Rapport PS6

handicap : **Alzheimer**

Rachid EL ADLANI - Julien KAPLAN - Abdelouhab BELKHIRI - Benjamin COUX

OK GOOGLE

Université Côte d'Azur - Polytech Nice – SI3-PS6

Table des matières

Partie 1 - Personas et scénarios

1.a) Personas

Persona 1: Aide-soignante

Persona 1

- Marie
- 28 ans
- Créative et dynamique
- Elle est sans cesse en quête de nouvelles idées pour les proposer à ses patients
- Elle a besoin d'un site simple où elle peut avoir la main sur ce qu'elle fait

Scenario 1

Scenario 2

Persona 2: Aide-soignante

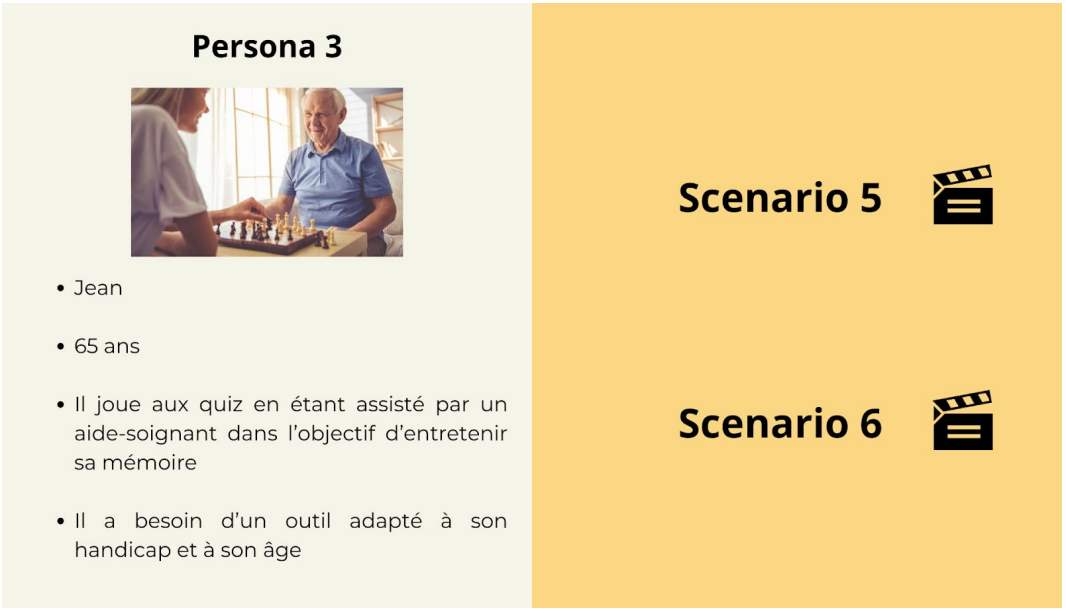
Persona 2

- Nathalie
- 25 ans
- S'occupe d'entretenir la mémoire des patients afin qu'ils préservent leur autonomie en dehors du centre
- Nécessite un outil de suivi des patients

Scenario 3

Scenario 4

Persona 3: Accueilli



1.b) Scénarios

Scénario 1 - Aide-soignante (Marie)

Une aide-soignante ouvre le lien du site internet, elle se connecte à son espace “administrateur”. Elle arrive sur une page contenant plusieurs choix/fonctionnalités, elle crée le quiz en ajoutant un thème et un titre. Elle crée ensuite des questions en insérant des images parfois dans les réponses et parfois dans les énoncés selon le modèle de question qu'elle choisit.

Scénario 2 - Aide-soignante (Marie)

Un aide-soignant a remarqué qu’un quiz était trop difficile pour l’un de ses patients, il se connecte à son compte et choisit de modifier le quiz, il supprime des questions, en rajoute, en modifie quelques unes. Il change également l’indice qu’il juge pas très efficace en vue des statistiques. Il remplace certaines images car il s’est rendu compte qu’elles ne sont pas très visibles.

Scénario 3 - Aide-soignante (Nathalie)

Une aide-soignante a reçu la liste des nouveaux arrivants et des départs de la semaine, elle se connecte à son compte administrateur, elle accède au menu, choisi d’ajouter de nouveaux profils en entrant le nom, le prénom et une rapide description de la personne afin d’avoir plus d’informations. Elle revient au menu et choisit de supprimer les profils des patients qui ne sont plus clients au centre.

Scénario 4 - Aide-soignante (Nathalie)

Une aide-soignante arrive au centre en début de semaine, elle se connecte sur la partie administrateur du site internet, et accède à la page “statistique” , elle établit les quiz à réaliser durant la semaine pour ses patients en se basant sur les résultats de chaque personne de la semaine précédente.

