Rapport PS6

LOSCIALE Vivian

AYOUB Hamza

MARTIN Thomas

KHAYATI Hiba

Nom d’équipe : TooTiredToWork

Table des matières

Les Personas et leurs scénarios 2

Dorothéa 2

Scénario 2

Léontine 2

Scénario 4

Baudoin 4

Scénario 4

Présentation de l’architecture Client Serveur 6

Front 6

Page de configuration 6

Page ThemeList 7

Back 9

Présentation de l’évaluation croisée et analyse des résultats obtenus 11

Conclusion et perspective 13

Annexes 14

Répartition des tâches dans l’équipe 15

Vidéo de démonstration finale 16

# Les Personas et leurs scénarios

Le but de ce projet est de créer une application de quiz pour les personnes âgées étant atteint de troubles spécifiques. Dans notre cas, nous nous sommes penchés sur la thématique des personnes âgées ayant des problèmes visuels.

Pour mener ce projet à bien, nous avons utilisé des personas ayant chacun un scénario différent. Nous avons à notre actif 3 personas différents. Une aide-soignante, et deux personnes âgées ayant des problèmes de vue différents.

## Dorothéa

Dorothéa a 38 ans, elle est aide-soignante et s’occupe des personnes âgées dans le centre. Elle est en charge de créer les quiz et/ou de les maintenir. Comme elle est très occupée, elle aimerait une création simple et rapide du quizz. Elle aimerait aussi pouvoir modifier ou supprimer un quizz si celui-ci est erroné ou n’est plus utile.

### Scénario

Je m’occupe de la création des quizz.

Pour cela,

* Je vais dans le mode création afin de gérer les quizz.
* Je choisis d’éditer le thème qui contient le quiz en sélectionnant les petits rouages qui apparaissent dans la case du thème.

Je veux créer un nouveau questionnaire.

Pour cela,

* Je défile dans le carrousel afin d’arriver au bouton + pour ajouter un quiz.
* Je clique dessus.
* Je donne un nom au futur quizz.
* Je clique sur créer.
* Dans la page d’édition du quizz que j’ai créé, je crée une nouvelle question en appuyant sur le bouton +.
* Je remplis la box dédiée à la question puis je remplis les box destinées aux réponses.
* Je confirme ma saisie et je crée d’autres questions.
* Je pense que le niveau du quizz est difficile, je mets donc le niveau de difficulté à 3.

## Léontine

Léontine a 82 ans, elle a un glaucome[[1]](#footnote-2). Elle a l’habitude d’utiliser l’outil informatique pour faire ses recherches personnelles. Elle veut mettre à rude épreuve ses connaissances et veut connaître ses résultats pour chaque quizz. Elle voudrait pouvoir lire les questions sans à constamment bouger la tête. Elle aime créer des quiz pour ses amies grâce aux connaissances qu’elle a.

### Scénario

* Je choisis le thème « Disney » en appuyant sur la case appropriée.
* Avant de choisir un quizz, j’appuie sur le rouage en haut à droite pour gérer les paramètres.
* J’augmente la taille des marges aux extrémités pour pouvoir voir l’intégralité de la question dans mon champ de vision.
* J’appuie sur confirmer et je choisis le quizz « La petite Sirène ».
* À la fin du quizz, je regarde mon taux de réussite à ce quizz.
* Je choisis de faire une relecture et de voir où étaient mes erreurs en cliquant sur relecture.
* Quand j’ai fini ma relecture, j’appuie sur le bouton quitter.

J’ai envie de rajouter un quiz sur « La reine des neiges ».

Pour cela,

* J’appuie sur le bouton mode création.
* J’édite le thème qui correspond à mon quiz, en l’occurrence, Disney.
* À l’aide du carrousel, je fais défiler les pages jusqu’à trouver une case avec un « + » et j’appuie dessus.
* Je rentre le nom du nouveau quiz et je clique sur créer.
* Je crée une nouvelle question en appuyant sur la case « + ».
* Je remplis les différents champs qui me sont proposés et je confirme la création de question.
* Après avoir créé différentes questions, j’appuie sur la flèche en haut à gauche jusqu’à revenir au menu principal.

