier les composants souscrits en cas de changement**. Chaque ressource possède alors son service dédié qui est accessible depuis n'importe quel composant.

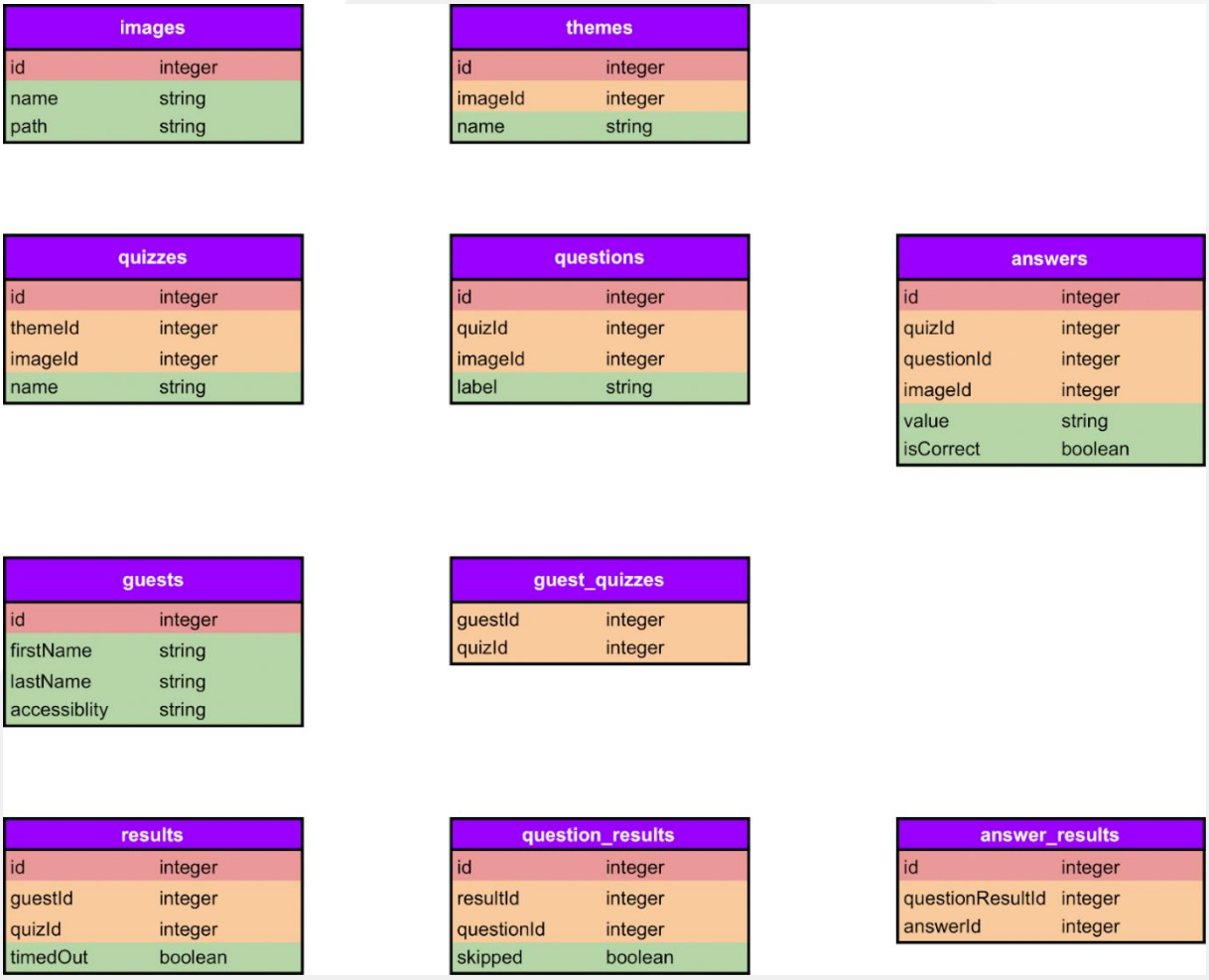
2. Présentation de la base de données

Au début du projet, nous nous sommes mis d'accord pour utiliser une **base de données SQL** plutôt qu'un stockage à l'aide de fichiers ou une base de données NoSQL. Plus précisément, nous avons choisi une base de données sous PostgreSQL car il s'agit d'un système de gestion de bases de données réputé comme fiable, robuste et apportant de nombreuses fonctionnalités avancées en cas de besoin.