Scénario 5 - Accueilli (Jean)

Un habitué du centre arrive, il souhaite jouer à un quiz, il s’installe sur une chaise, la tablette est allumée, il se connecte en tant qu’accueilli, il ne trouve pas son profil utilisateur, il choisit donc le profil par défaut, il choisit son thème et commence à répondre aux questions. Une fois le quiz terminé il dit qu’il a bien aimé et il relance le même quiz.

Scénario 6 - Accueilli (Jean)

Un patient du centre est installé devant une tablette, sur le site internet du quiz. L’aide soignant sélectionne le profil du patient, il choisit avec lui un thème parmis ceux affichés, puis il lui sélectionne le premier quiz. Le patient effectue les premières questions puis il se rend compte que ça le fatigue de devoir se concentrer sur des petites lettres, il fait donc la remarque à l’aide-soignant. Celui-ci va dans les options et modifie les paramètres de luminosité et de taille de police jusqu’à ce que le patient valide.

Partie 2 - Architecture Client/Serveur

2.a) Partie architecture projet

Nous avons essayé d’organiser notre projet en packages représentatifs de leur contenu, nous pouvons trouver dans le dossier **frontend** 4 packages principaux (***cf: figure1***)

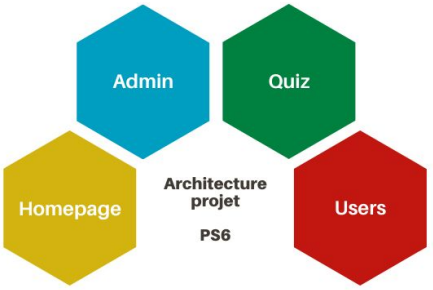


figure1: architecture_projet

Nous sommes ici dans le package Admin (***cf: figure2***), qui fait référence au personnel santé qui sera en mesure d'effectuer des actions de création, de modification ou autre. Donc dans ce package il y a différentes parties.

Premièrement nous avons la partie création qui contient différents composants. Les composants sont une représentation graphique d'un bout d'interface qui peut être personnalisable et donc composé de un ou plusieurs éléments tout en étant réutilisable.

Nous avons dans cette partie "**création**" tous les composants permettant la création des différents éléments qui les composent : **create_theme** : pour les thèmes, **create_quiz** : pour les quiz, **create_user** pour les utilisateurs et **question_form** pour tous ce qui concerne la création de questions/réponses.

Ensuite la partie "**gestion**" avec des composants qui permettent l'édition des éléments présentés dans la partie création.

Et enfin nous avons la partie "**accueil**" qui représente l'interface graphique pour la page d'accueil de l'aide soignante.

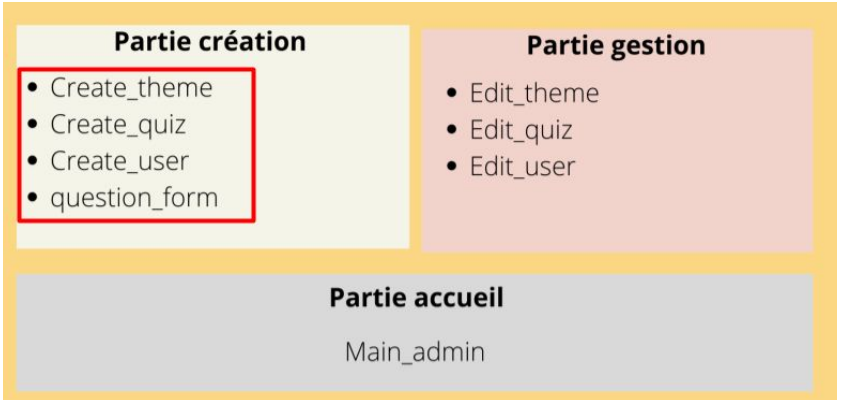


figure2: package_admin

Nous avons le package Quiz (***cf: figure3***) qui contient les définitions des éléments tournant autour du Quiz donc Question Réponse Thème Quiz.

On a à chaque fois le composant qui définit l'élément et un composant -**list** qui permet l'affichage sous forme de liste de cet élément.

Nous avons aussi **play-quiz** dans la partie quiz qui sert à jouer le quiz il s'agit de l'interface graphique qui dispose la question ainsi que les réponses mais également tout ce qui est visible à l'accueilli lorsqu'il joue son quiz.

figure3: package_quiz

Pour finir, notre package **home-page** à la racine contient le composant de la première page du site et le package **User** contient tout ce qui concerne la définition des utilisateurs notamment les statistiques des utilisateurs. Même s'ils concernent les administrateurs, nous trouvons ça plus intuitif de devoir aller dans le dossier **User** plutôt que le dossier **Admin** pour trouver les statistiques des User.