## Baudoin

Baudoin a 88 ans, ses petits-enfants lui ont appris à utiliser les nouvelles technologies. Il aime faire des quizz pour se détendre, et apprendre de nouvelles choses sur des sujets qu’il ne connait pas forcément. Il a du mal à lire de près. Il possède aussi une cataracte[[2]](#footnote-3). Il voudrait donc pouvoir changer la police d’écriture ainsi que la couleur de fond en fonction de son état de fatigue.

### Scénario

* Au début de l’application, je configure la taille de police afin de pouvoir lire les questions.
* Je veux faire un quizz sur le « sport ». Je choisis donc ce thème.
* Je remarque que j’ai encore un peu de mal à lire, j’appuie sur l’icône rouage en haut à droite pour changer les paramètres.
* J’augmente encore la taille de police et je change la palette de couleur du site.
* Quand les paramètres me vont, j’appuie sur confirmer.
* Je sélectionne le quizz sur le « handball » et je réponds aux différentes questions.
* À la fin du quizz, je sélectionne quitter.

# Présentation de l’architecture Client Serveur

Ce site a été réalisé dans un cadre d’apprentissage. Aussi, nous avons eu des cours sous formes de travaux pratiques pour prendre en main et comprendre le fonctionnement de la technologie utilisé : Angular. Lors de ceux-ci, nous avons vu qu’un site web était séparé en deux partie, le front et le back.

Le front est la partie de projet, du code, qui va être envoyée au navigateur web pour afficher la page internet. Le back, quant à lui, est la partie « serveur » du site. C’est elle qui contient les données et qui reçoit les requêtes des utilisateurs.

Tout au long de la réalisation du ce projet, nous avons tenu à garder cette séparation, si bien que nous avons deux dépôts git différents. Nous avons donc travaillé en parallèle sur ces deux parties de notre site, et nous allons présenter l’architecture en suivant la même logique.

## Front

La partie front est entièrement réalisé avec le framework Angular en langage Typescript. L’un des principaux éléments majeurs d’Angular est le Component. Ce dernier est simplement un élément graphique qui fonctionne grossièrement avec un système vu – contrôleur. La vue est représentée par une page html (et une feuille de style css) et le contrôleur est représenté par une classe Typescript.

Ces components servent donc à définir un composant graphique de notre site web, ils sont la base de ce dernier. Pour définir quand créer un component pour un élément graphique, nous avons décidé d’opérer de cette façon : chaque page du site est un component et chaque élément graphique nécessitant des données en entré en est un aussi.

Enfin, il nous faut aussi parler des services. Ce sont des classes Typescript, qui ne sont instanciées qu’une fois et qui sont données aux components du site. Ils permettent de partager des données et des méthodes entre plusieurs component. Ainsi, une opération réalisée par un component A sera visible sur un component B.

Pour présenter l’architecture de la partie Front de notre projet, nous allons donc présenter les principaux components et services.

### Page de configuration

La page de configuration est l’une des pages essentielles de notre site web. Elle est accessible à n’importe quel instant et permet la configuration de divers paramètres graphiques pour que l’utilisateur adapte le rendu à sa vision.

#### Configuration Component

Comme dit précédemment, chaque page nécessite un component et la page de configuration ne déroge pas à cette règle.

Le ConfigurationComponent n’a qu’un rôle, afficher des « sélecteurs » (slider, color picker…) et récupérer leurs valeurs. Il y a une petite particularité cependant dans son fonctionnement.

En haut de cette page se trouve un champ « exemple ». Lorsque l’utilisateur modifies les paramètres, ses choix sont répercutés sur un exemple situé dans le haut de la page. Par exemple si un utilisateur modifie la taille de la police seul la taille de la police de l’exemple va changer et non pas toute la page. Pour réaliser cela il suffit d’appliquer la configuration courante aux éléments de la page et d’appliquer les choix de l’utilisateur uniquement sur l’exemple (ces choix sont ensuite appliqués sur l’ensemble du site lorsqu’il valide ces changements)

#### Configuration Service

Le ConfigurationService est le service qui accompagne le ConfigurationComponent. Il est simple et a uniquement pour rôle de stocker la configuration courante.

