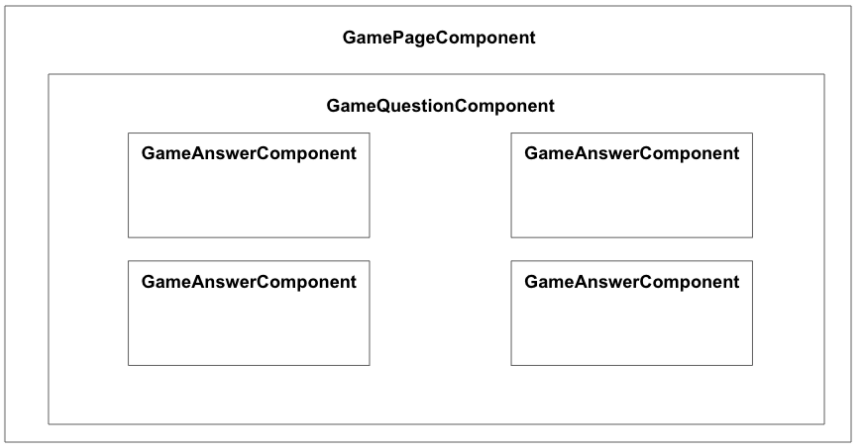
Jouer à un Quiz

Afficher un quiz

L'objectif de cette partie est de vous apprendre à afficher un quiz ressemblant au schéma ci-dessous. Cependant, vous êtes libres de (voire même encouragés à) modifier la structure pour qu'elle s'adapte au mieux à votre maquette.



Il vous faudra tout d'abord créer 3 composants (vides pour l'instant) :

1. GamePageComponent, qui représentera la page d'affichage d'un quiz. N'oubliez pas d'associer ce composant à une URL pour pouvoir appeler la page.
2. GameQuestionComponent qui vous servira à afficher la question et les réponses.
3. GameAnswerComponent qui gèrera l'affichage des réponses.

L'idée étant que votre GamePageComponent contienne un GameQuestionComponent, contenant lui-même autant de GameAnswerComponent que vous désirez (4 dans l'exemple ci-dessus).

Il vous faut également réfléchir à vos modèles :

- Comme nous l'avons vu dans le cours/les vidéos/la correction, nous avons des exemples de modèles qui définissent vos quizs, questions et réponses. Exemple ici.
- Le modèle pour jouer un quiz est un peu différent car nous ne modélisons pas un quiz mais le fait de jouer à un quiz, cad une instance de jeu. Il nous faut donc un modèle différent qui nous permettra de connaitre le quiz qui a été joué, les réponses données, la durée du jeu etc... Nous vous proposons un exemple de modèle de jeu que vous pouvez utiliser. Il faudra certainement que vous la modifiez (maintenant ou plus tard) pour l'adapter à vos besoins.

```
GameInstance {
        Id: string;
        quizId:string;
        gameQuestionsAnswers: GameQuestionAnswer[ ];
        startTime: Date;
        endTime: Date;
}

GameQuestionAnswer {
        startDate: Date;
        submissionDate: Date;
        questionId: string;  // or question: Question to keep the question and answers given even if the question or answers are updated later
        answerIds: string[]; // or answers: Answers[]
}
```

- Créer vos modèles
- Créer vos mocks

Ensuite, il faut faire en sorte que GameQuestionComponent sache quelle question afficher. Pour cela, vous devez :

- Créer un service GameService qui expose un observable avec la question courante à afficher et ses réponses. Pour l'instant, vous utiliserez un mock pour vos questions/réponses.
- Dans GameQuestionComponent, récupérer les données de l'observable créé à l'étape précédente.
- Toujours dans GameQuestionComponent, côté HTML, afficher la question et les réponses. Pour afficher les réponses, il vous faudra utiliser la directive ngFor pour boucler sur les réponses et appeler à chaque fois le composant GameAnswerComponent. La réponse à afficher par le composant lui est transmis grâce à un @Input.

Soumettre le résultat d'une question

- Ajouter un évènement click dans le composant GameAnswerComponent, de sorte que quand l'utilisateur clique sur une réponse, le composant GameAnswerComponent renvoie la réponse au composant parent grâce à un @Output.
- Dans GameQuestionComponent, récupérer la valeur émise par GameAnswerComponent et informer GameService (en appelant une de ses méthodes) de la réponse choisie par l'utilisateur.

Naviguer vers la question suivante

Quand GameService est informé d'une réponse :

- Mettre à jour l'instance de jeu, avec le score et tout ce dont vous avez besoin. Vous devez modifier le modèle d'instance jeu en fonction de vos besoins en terme de score.
- Passer à la question suivante de la liste. N'oubliez pas d'emit le changement de question via votre observable créé précédemment, de façon à ce que le composant affichant la question en cours soit automatiquement mis à jour avec la nouvelle question.

> Vous seuls savez ce que vous devez faire quand l'utilisateur choisit une réponse (afficher un message d'erreur ou de réussite, proposer de retenter, remettre la question dans la liste si la réponse était mauvaise, ou encore ne rien faire et passer à la question d'après...). Dans tous les cas, créez les observables et fonctions permettant de faire ce dont vous avez besoin et n'oubliez pas de faire communiquer votre service gérant la partie avec votre composant d'affichage.

Soumettre la fin du jeu

Quand votre service reçoit la réponse à la dernière partie du jeu, il est temps d'Afficher les résultats.

