Dossier de conception - Projet Tarot

Sommaire

Introduction . . . 1
Présentation du projet . . . 1
La liste des fonctionnalités minimales développées . . . 1
La liste des fonctionnalités bonus développées . . . 2
Présentation rapide des classes . . . 2
  Les services . . . 2
  Les models . . . 2
  Les composants . . . 3
Présentation de l'ergonomie de l'application . . . 4
  Présentation de l'interface . . . 4
Présentation des routes . . . 4
  Listage des routes . . . 4
    5 routes statiques . . . 4
    2 routes dynamiques . . . 4
  Passage d'une route à une autre . . . 4

Introduction

Ce projet a été réalisé dans le cadre du cours de développement web de l'année 2023 à l'Université de Blagnac par:

- Hugo Wendjaneh - Groupe: 2A

Présentation du projet

Le projet consiste à réaliser une application web via Angular.
Il s'agit d'un compteur de points pour les parties de tarot. Il permet aussi d'afficher le détail de chaque manche de chaque partie.

La liste des fonctionnalités minimales développées

1. Création d'une nouvelle partie
2. Sélection/Reprise d'une ancienne partie
3. Détail d'une partie avec toutes ces manches

4. Saisie des informations d'une manche pour le comptage automatique des points

La liste des fonctionnalités bonus développées

1. Affichage du détail d'une partie sous forme de graphique dinamique (via Chart.js).
2. Barre de navigation dynamique
3. Suppréssion d'une partie
4. Stockage de la date de la partie
5. CSS personnalisé
6. Site totalement responsive
7. Redirection vers la page d'accueil en cas de route non définie. Exemple: `/jenesaispas`
8. Redirection vers la page d'accueil en cas de route invalide. Exemple: l'id d'une partie n'existe pas: `/game/1230` ou `/details/1230`

Présentation rapide des classes

Les services

Il y a 2 services:

1. `GameService`: Permet de gérer la communication avec le JSON Server. Il permet de récupérer les données, de les modifier et de les supprimer.
2. `navigationHistoryService`: Permet de gérer la navigation entre les routes. Il permet de stocker la route précédente et la route actuelle afin de pouvoir revenir à la route précédente dans certaines conditions.

Les models

Il y a 5 models:

1. `Game`: Permet de gérer les données d'une partie. Il contient les attributs suivants:
    a. `id`: L'id de la partie
    b. `time`: La date de la partie
    c. `players`: Les joueurs de la partie
2. `Player`: Permet de gérer les données d'un joueur. Il contient les attributs suivants:
    a. `name`: Le nom du joueur
    b. `score`: Le score du joueur
3. `Round`: Permet de gérer les données d'une manche. Il contient les attributs suivants:

    a. `id`: L'id de la manche
    b. `idGame`: L'id de la partie
    c. `roundNumber`: Le numéro de la manche
    d. `scores`: Les scores de chaque joueurs la manche
4. `Time`: Permet de gérer les données d'une date. Il contient les attributs suivants:
    a. `date`: La date
    b. `hour`: L'heure
    c. Il a aussi une méthode statique public qui permet de récupérer un `time` avec la date et l'heure actuel bien formaté. `getTime`: Retourne un `time` avec la date et l'heure actuel bien formaté.
5. `loading-status`: Permet de gérer le status de chargement des données. C'est une class enum qui contient les attributs suivants:
    a. `LOADING`: Le status de chargement
    b. `LOADED`: Le status de chargement terminé
    c. `ERROR`: Le status d'erreur

Les composants

Tout les composant sont dans le dossier tarot.