Il est fourni à l’ensemble des components du site qui en ont besoin et qui peuvent donc y récupérer les valeurs de configuration pour pouvoir les appliquer.

### Page ThemeList

La page ThemeList est, comme son nom l’indique, la page qui liste les thèmes. C’est la page principale du site, c’est à partir d’elle que l’on peut rentrer dans le « mode création » et c’est bien sûr à partir d’elle que l’on va sélectionner le thème sur lequel on souhaite réaliser un quiz.

#### ThemeListComponent

Le ThemeListComponent, est un component de « liste ». Il y en a plusieurs dans l’architecture de notre projet et ont tous un fonctionnement globalement similaire.

Ce component a pour rôle principal, de permettre à l’utilisateur de choisir un thème parmi l’ensemble des thèmes disponibles. Il doit donc pouvoir afficher correctement cette liste et permettre à chacun des éléments la composant d’être « cliquable ».

Pour rendre les éléments cliquables, il nous fallait représenter chaque thème par un élément graphique qui prenait en entré le thème qu’il doit afficher, autrement dit, il nous fallait un component. Ce dernier n’est rien d’autre qu’une adaptation de ce que nous avons vu lors de nos différentes séances de travaux pratiques. Chaque page « liste » possède donc son « ItemComponent » qui est ici ThemeComponent.

Nous avons fait le choix de supprimer les scrollbar de notre site, nous avons veillé et fait de notre maximum pour qu’il n’y ait jamais besoin de faire défiler la page de manière verticale ou horizontale. Pour réaliser cela, nous avons eu recours à ce que l’on appelle un Carousel.

À l’instar des « ItemComponent », chaque page « liste » possède un carousel, il convient donc d’en parler.

#### CarouselComponent

Le CarouselComponent est assurément le component le plus complexe de notre projet et celui qui nous a demandé le plus de temps à réaliser.

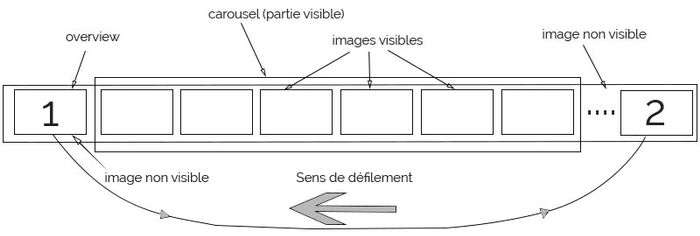


Figure 1 - Schéma représentant le fonctionnement d'un carousel

Un carousel est un composant « classique » du web. On le retrouve souvent pour la présentation d’images. Son fonctionnement est comparable à une fenêtre devant laquelle ont ferait défiler des éléments à la demande. Une partie des données seulement est visible.

Pour ne pas réinventer la roue, nous sommes parties en quête de trouver un carousel utilisable dans notre développement que l’on aurait jute à adapter rapidement. Hélas, il n’en était rien.

On trouve facilement de nombreux carousel utilisable pour des images et par extension pour une liste d’éléments uniques de même dimension. Une liste d’images de même taille, une liste de boutons de même taille etc. Dans notre cas, nous ne voulions pas afficher uniquement un item par page du carousel, cela ferait un trop grand nombre de pages et surtout une grosse perte de temps. Nous avions donc besoin de réaliser notre propre carousel et c’est ce que nous avons fait.

Le fonctionnement du CarouselComponent est assez complexe et il ne sera pas détaillé ici, seulement, ce qui le différencie de ce que l’on vient de voir, c’est qu’au lieu d’afficher une liste d’éléments il affiche en réalité une liste de liste d’éléments.

Le carousel est composé de pages qui chacune contient des items. Une page à la fois est affichée, et on peut donc naviguer à la manière classique d’un carousel entre les différentes pages.

De nombreuses problématiques sont alors soulevées par ce fonctionnement, comment déterminer le nombre d’éléments affichable par page sachant que leur taille dépend de leur contenu et des paramètres graphique ? Comment généraliser le carousel pour qu’il affiche les items avec le bon component ?