Afficher les résultats

Cette partie dépend quasi-entièrement de votre projet (certains n'afficheront potentiellement même pas les résultats), donc peu de conseils génériques à vous donner, mais la marche à suivre est similaire à l'affichage d'un quiz :

- Créer un composant pour afficher les résultats, et tous les sous-composants dont vous pourriez avoir besoin.
- Associer le composant d'affichage à une URL pour pouvoir rediriger l'utilisateur vers ses résultats.
- Intégrez le service (GameService) qui contient les informations sur le résultat du quiz.
- N'oubliez pas de rajouter des boutons/liens pour revenir aux menus qui vous semblent pertinents.

Connexion / choisir son utilisateur

Nous n'allons pas mettre en place un vrai système d'inscription et de connexion car ces fonctionnalités ne sont pas nécessaires dans le cadre de notre projet (sauf si elles le sont d'après votre maquette).
Ce que nous souhaitons faire, c'est un système de connexion plus simple qui nous permet de choisir au lancement de l'application l'utilisateur avec lequel nous souhaitons jouer/interagir (comme sur Netflix).
Si vous souhaitez afficher une liste d'utilisateurs pour que l'utilisateur choisisse, il vous faudra :

- Créer un service exposant un observable contenant la liste des utilisateurs. Dans un premier temps, cette liste sera un mock.
- Créer un composant servant à afficher la liste et un autre pour afficher un seul profil.
- Souscrire au service pour récupérer la liste des utilisateurs depuis votre service.
- Afficher la liste avec ngFor et appeler à chaque fois le composant permettant d'afficher un profil. Les détails du profil seront transmis via un @Input.
- Quand l'utilisateur clique sur un profil, faire remonter l'information jusqu'au service grâce à un @Output.

Gérer un quiz

Créer un quiz

Pour créer un quiz, vous devez :

- Avoir un modèle décrivant les champs nécessaires pour créer un quiz.
- Avoir un service contenant (entre autres) la liste des quizs.
- Créer une fonction `add(quiz: Quiz)` dans votre service, permettant d'ajouter un quiz à la liste des quizs (N'oubliez d'emit si vous avez un observable sur la liste des quizs pour que tous les composants soient au courant de l'ajout).
- Créer un composant permettant à l'utilisateur courant de créer un nouveau quiz.
- Dans ce composant, créer un formulaire contenant les mêmes champs que votre modèle de quiz.
- Quand l'utilisateur valide sa saisie, appeler la fonction add du service pour ajouter le quiz à la liste.

Note: Vous aurez peut-être besoin de créer des sous-composants pour pouvoir ajouter des questions à votre quiz.

Modifier un quiz

Pareil que créer un quiz, à ceci près que :

- Il faut ajouter une méthode `update(quiz: Quiz)` qui ne va cette fois pas ajouter le quiz à la liste mais modifier le quiz de la liste passé en paramètre (retrouvé via son id par exemple). À noter qu'il faut tout de même emit

n'importe quel observable sur la liste des quizs.
- Le formulaire doit déjà être pré-rempli avec les données actuelles du quiz.

Supprimer un quiz

Encore plus simple, il suffit de créer et d'appeler une méthode `delete(quiz: Quiz)` dans le service qui va supprimer l'utilisateur de la liste.

Gérer un utilisateur

c.f. Gérer un quiz, en remplaçant le mot quiz par le mot utilisateur.

> Bien que les étapes soient effectivement les mêmes que pour gérer un quiz, n'oubliez pas les cas particuliers des utilisateurs, notamment de différencier les patients du personnel, en attribuant par exemple des rôles spécifiques à chacun.

Afficher la liste des quizs

Pour afficher la liste des quizs, il vous faut :

- Créer un service exposant un observable contenant la liste des quizs. Pour l'instant, la liste des quizs sera un mock.
- Créer un composant (qu'on appellera composant parent) qui servira à afficher la liste des quizs.
- Associer une URL à ce composant pour pouvoir y accéder.
- Dans le composant, souscrire à la liste des quizs du service pour avoir toujours la liste à jour.
- Créer un sous-composant (qu'on appellera composant enfant) affichant les détails d'un quiz. Ce sous-composant sera appelé dans le composant créé juste avant pour chaque quiz à afficher.
- Utiliser ngFor dans le composant parent pour appeler le composant enfant pour chaque quiz. Les détails sur le quiz courant seront transmis au composant enfant via @Input.
- Créer un évènement clic dans le composant enfant pour savoir quand un utilisateur a sélectionné un quiz.
- Remonter le quiz sélectionné au parent grâce à @Output.
- Prévenir le service du quiz sélectionné (en appelant une fonction du service que vous aurez créé par exemple)
- Rediriger l'utilisateur vers la partie Jouer à un quiz.

Connexion

Nous n'allons pas mettre en place un vrai système d'inscription et de connexion car ces fonctionnalités ne sont pas nécessaires dans le cadre de notre projet (sauf si elles le sont d'après votre maquette).
Ce que nous souhaitons faire, c'est un système de connexion plus simple qui nous permet de choisir au lancement de l'application l'utilisateur avec lequel nous souhaitons jouer/interagir (comme sur Netflix).

Pour permettre à vos utilisateurs de se connecter, il vous faut :

- Créer une page
- Afficher la liste des utilisateurs
- Au clique sur un utilisateur, naviguer jusqu'à la prochaine page.

Contacter l'assistance du site

Connecté sous le nom « duong Thi Thanh Tu » (Déconnexion)
Résumé de conservation de données
Obtenir l'app mobile

Fourni par Moodle