Nous avons fait ce choix pour diverses raisons :

- Les bases de données SQL sont optimisées pour une grande quantité de données et de requêtes.
- Les données prennent moins de place sur le stockage.
- Les données peuvent être utilisées en parallèle par plusieurs applications.
- Les données sont plus sécurisées avec un système de permissions.
- Les données ont une garantie d'intégrité à chaque requête.

Nous avons ensuite utilisé nos connaissances du cours de base de données pour concevoir un schéma relationnel simple et robuste :



3. Présentation du serveur

Notre serveur est basé sur une **architecture MVC (Model, View, Controller)** sans View puisqu'il s'agit du rôle du client. Cette architecture permet notamment de bien séparer les tâches, de gagner du temps avec la réutilisabilité de chaque composant et une grande souplesse pour répartir les tâches entre les membres de l'équipe.

En outre, nous avons tenu à respecter les principes de conception d'une **API REST**. En effet, cette architecture permet de réduire le couplage entre le client et le serveur pour une meilleure réutilisabilité de notre code. Par la même occasion, elle permet d'uniformiser les API pour une facilité d'utilisation si une autre équipe de développement devait reprendre notre code.

Par ailleurs, nous avons utilisé **Sequelize**, un **ORM (Mapping Objet-Relationnel) Node.js pour les bases de données SQL**. Ce-dernier facilite grandement l'implémentation d'une base de données SQL avec notre serveur. En effet, il suffit de définir les modèles et les tables des différentes ressources pour que cet outil fournisse de nombreuses méthodes facilitant les différentes requêtes nécessaires.

Toutes ces décisions, nous ont permis de conserver une hiérarchie simple et lisible :

- Le dossier « **models** » contient tous les fichiers définissant le contenu et les liens entre les différents objets.
- Le dossier « **controllers** » contient tous les fichiers définissant les méthodes utiles agissant sur les différents objets. Ces fichiers définissent alors la logique du serveur.
- Le dossier « **routes** » contient tous les fichiers définissant les différents points d'accès du serveur. Ces fichiers font alors appel aux différentes méthodes des contrôleurs.
- Le dossier « **migrations** » contient tous les fichiers définissant le schéma relationnel de la base de données.
- Enfin, le dossier « **seeders** » contient tous les fichiers définissant les données de développement. Il s'agit de données permettant de tester le bon fonctionnement du serveur et du client.

Ensuite, les différents objets que nous manipulons sont :

- L'objet « **image** » qui contient un nom et un lien vers une image.
- L'objet « **theme** » qui contient un nom, un éventuel identifiant d'image et permet de regrouper plusieurs quiz.
- L'objet « **quiz** » qui contient un nom, un éventuel identifiant d'image, l'identifiant du thème auquel il appartient et permet de regrouper plusieurs questions.
- L'objet « **question** » qui contient le texte de la question, un éventuel identifiant d'image, l'identifiant du quiz auquel il appartient et permet de regrouper plusieurs réponses. Il est important que dans l'optique de servir des accueillis ayant des troubles de la mémoire, **il ne peut pas y avoir plus de 4 réponses par question**.
- L'objet « **answer** » qui contient la valeur de la réponse, un booléen pour vérifier si la réponse est juste ou non et un éventuel identifiant d'image.
- L'objet « **guest** » qui représente un accueilli du centre d'accueil de jour et contient son nom, son prénom, un éventuel profil d'accessibilité et tous les quiz auquel il est associé.

De plus, pour les résultats d'un quiz, nous utilisons :

- L'objet « **answer-result** » représente une réponse sélectionnée par un accueilli pendant la lecture d'un quiz.
- L'objet « **question-result** » regroupe toutes les réponses sélectionnées par un accueilli ainsi qu'un booléen pour vérifier si la question a été sautée.
- Enfin, l'objet « **result** » regroupe tous les résultats des questions d'un accueilli pendant la lecture d'un quiz. Il contient également un booléen pour vérifier si le quiz a été arrêté en cours de route (time out). Cet objet permet de conserver les différentes **données utiles aux aides médico-psychologiques pour déterminer les quiz qui posent un problème à un accueilli**.