Nous avons bien sûr trouvé les réponses à ces questions puisque notre carousel est fonctionnel. Nous avons aussi ajouté une fonctionnalité pour le mode création, il s’agit d’un bouton « plus » qui s’affiche en fin de liste (dernier élément de la dernière page) et qui va permettre d’ajouter un élément à la liste affichée lorsque l’on clique dessus. (Techniquement, il ne va pas ajouter mais va plutôt effectuer une action qui est passée en entrée au carousel et c’est d’ailleurs la présence de cette action qui détermine l’affichage du bouton plus).

### Page de quiz

Parce que notre site est un site de quiz, il nous faut évidemment une page pour en faire un.

#### Quiz Game Component

Le QuizGameComponent est le component de la page quizgame. Il a pour rôle d’afficher, une par une, les différentes questions d’un quiz donné avec leurs réponses. Il va donc afficher dans l’ordre les questions, et, chaque fois que l’utilisateur réponds à une question, afficher la suivante.

Les différentes réponses sont disposées aléatoirement.

#### Game Service

Le GameService est le service qui accompagne le QuizGameComponent. Là encore, son rôle est très simple, il doit uniquement conserver la partie courante (le quiz et les réponses de l’utilisateurs). Nous avons utilisé ce service pour ne pas avoir à donner la partie dans l’url de la page de résultat. C’est donc pour cette dernière que ce service existe.

### Page de résultat

Elle est la page finale du quiz. Elle affiche le score à l’utilisateur et lui offre la possibilité de rejoindre la page de relecture.

### Result Component

Il n’y pas beaucoup de choses à dire sur ce Component qui n’a pas déjà été dit. C’est un composant simple qui comme dit précédemment fonctionne avec le GameService duquel il va récupérer les informations de la partie courante pour pouvoir en afficher le score.

### Autres pages et components

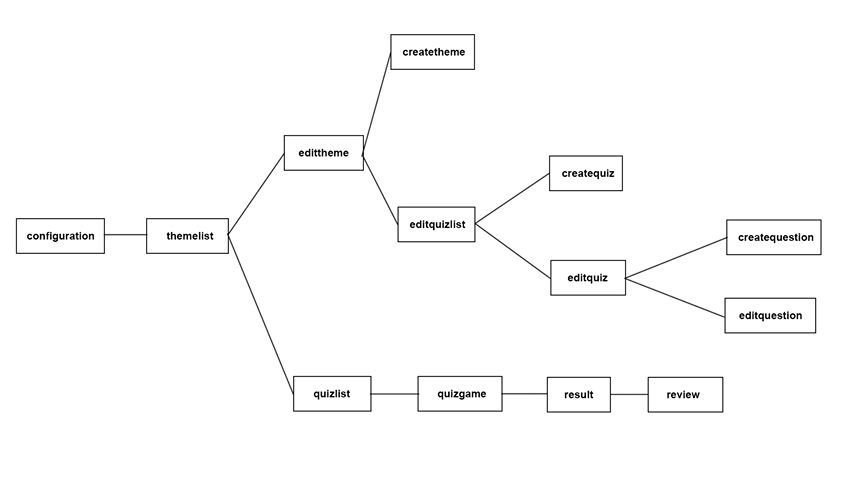


Figure 2 - Arborescence du site

Notre site comporte d’autres pages et donc d’autres components dans sa parte « création ». Cette dernière est là pour pouvoir éditer le contenu du site, ses thèmes, ses quiz, ses questions…

Parmi ces pages-là, ces components-là, on retrouve finalement deux types : les pages de liste et les pages de créations. Beaucoup d’éléments de notre site se ressemblent et ces dernières ne font pas exception. Les pages de liste ont, comme nous l’avions dit, toutes un carousel pour afficher leur contenu et les pages de créations ont toutes un formulaire pour récupérer les données que l’utilisateur saisit.

Aussi, puisque nous avons déjà vu ces principaux éléments et qu’on ne peut pas détailler tout le fonctionnement du site, nous n’irons pas plus loin dans la présentation de nos components.

### Quiz Service

Pour conclure cette présentation de notre partie front, il nous faut parler du service principal de notre site.