2.b) Service

Nous arrivons à la partie **service**. Un service est un ensemble de fonctions partagées par l'ensemble des composants. Nous avons décidé d'utiliser différents **services** dans notre projet car nous avons différentes fonctionnalités (***cf: figure4***). De plus, l'utilisation de différents services nous a permis d'avoir un code plus compréhensible et plus facilement maintenable.

Premièrement nous avons les services : Thème / Quiz et User qui ont des fonctionnalités assez similaires qui sont **l'ajout**, la **modification** et la **suppression** des éléments qui les composent (**thème/quiz/user**). Ils ont également des méthodes qui permettent de récupérer les données sous forme de listes ou de simples objets (***liste des utilisateurs / listes de quiz***).

Deuxièmement, le **result service** qui permet la gestion de certaines fonctionnalités du jeu telles que la possibilité d'avoir des indices pour la gestion de problèmes de mémoire. L'accueilli pourra en cas de blocage sur la question, utiliser une aide qui va lui fournir un indice texte ou sonore pour une expérience de jeu la plus fluide et la moins frustrante possible.

Le **result service** fournit également la fonctionnalité d'une possibilité de retour à la question précédente, où le personnel santé peut utiliser un bouton pour faire recommencer la question à l'accueilli s'il estime que l'accueilli n'a pas bien compris la question ou alors pour voir si l'accueilli à bien assimilé la réponse toujours pour un soucis de maintien de la capacité de mémoire des patients. De plus, ce service va stocker les résultats des accueillis dès lors qu'ils ont fini de jouer un quiz.

Ensuite, le **stat service** rend possible la liaison entre les résultats et l'accueilli qui a joué ce quiz avec la récupération au préalable.

Nous terminerons par le **router-history service** qui va rendre possible un historique de navigation. Nous l'avons mis en place pour pallier au problème de mémoire des accueillis, en effet cela va leur permettre de pouvoir visualiser la page dans laquelle ils vont atterrir s'ils clique sur le bouton retour. Nous avons également fournis cette fonctionnalité aux aides soignantes pour avoir une homogénéité dans notre application.

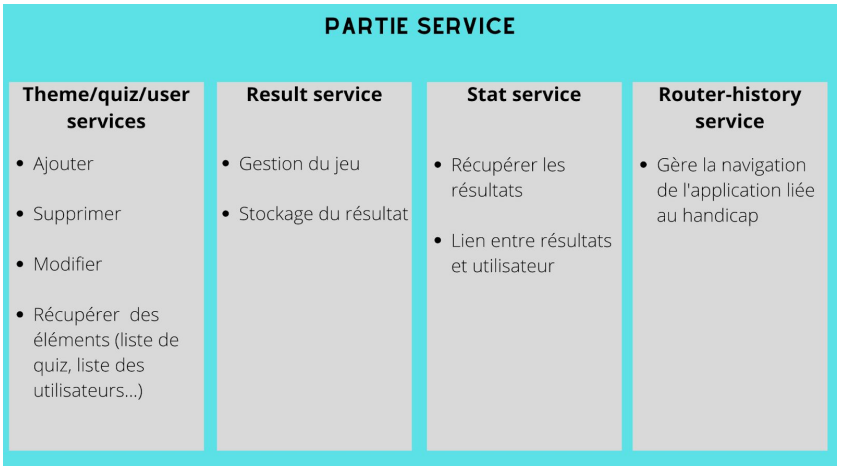


figure4: services

Le fonctionnement :

1. On va gérer dans ce service l'enchaînement du quiz qui va fonctionner avec un pointeur de question (***cf: figure5-étape1***)

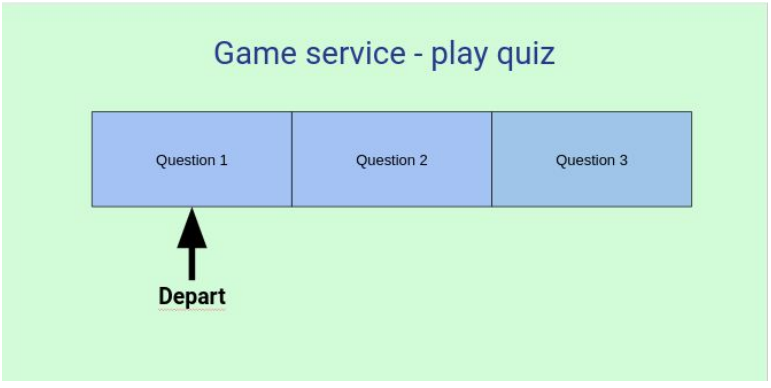


figure5-étape1: Game-service

On va avoir un pointeur qui va s'incrémenter à chaque bonne réponse, on pourra voir dans le composant **playquiz** si c'est une bonne réponse (***cf*: *figure5-étape2***).