Pour terminer, voici les différents points d'accès de notre serveur :

Method	URL	Description	Parameters
GET	/status	Check if the server works properly.	
GET	/images	Find all the images.	
POST	/images	Create an image.	name, path
GET	/images/{imageId}	Find an image by id.	
PUT	/images/{imageId}	Update an image by id.	name, path
DELETE	/images/{imageId}	Delete an image by id.	
GET	/themes	Find all the themes.	
POST	/themes	Create a theme.	name, [imageId]
GET	/themes/{themeId}	Find a theme by id.	
PUT	/themes/{themeId}	Update a theme by id.	name, [imageId]
DELETE	/themes/{themeId}	Delete a theme by id.	
GET	/quizzes	Find all the quizzes.	
POST	/quizzes	Create a quiz.	name, themeId, [imageId]
GET	/quizzes/{quizId}	Find a quiz by id.	
PUT	/quizzes/{quizId}	Update a quiz by id.	name, themeId, [imageId]
DELETE	/quizzes/{quizId}	Delete a quiz by id.	
GET	/quizzes/{quizId}/questions	Find all the questions of the quiz.	
POST	/quizzes/{quizId}/questions	Create a question to the quiz.	label, [imageId]
GET	/quizzes/{quizId}/questions/{questionId}	Find a question by id.	
PUT	/quizzes/{quizId}/questions/{questionId}	Update a question by id.	label, [imageId]
DELETE	/quizzes/{quizId}/questions/{questionId}	Delete a question by id.	
GET	/quizzes/{quizId}/questions/{questionId}/answers	Find all the answers of the question.	
POST	/quizzes/{quizId}/questions/{questionId}/answers	Create an answer to the question.	value, isCorrect, [imageId]
GET	/quizzes/{quizId}/questions/{questionId}/answers/{answerId}	Find an answer by id.	
PUT	/quizzes/{quizId}/questions/{questionId}/answers/{answerId}	Update an answer by id.	value, isCorrect, [imageId]
DELETE	/quizzes/{quizId}/questions/{questionId}/answers/{answerId}	Delete an answer by id.	
GET	/guests	Find all the guests.	
POST	/guests	Create a guest.	firstName, lastName, [accessibility]
GET	/guests/{guestId}	Find a guest by id.	
PUT	/guests/{guestId}	Update a guest by id.	firstName, lastName, [accessibility]
DELETE	/guests/{guestId}	Delete a guest by id.	
POST	/guests/{guestId}/quizzes/{quizId}	Add a quiz to the guest.	
DELETE	/guests/{guestId}/quizzes/{quizId}	Remove a quiz from the guest.	
GET	/results	Find all the results.	
POST	/results	Create a result.	guestId, quizId, timedOut
GET	/results/{resultId}	Find a result by id.	
PUT	/results/{resultId}	Update a result by id.	guestId, quizId, timedOut
DELETE	/results/{resultId}	Delete a result by id.	
GET	/results/{resultId}/questions	Find all the question results.	
POST	/results/{resultId}/questions/{questionId}	Create a question result.	skipped
GET	/results/{resultId}/questions/{questionId}	Get a question result by id.	
PUT	/results/{resultId}/questions/{questionId}	Update a question result by id.	skipped
DELETE	/results/{resultId}/questions/{questionId}	Delete a question result by id.	
GET	/results/{resultId}/questions/{questionId}/answers	Find all the answer results.	
POST	/results/{resultId}/questions/{questionId}/answers/{answerId}	Create an answer result.	
DELETE	/results/{resultId}/questions/{questionId}/answers/{answerId}	Delete an answer result by id.	

III. Présentation de l'évaluation croisée et analyse des résultats obtenus

1. Préparation de l'évaluation croisée

Pour l'évaluation croisée, nous avons choisi la **méthode numéro 3**. Nous n'avions donc aucun contact direct avec nos interlocuteurs, contrairement à une vraie évaluation de l'expérience utilisateur.

Pour pallier ce manque, nous avons préparé un document avec nos personas, des scénarios à suivre, des questions précises sur les points visualisés par chaque scénario, des questions plus générales sur la totalité de notre site et enfin une partie "retours complémentaires" qui permettrait à l'autre équipe de faire des remarques qu'ils n'auraient pas pu exprimer dans leurs réponses aux questions.