Le QuizService est au cœur du fonctionnement du notre projet, c’est lui qui communique avec le back pour récupérer, mettre à jour ou insérer des données. Il a aussi pour rôle de garder les sélections courantes de l’utilisateur. Par exemple, si l’utilisateur va dans le mode création, choisie un thème A et édite un quiz B, le QuizService stockera ces deux éléments et la page d’édition de quiz n’aura qu’à les récupérer pour savoir quel quiz afficher. Ici, l’objectif est encore une fois de ne rien transmettre à travers l’url.

## Back

Le back récupère les requêtes provenant du front afin de renvoyer les données demandées par celui-ci. Nous allons vous expliquer plus en détails le fonctionnement du back.

Nous possédons plusieurs routes distinctes, effectuant des instructions spécifiques selon celle utilisée.

Pour récupérer les données, il faut spécifier dans la requête GET, les données de l’élément que nous voulons. Par exemple, si nous voulons récupérer tous les quiz d’un thème, il nous suffira d’utiliser la route GET suivante :

/theme/:id

Avec « :id » qui correspond au nom du thème en question. De la même manière, si nous voulons avoir les données d’un quiz se trouvant dans un thème, on utilisera la route GET suivante :

/:theme/:id

Avec « :theme » et « :id » les noms du thème et du quiz respectif.

Il n’y a pas de route spécifique pour récupérer de question, en effet, le front a besoin uniquement du quiz voulu et récupèrera les questions via le quiz.

Maintenant, dans le cas d’ajout de thème, de question ou bien de quiz, il faudra utiliser une requête POST.

Il suffira d’utiliser le suffixe « /add » à la fin de la route. Par exemple, si nous voulons ajouter un nouveau thème, la route utilisée sera :

/add

Nous pouvons remarquer ici la non-présence d’une variable spécifiant un thème. Ici, nous ne sommes pas dans une requête demandant de renvoyer des données. Par conséquent, nous avons juste besoin de rajouter dans les données du back le thème crée se trouvant dans le corps de la requête.

Pour rajouter un quiz, il nous faut l’information du thème dans lequel nous voulons l’ajouter. La route ressemblera donc à :

/:theme/add

Avec « :theme » qui correspond au nom du thème.

Pour finir, nous passerons aux routes pour supprimer des questions, celle-ci utiliserons la requête DELETE. Ces routes ressemblent beaucoup à celles utilisées dans les requêtes GET. Comme les explications de leur fonctionnement ont été données précédemment, nous allons seulement vous montrer les routes que nous utilisons.

Pour supprimer un thème, nous utilisons la route suivante :

Pour supprimer un quiz :

/:theme/:id

Il y a une exception, c’est pour supprimer une question. Contrairement aux requêtes POST et GET, la requête DELETE est la seule qui a une route pour changer les questions. À l’instar de l’ajout, il nous faut savoir la position de la question dans le quiz. Dans le cas de l’ajout de question, il suffit juste de le rajouter à la fin de la liste des questions.

Mais dans le cas d’une suppression, nous avons besoin de connaître son emplacement afin de ne pas supprimer la mauvaise question.

/:theme/:id/:label

Le « label » correspond à la position de la question dans la liste du quiz de nom « :id » se trouvant dans le thème de nom « :theme»

Pour la sauvegarde des données, nous utilisons un fichier JSON. Pour cela, nous avons des modèles d’objets, un pour le quiz, un pour le thème, et un autre pour les questions.

Ainsi notre fichier JSON se compose d’une liste de thème, et l’objet thème est défini de la façon suivante :

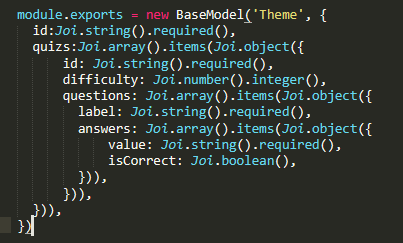


Figure 2- Modèle de l'objet Thème

Comme on peut le voir, l’objet Thème possède un « id », qui correspond au nom du thème et d’une liste d’objet Quiz. Cet objet Quiz est défini avec un « id », le nom du quiz, le niveau de difficulté de celui-ci et une liste de d’objet Question.