On sauvegarde toutes les réponses que l'accueilli saisit pendant le jeu dans un tableau de réponse (***cf*: *figure5-étape2***)

On va avoir un observable **questionSelected** dans le service qui se mettra à jour à chaque bonne réponse et qui donnera la question suivante donc cela correspond a l'indice + 1 dans le tableau de question du quiz.

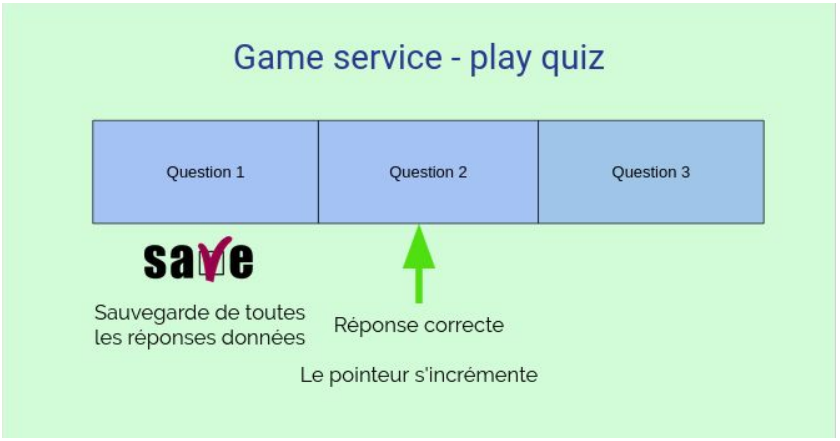


figure5-étape2: Game-service

Le pointeur se décrémente si la personne décide de revenir à la question précédente mais on garde toujours les réponses données à la **question 1** car cela a son utilité dans les statistiques (***cf*: *figure5-étape3)***. Lors du retour à la **question 2**, on aura gardé le même contexte qu'avant le retour à la question précédente afin qu'il puisse reprendre normalement.

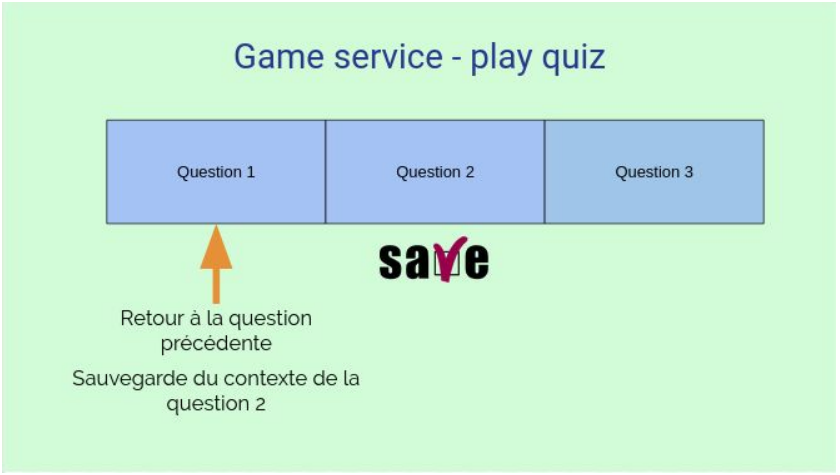


figure5-étape3: Game-service

A la fin du jeu, on va rassembler toutes les données du jeu dans un objet ***Result*** et on va faire une requête ***HTTP*** de type ***POST*** vers le serveur avec l'objet *Result.* Puis, dans le serveur on va traiter les réponses pour avoir le nombre de bonnes et mauvaises réponses ensuite, on stock les infos dans le fichier ***JSON Result.mock***.