2. Analyse des résultats obtenus

Les retours de l'autre équipe furent nombreux et complets, avec, pour certains, des captures d'écran permettant de les illustrer. Nous avons réussi à les regrouper en **une quinzaine de points sur lesquels nous devions travailler** afin d'améliorer notre projet. Certains étaient plus compliqués à réaliser que d'autres, et le temps restant étant limité, nous avions à choisir sur quels points travailler et quels points laisser de côté avant le rendu final.

Par conséquent, les points "**ajouter une justification aux réponses**" et "**charger des images différentes en fonction du contraste**" furent abandonnées car elles nous demandaient de changer notre architecture des deux côtés du projet. En outre, le second point portait sur un problème visuel. Or, le handicap principal visé par notre projet étant le handicap mémoriel, l'abandon de ce point fut une évidence pour nous (même si, on le rappelle, ce point est intéressant et devrait être implémenté dans une version future de notre projet).

3. Réalisations des points d'amélioration

a. Image d'une question

Les premiers points que nous avons réalisés étaient potentiellement les plus compliqués, dans la mesure où ils nous demandaient une certaine réflexion quant à la meilleure solution. L'exemple le plus probant est "**que faire de l'image correspondant à une question ?".**

En effet, dans notre maquette, nous avions prévu la possibilité d'associer une image à la question en plus des images associées aux réponses. Néanmoins, en cours de développement, nous nous sommes rendu compte que nous n'avions pas assez de place à l'écran pour afficher toutes les images proprement. Nous avons alors conservé seulement les images associées aux réponses.

Cependant, dans la partie administration, et plus précisément, dans la page de création de questions, cette option était toujours présente. Cela pouvait confondre les utilisateurs du site. Pour cette raison, **nous avons finalement laissé l'option d'associer une image à une question et nous l'avons affiché en dessous de l'énoncé de la question**. En effet, même si un défilement est nécessaire lorsque tous les éléments ont une image, cela permet aux accueillis qui ont des problèmes de mémoire de mieux comprendre la question, si les aides médico-psychologiques en ressentent le besoin.

b. Débogage des animations

Un autre exemple de ces points compliqués à réaliser est le **débogage des animations**. En effet, certains comportements des animations n'étaient pas souhaités. Par exemple, les réponses erronées ayant disparues après avoir été sélectionnées, réapparaissaient après la sélection de la bonne réponse. Dans la même optique, lorsque l'on sélectionnait une réponse trop rapidement, il était possible qu'une des questions soit sautée. Nous avons donc passé du temps à **corriger les timings et les conditions de déclenchement des animations** pour résoudre ce problème.

c. Correction de certains détails

Ensuite, nous avons identifié des points plus simples à réaliser comme “**ajouter des placeholders**”, “**renommer les variables des handicaps visuels**”, “**modifier le bouton de navigation pour qu’il soit plus explicite**” et “**valider les champs lors de la création d’un accueilli**”.

Ces derniers relevaient plus d’oublis au cours du développement et, au vu de leur facilité, furent réalisés quasiment instantanément. Ainsi, nos différents profils d’accessibilité ont été renommés “agrandissement” et “contraste élevé” afin de décrire correctement les améliorations d’affichage qu’ils engendrent.

Aussi, afin de pouvoir agrandir l’affichage des images des réponses lorsque l’on joue un quiz nous avons choisi d’adapter le container principal du site pour prendre plus de place sur l’écran lors du lancement d’un quiz et donc pouvoir par la suite améliorer encore la visibilité des réponses à chaque question.

d. Ajouts de fonctionnalités

Finalement, nous avons les points visant l’ajout de fonctionnalités dans notre site. Ceux-ci étaient “**ajouter du texte explicitant la disparition des réponses erronées sélectionnées**” et “**ajouter des données dans la visualisation des résultats**”. Nous avons choisi de réaliser ces deux points, d’une part car ils nous semblaient importants, et d’autre part car il semblait possible de les réaliser dans le laps de temps que nous avions.