Une question est composée d’un « label », qui correspond à l’intitulé de la question, et de réponse. Dans notre projet, la liste des réponses est toujours de taille 4.

Concernant les ressources, nous utilisons le modèle Quiz pour faire toutes les opérations nécessaires sur la gestion des données de l’application.

Nous récupérons dans le corps de la requête un objet quiz, et c’est avec le modèle du quiz que l’on peut voir s’il y a eu un changement sur le quiz. Que se soit le niveau de difficulté, un ajout de question, ou autre.

C’est pour cela que dans les routes de notre API, nous spécifions toujours le thème dans lequel le quiz se trouve. Comme l’objet quiz ne nous donne pas d’information sur le thème, il fallait récupérer l’information d’une autre manière.

# Présentation de l’évaluation croisée et analyse des résultats obtenus

L’évaluation croisée est un moyen qui nous permet d’identifier les problèmes les plus importants, que ce soit de l’ordre fonctionnel ou bien de la compréhension de l’application, fait avec des utilisateurs lambdas. Le but de cette démarche est d’obtenir des retours sur le site qui permettront de le modifier afin qu’il soit plus compréhensible pour les utilisateurs.

Pour ce projet, nous avons été mis en binôme, avec l’équipe « Oui ». Nous avons opté pour l’évaluation v2, celle où nous sommes la main de l’utilisateur. Ce choix vient juste du fait que nous ne voulions pas perdre de temps à télécharger les projets de l’autre équipe et de le mettre en place pour qu’il fonctionne. Dans des conditions normales, nous aurions évidemment choisi l’évaluation v1 en laissant le testeur sur notre machine pour qu’il teste l’application.

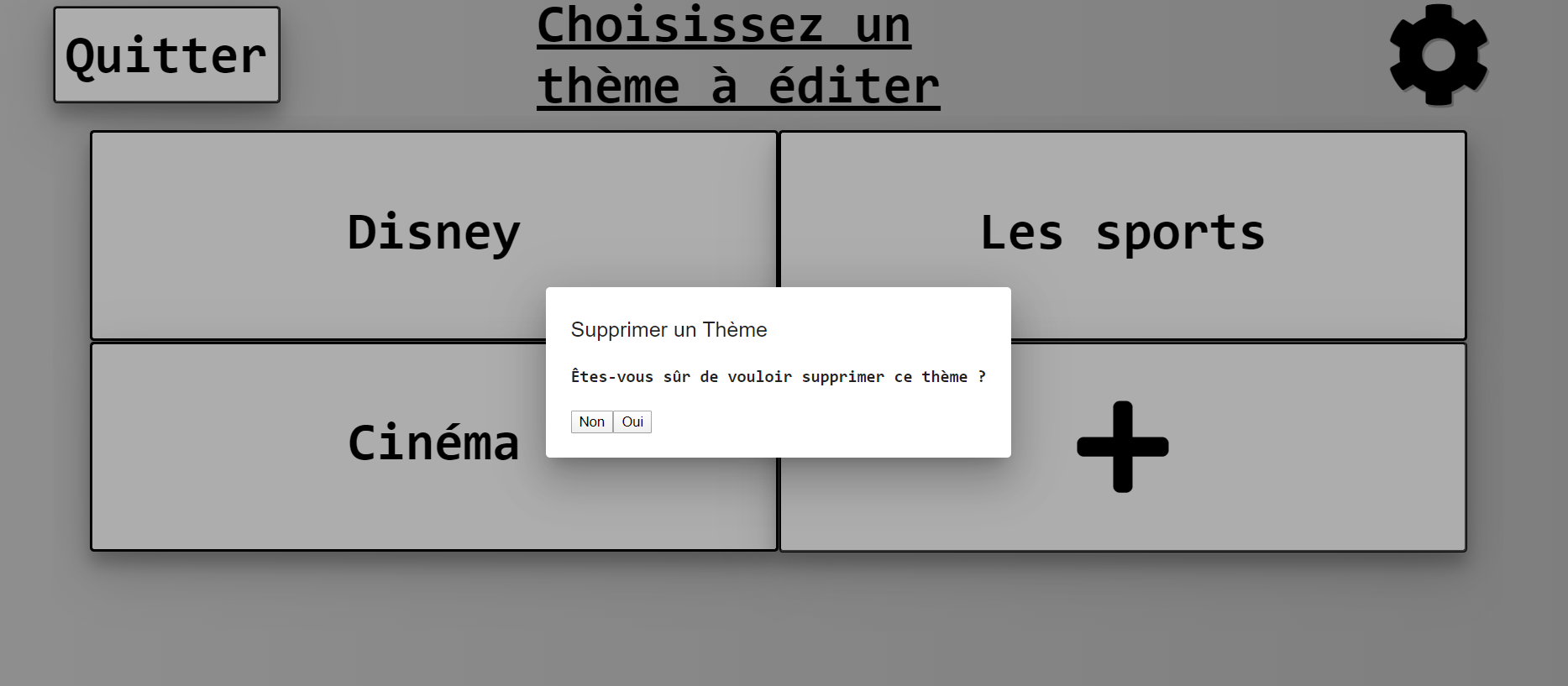
À la fin de celle-ci, le groupe « Oui » nous a fait un retour sur l’application (les bons côtés, ce qui n’allait pas, etc…)