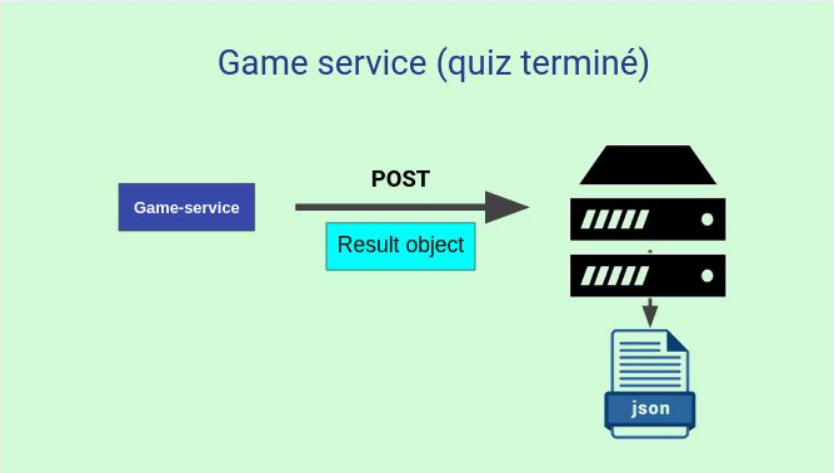


figure5-bis: ***Game-service***

2.c) Route

Concernant la partie route, on a une **API *REST*** qui est un style d'architecture permettant de centraliser des services partagés.

On va avoir le module **Result** qui va avoir plusieurs ressources dont :

- la ressource **get** avec en paramètre l'identifiant de l'utilisateur qui va récupérer les résultats de quiz d'un utilisateur en particulier, on a dû créer une fonction ***getByUserId()*** dans le fichier ***base-model.js***, qui à été fournie avec l'architecture de base côté back end.
- la ressource **get** avec en paramètre l'identifiant de l'objet **Result** qui va récupérer un résultat en particulier
- la ressource **post** qui va ajouter un objet **Result**

On va avoir le package **Users** qui contient des ressources qui font des opérations de base telles que la création, la lecture, la modification et la suppression de données, ce sont les opérations **CRUD** qu'on utilise le plus souvent.

On va avoir aussi des modules qui seront imbriqués en commençant par thèmes (***cf: figure6***), qui vont tous avoir leurs opérations **CRUD**, dans les packages allant du module **Theme** jusqu'au module **Question**, on va aussi utiliser la redirection de ressource avec **router.use()** par exemple lorsqu'il y a :**themeid/quizzes** en paramètre de l'url, il sera redirigé vers les ressources de **Quiz**.

Pour tous les modules imbriqués dans le module *thème*, ils devront lier les paramètres de leurs url avec ceux des modules parents avec l'attribut **mergeParams** qui sera mis à **True**.

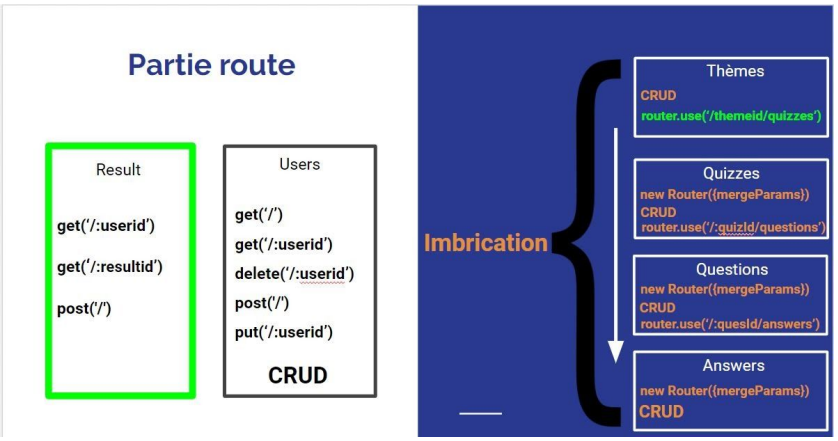


***figure6:* Partie route**

2.d) Model

Pour la partie **model** notre site comporte des mocks qui représentent la principale base de données présentes dans notre site, chaque **model** (***cf: figure7***) représentent les caractéristiques de chaque objet.

Tout d'abord nous avons la ressource **User** qui va contenir tous les champs utiles pour un utilisateur sur notre site. Le champ **id** est généré automatiquement dans la partie back end, les autres champs sont rentrés lors de la création du **User** via le formulaire présent sur le site web.

Un objet **Theme** contient aussi un nom, une ***Url*** pour y stocker une ou plusieurs images qui sont saisies lors de la création de **Theme** avec le formulaire. Un thème contient aussi un tableau de quiz qui représente tous les quiz reliés à ce thème.

Le model **quiz** contient lui aussi un **id** et tous les champs saisis par un utilisateur lors de sa création comme un nom ou encore une **Url** pour une image et un **id** de thème pour savoir à quel thème le quiz appartient, l'attribut **creationDate** est lui saisi automatiquement lors de la création. Un quiz contient aussi un tableau de questions qui sont toutes les questions appartenant au quiz.