Pour le premier point, nous avons simplement **ajouté un texte remplaçant la réponse erronée sélectionnée** après qu’elle ait disparue. Pour le second, nous avons **ajouté une option permettant de regrouper les résultats par quiz**, permettant de visualiser quels quiz posent vraiment un problème.

Cependant, nous n’avons pas pu réaliser toutes les idées que nous avons eu pour ces deux points. Par exemple pour le premier, nous avons pensé à ajouter une animation faisant disparaître le texte après un certain temps afin de ne pas laisser l’échec de l'accueilli sous son nez.

Pour le second point, nous avons eu l’idée d’ajouter une page permettant de voir la progression dans le temps de la réussite d’un accueilli à répondre aux questions d’un quiz, avec un graphe et la question sur laquelle il s’est le plus trompé. Cela permettant ainsi de faciliter le travail des aides médicales qui n’auraient pas à chercher à la main quelles questions posent le plus problème. Ces idées font donc partie des améliorations du site à réaliser pendant une potentielle suite du projet.

IV. Conclusion perspective

Pour conclure, nous avons trouvé ce **projet très intéressant**, avec une **problématique initiale importante et moderne**. Nous avons essayé d'y répondre de notre mieux. Dans cette optique, nous avons toujours gardé en tête nos personas qui nous permettaient de bien identifier les besoins de nos utilisateurs.

De plus, ce projet a été l'occasion de **découvrir de nouvelles technologies**. En effet, pour la plupart des membres de l'équipe, il s'agissait d'une première expérience avec Node et Angular que nous jugeons très instructive. Nous avons également pu **découvrir de nouvelles architectures** de code permettant de faciliter le développement du projet. Enfin, nous avons pu **assimiler de nouveaux concepts** tels que les promesses et les observables en JavaScript.

Finalement, nous sommes assez fiers de notre rendu ! Cependant, nous gardons à l'esprit de **nombreuses pistes d'amélioration pour ce projet**. En effet, l'association des quiz aux différents accueillis existe du côté serveur, mais est absente du côté client. Cette fonctionnalité est très utile puisqu'elle permettrait de concevoir des quiz spécifiques pour certains accueilli et donc de mieux les retrouver ultérieurement. De plus, comme nous l'avons dit plus haut, nous aurions voulu afficher les résultats des accueillis de manière plus visuelle pour simplifier le travail des aides médico-psychologiques. Enfin, nous pourrions stocker la question actuelle dans la session pour ne pas perdre la progression de l'accueilli en cas de rafraîchissement accidentel de la page.

Merci pour votre attention,

Equipe Redemption ❤️

V. Répartition des tâches

1. Répartition initiale des tâches

Dans un premier temps, nous avons décidé de nous répartir les tâches en deux groupes de deux étudiants : un groupe pour le **frontend** et un groupe pour le **backend**. Cette répartition était destinée à évoluée puisque tous les membres devaient assimiler les différentes connaissances du projet.

- Frontend :
  - **Valentin** : Jouer un quiz.
  - **Martin** : Sélectionner un quiz.
- Backend :
  - **Joao** : Serveur (routes et contrôleurs).
  - **Olivier** : Base de données (modèles et migrations).

2. Répartition finale des tâches

Dans un second temps, comme prévu initialement, la répartition des tâches a quelque peu évolué. En effet, le développement du serveur arrivant à sa fin, l'effort s'est concentré sur le **frontend** du projet.

- Frontend :
  - **Valentin** : Visualiser les résultats, jouer un quiz, sélectionner un quiz, et créer ou modifier des ressources (images, thèmes, quiz, questions, réponses et accueillis).
  - **Joao** : Refonte visuelle et fonctionnelle de certains composants pour la création ou la modification de ressources. Il s'agissait de bien structurer le client et d'offrir une ergonomie optimale lors de l'utilisation du site.
  - **Martin** : Gérer les problèmes visuels et réglage des styles du client.
- Backend :
  - **Joao** : Serveur, base de données et docker.
  - **Olivier** : Aide pour le serveur.

VI. Bibliographie

1. Documentation

- Angular
- Pagination
- Search filter
- Font awesome
- Bootstrap
- Bootstrap modal
- Sequelize
- Express

2. Articles et vidéos

- Model-View-Controller (MVC)
- Build a scalable Node.js REST API using Express.js
- Academind - Creating a REST API with Node.js