Nous allons donc vous expliquer les différentes remarques que nous avons eu :

Sur le point fonctionnel, l’application répond aux besoins des différents personnas. Il n’y a pas de problème particulier. On peut modifier aisément un quiz ou une question.

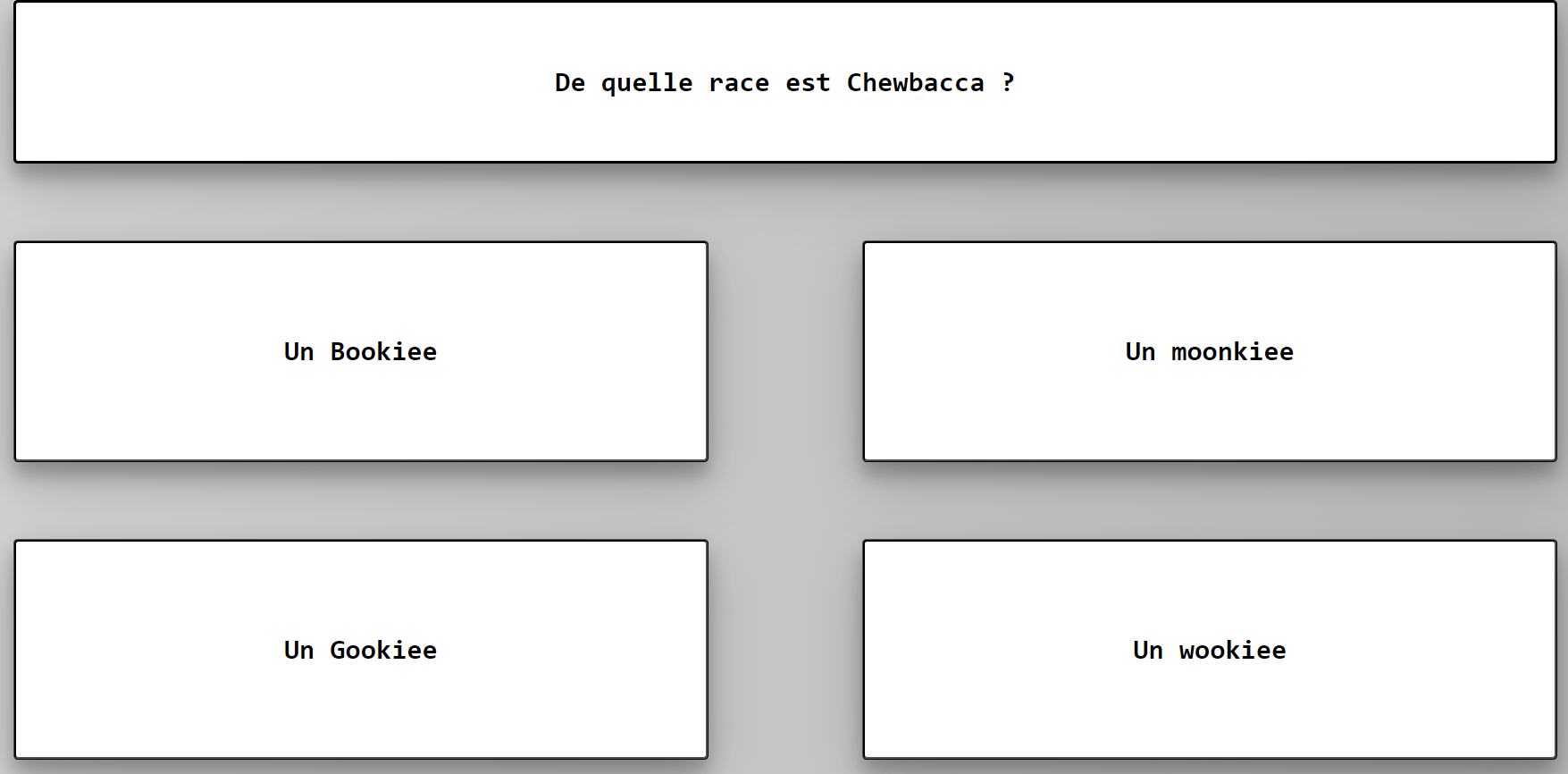
Concernant l’IHM, l’application manque d’élément, il est certes fonctionnel, mais lorsque l’on regarde l’application, celle-ci fait « vide ». Ils ont proposé de rajouter des éléments visuels tel qu’une banderole en haut de la page pour combler le vide. Cependant, nous n’avons pas ajouté d’ « élément » car nous voulions au maximum simplifier l’interface pour les personnes âgées en enlevant des informations non essentielles.