Le modèle **question** contient aussi un **id** et un **label** correspondant à l'intitulé de la question, un champ **indice** qui correspond à la phrase qui sera affiché comme indice. Une question peut aussi avoir un lien vers un fichier audio comme indice ce qui correspond au champ **sonUrl**. Ce modèle contient aussi un booléen **aideUtilise** qui sera lui utilisé dans la ressource **Result**. De plus, une question possède un tableau de réponses (**answers)** qui représente toutes les réponses possibles à la question.

Le modèle **Answer** va contenir un **id**, une **Url** d'image dans le cas où le modèle **question** correspond à celui où les images sont dans les réponses, et un booléen **isCorrect** qui vaut vrai ou faux et qui sera lui aussi utilisé dans le modèle **Result**.

Enfin le modèle **Result** correspond au résultat d'un quiz par un utilisateur. Un objet **result** va contenir un **id** généré automatiquement, un **userid** et un objet **quiz** pour savoir quel utilisateur a réalisé quel quiz. Les champs **nbErreurs**, **nbCorrect** et **nbAides** sont déduits des booléens **isCorrect** et **aideUilise** des ressources **Answer** et **Question**.

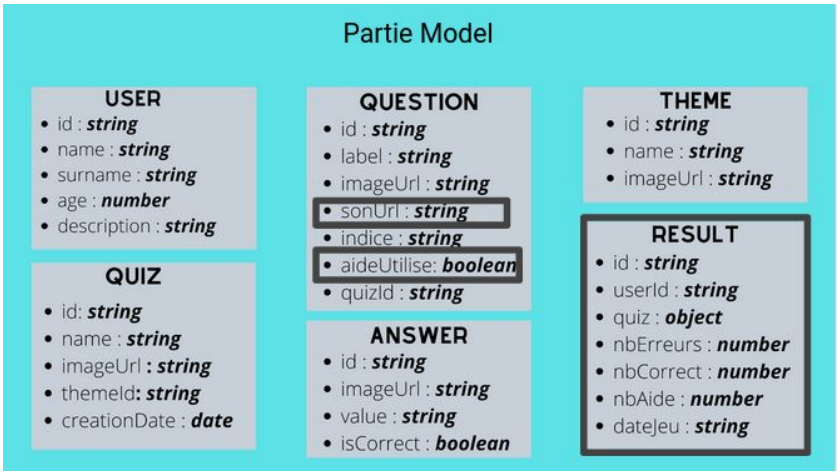


***figure7:* Partie route**

Partie 3 - Évaluation croisée

Nous avons choisi pour l'évaluation croisée la première méthode proposée par le cours, c'est-à-dire un testeur qui réalise les scénarios en pensant à haute voix et des observateurs qui prennent des notes. Nous étions les 2 équipes (la Team-js et nous) dans une conversation vocal, un membre de leur équipe réalisait nos scénarios pendant que les 3 autres membres donnaient leur avis, et nous prenions en note leurs retours. Puis nous prenions un temps pour leur poser nos questions préparées.

Globalement leurs retours étaient positifs, ils trouvaient en quelques secondes les fonctionnalités qu'ils recherchaient, ils n'ont jamais été bloqués dans une page sans savoir quoi faire, et ils nous ont dit que notre site était intuitif et que nos idées étaient intéressantes dans le traitement de l'handicap d'Alzheimer. Il ont particulièrement apprécié nos animations entre les questions du quiz (celles qui indiquent que la question suivante se lance dans 5 secondes)

Grâce à cet entretien nous avons pu repérer de petites fautes comme des fautes d'orthographe, d'ergonomie, ou du texte écrit en noir sur noir (le commit qui corrigeait les fautes d'orthographe avait été écrasé par un autre commit). Mais aussi il nous a permis d'établir une liste de problèmes pertinents à résoudre pour notre site:

- Comprendre que l'on doit scroller vers le bas quand on crée un thème. Nous avons réglé ce problème en réduisant la taille de l'image du thème pour permettre de voir qu'il y a les options de création de quiz (***cf: figure5***).