1. `game`: Permet de jouer une partie. Il permet de saisir les informations d'une manche et de compter automatiquement les points. Il appel aussi le service `GameService` pour récupérer les données de la partie et modifier / ajouté des données.
2. `new`: C'est un composant qui est placé dans le composant game pour indiquer que l'on est en train de créer une nouvelle partie. Il permet de saisir les informations d'une nouvelle partie comme par exemple le nom des joueurs. Il appel aussi le service `GameService` pour créer une nouvelle partie.
3. `game-details`: Permet d'afficher les détails d'une partie. Il à un composant enfant `game-details-graphic-item` qui va ce charger de l'affichage du graphique. Il appel aussi le service `GameService` pour récupérer les données de la partie.
4. `game-details-graphic-item`: Permet d'afficher le graphique de la partie. Il utilise la librairie Chart.js pour afficher le graphique. Il appel aussi le service `GameService` pour récupérer les données de la partie.
5. `game-list`: Permet d'afficher la liste des parties. Il a un composant enfant `game-list-item` qui va ce charger de l'affichage d'une partie. Il appel aussi le service `GameService` pour récupérer les données de la partie.
6. `game-list-item`: Permet d'afficher une partie. Il appel aussi le service `GameService` pour récupérer les données de la partie.
7. `home`: Permet d'afficher la page d'accueil.
8. `nav-bar`: Permet d'afficher la barre de navigation. Il appel aussi le service `navigationHistoryService` pour récupérer la route précédente et la route actuelle.

Présentation de l'ergonomie de l'application

Présentation de l'interface

1. La barre de navigation est présente sur toutes les pages sauf la page d'accueil. Elle permet de naviguer entre les différentes pages de l'application. Elle est dynamique et affiche le bouton `Retour` si une partie est en cours.
2. La page d'accueil permet de créer une nouvelle partie ou de voir les anciennes parties.
3. La page de création d'une nouvelle partie permet de saisir les informations d'une nouvelle partie comme par exemple le nom des joueurs.
4. La page de détail d'une partie permet d'afficher les détails d'une partie. Il y a un graphique qui permet d'afficher l'évolution du score de chaque joueur au cours de la partie.
5. La page de liste des parties permet d'afficher la liste des parties. Il y a un bouton pour supprimer une partie. Il y a un bouton pour afficher les détails d'une partie. Il y a un bouton pour continuer une partie.
6. La page de jeu permet de jouer une partie. Elle permet de saisir les informations d'une manche et de compter automatiquement les points.

Présentation des routes

Listage des routes

5 routes statiques

1. ou `/`: Redirige vers la route `/home`
2. `/home`: Affiche la page d'accueil
3. `/games/list`: Affiche la liste des parties
4. `new/game`: Affiche le formulaire de création d'une nouvelle partie
5. `**`: Tout autre route non définie redirige vers la route `/home`

2 routes dynamiques

1. `/details/:id`: Affiche le détail d'une partie en fonction de son id
2. `game/:id`: Affiche le formulaire de saisie d'une manche en fonction de l'id de la partie

Passage d'une route à une autre

Depuis la route `/home`, nous pouvons accéder à la route `/new/game` via le bouton `Nouvelle partie` et à la route `/games/list` via le bouton `Anciennes parties`.

Depuis la route `/new/game`, via la navBar nous pouvons accéder à la route `/home` via le bouton `Accueil`

et à la route `/games/list` via le bouton `Liste des parties`.

Depuis la route `/games/list`, via la navBar nous pouvons accéder à la route `/home` via le bouton `Accueil` et à la route `/new/game` via le bouton `Nouvelle partie`.

Depuis la route `/games/list`, nous pouvons accéder à la route `/details/:id` via le bouton `Détails de la partie` et à la route `/game/:id` via le bouton `Continuer la partie`. Nous pouvons aussi aller à la route `/new/game` via le bouton `Nouvelle partie` dans la navBar.

Depuis la route `/details/:id`, via la navBar nous pouvons accéder à la route `/home` via le bouton `Accueil` et à la route `/games/list` via le bouton `Liste des parties`. Nous pouvons aussi aller à la route `/new/game` via le bouton `Nouvelle partie`.
Si nous étions sur une partie en cours `/game/:id`, nous pouvons aussi accéder à la route `/game/:id` via le bouton `Retour`.

Depuis la route `/game/:id`, via la navBar nous pouvons accéder à la route `/home` via le bouton `Accueil` et à la route `/games/list` via le bouton `Liste des parties`. Nous pouvons aussi aller à la route `/new/game` via le bouton `Nouvelle partie` et nous pouvons aussi accéder à la route `/details/:id` via le bouton `Détails de la partie`.