Ils ont aussi noté que la taille des caractères dans le pop-up qui permettent de confirmer la suppression d’un élément sur le site ne sont pas conformes à la taille que l’utilisateur a choisi dans la page de configuration mais leur taille reste fix à une taille standard comme montré ci-dessous :



Comme ce problème était rapide à résoudre, nous l’avons donc corrigé.

Nous avons eu aussi des retours à propos de la différenciation entre les éléments cliquables et ceux qui ne le sont pas, et cela peut créer des confusions pour les personnes âgées alors il sera mieux de bien les guider et préciser quels sont les endroits où ils doivent cliquer. Comme vous pouvez le voir, la plupart de nos éléments sont dans des boites, mais quelques-unes ne sont pas cliquables. Par exemple lors du quiz, la boite où se trouve la question ne possède aucune interaction :



Pour pallier à ce problème, il faudrait trouver un autre élément visuel qui pourrait différencier les éléments cliquables et ceux qui ne le sont pas.

Pour finir, la partie relecture pourrait être développée davantage, vu que pour l’instant, cela nous permet juste de relire toutes les questions avec les réponses correctes.

La piste d’amélioration serait d’ajouter la réponse de l’utilisateur et faire apparaitre une image si le choix de l’utilisateur correspond à la bonne réponse (un logo vert) ou bien l’inverse (une croix rouge si cela ne correspond pas) pour chaque question du quiz.

# Conclusion et perspective

Au final le projet était un bon outil pour assimiler les connaissances et développer nos capacités au niveau d’Angular et aussi de viser à créer un produit centré utilisateur .

Notre organisation était bonne, il y avait une bonne cohésion de groupe. Nous avons appris à bien gérer le projet et résoudre les problèmes qu’on a rencontré.

En plus d’être un projet pédagogique, on était contents de créer un outil qui servira pour d’autres personnes surtout si c’est des personnes avec des difficultées.

# Annexes

# Répartition des tâches dans l’équipe

# Vidéo de démonstration finale

1. Le glaucome est une maladie dégénérative du nerf optique qui entraîne une perte progressive de la vision commençant tout d'abord en périphérie et progressant graduellement vers le centre. (<https://fr.wikipedia.org/wiki/Glaucome>) [↑](#footnote-ref-2)
2. La cataracte est l'opacification partielle ou totale du cristallin, lentille convergente située à l'intérieur de l'œil. Cette opacification est responsable d'une baisse progressive de la vue, au début accompagnée de gêne à la lumière (photophobie). (<https://fr.wikipedia.org/wiki/Cataracte_(maladie)>) [↑](#footnote-ref-3)