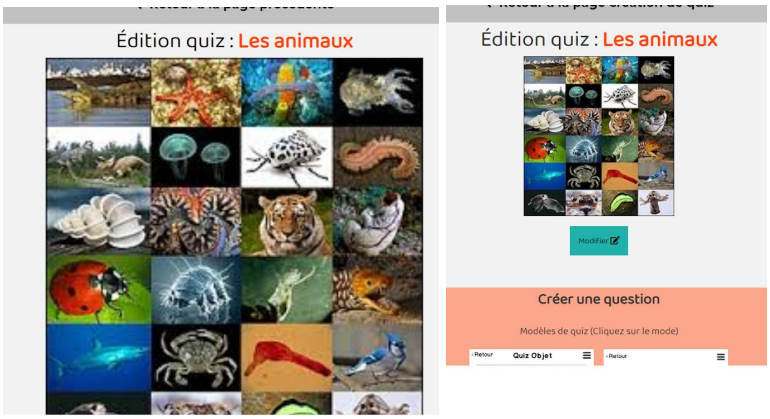


figure 5: edition-quiz

- Le champ de recherche de thèmes/quiz était trop accessible pour les accueillis sachant que sur tablette cela fait apparaître le clavier qui gène la visualisation de la page. Pour empêcher les accueillis d'ouvrir le clavier sans le faire exprès, nous avons rajouté une icône qui sera cliquable (***cf: figure 6***), lorsque cet événement est déclenché, ceci affichera le champ, sans que le curseur du clavier soit ciblé (ou focus) dessus. Donc un seul mauvais clic n'affichera pas le clavier sur tablette.

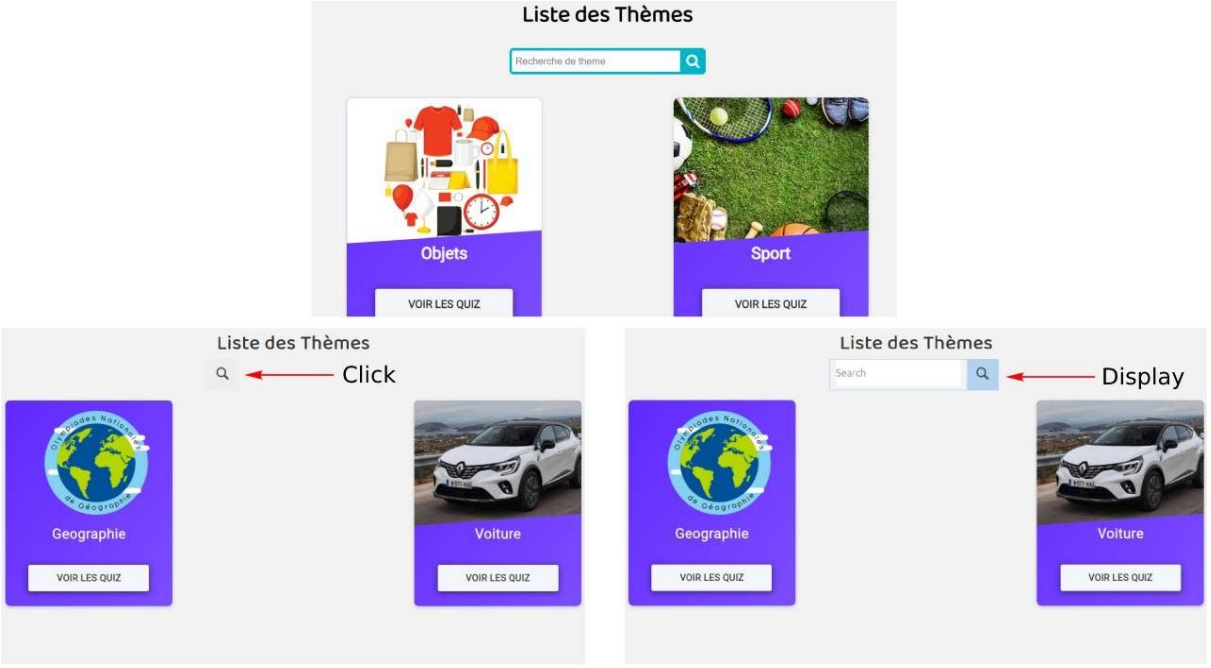


figure 6: liste des thèmes

- Lorsqu'un personnel soignant va dans "gestion des quiz" alors que aucun thème n'a été créé, il y avait simplement un texte indiquant l'absence de quiz, nos testeurs nous ont proposé de mettre un bouton de redirection vers la création de thème.

- Afficher un thème sans quiz dans la partie accueilli n'est pas nécessaire. Nous avons donc retiré l'affichage des thèmes sans quiz et par la même occasion, l'affichage des quiz sans question dans la partie accueilli.

- Sur les quiz, le bouton d'aide était affiché alors qu'il n'y avait pas d'indice entré. Ce qui n'a pas de sens, et peut gêner nos utilisateurs. Désormais nous n'affichons le bouton d'aide que lorsqu'il contient un indice.

Il y a tout de même une remarque tirée de cette analyse croisée dont nous n'avons pas apporté de solution. Lorsqu'un utilisateur modifie la taille de la police d'écriture dans un quiz et qu'il finit son quiz, cette modification n'est pas sauvegardée. Donc il devra refaire ses paramètres en lançant un autre quiz. Nous sommes d'accord que ça serait mieux de sauvegarder cette information pour chaque profil mais nous n'allons pas l'implémenter car cela demanderait beaucoup de modifications alors que ce n'est pas un élément indispensable au bon fonctionnement du site.

Cette expérience d'évaluation croisée nous a permis de recueillir des informations objectives d'autres personnes ayant été confrontées aux mêmes problèmes que nous. Notre équipe a été constamment baignée dans ce projet et le fait d'avoir l'avis de personnes extérieures permet de voir certains points auxquels nous sommes passés à côté. Cela nous a aussi permis de nous réconforter sur certains points, comme par exemple, si notre gestion de la maladie d'Alzheimer est efficace, l'autre groupe a validé nos différents choix et implémentations concernant cela.

Partie 4 - Conclusion et perspective

4.a) Résolution technique

Lors de la soutenance technique nous avions présenté quelques problèmes techniques notamment la mauvaise ouverture du quiz ou du thème quand on change trop rapidement de page, notre solution était de **rafraîchir** la page. Nous avions identifier un problème concernant certaines **requêtes** qui étaient dépendantes d'autres requêtes. Donc, dès lors qu'une requête prenait beaucoup de temps d'exécution l'autre ne pouvait se lancer. Nous avons donc réparer ce problème en supprimant la dépendance de ces requêtes.

.

4.b) Conclusion et perspective

Ce projet nous a permis de confirmer (et d'acquérir pour certains) nos compétences dans les technologies du web avec **Angular** et les langages comme **html typescript.** Nous avons également pu voir la disposition des éléments, l'ergonomie de l'application grâce au **CSS** qui nous a permis de placer correctement les éléments pour répondre aux besoins des accueillis et des utilisateurs de notre application. De plus, notre application respecte ce que l'on a appelé le **responsive design**, où notre application et la manière dont sont disposés les éléments sont adaptés aux écrans (ordinateur ou tablette majoritairement).

Avec des projets en équipe, nous avons développé des compétences de communication et nous avons appris à nous organiser de manière différente notamment avec cette crise sanitaire du **covid-19**, où nous avons utilisé des outils tels que **slack** ou **discord** pour se réunir et se mettre d'accord sur certains points.

Nous avons eu, après l'évaluation croisée, des remarques sur certains éléments que l'on pourrait implémenter tels que la sauvegarde des paramètres : zoom ou contraste inversé une fois que l'utilisateur quitte son quiz. Nous pensons que c'est une bonne idée d'implémentation sur la suite du projet.

Ce fut notre premier projet conduit pour de vrais utilisateurs, cela nous a apporté une motivation supplémentaire, nous avions envie de satisfaire au mieux nos clients. Nous regrettons tout de même le manque d'interaction avec eux, car nous avons dû prendre certains choix où leur avis aurait pû nous guider.

Au final nous sommes assez fiers de ce que nous avons réalisé, nous trouvons notre site ergonomique, répondant aux besoins, où chaque élément a été pensé pour être le plus intuitif possible.

Annexe

1) Répartition des tâches

Afin de nous répartir les tâches nous avons crée un "**google sheets**" où l'on écrivait toutes les implémentations à réaliser, lorsque quelqu'un décidait de faire quelque chose il indiquait son nom à côté de la tâche et lorsqu'il avait fini il barrait la tâche. Cela nous a permis de faire participer tout le monde et d'avoir une vue claire sur ce qu'il restait à faire.

Nous avions également un répertoire **github** qui nous a permis de centraliser toutes les ressources afin que chaque membre du groupe puisse accéder au code. Nous avons également utilisé **Discord** et **Slack** pour se réunir et définir les différentes tâches à réaliser, mais également pour s'accorder sur les différentes idées et stratégies de développement.

2) Bibliographie

Documentation Angular

https://angular.io/guide/universal

StackOverFlow pour des recherches ciblées

https://stackoverflow.com/

Documentation de TypeScript

https://www.typescriptlang.org/

Github : la base

https://github.com/

Tutoriels **CSS**

https://flexboxfroggy.com/#fr

Express.js : Enregistrement et récupération du son

https://expressjs.com/

Module **NodeJs**

https://www.npmjs.com/