Documentation du Projet de Jeu de Dés

Table des Matières

Introduction

Ce projet est un jeu de dés développé en React pour le frontend et Express.js pour le backend, avec une gestion de scores et sessions stockées dans une base de données MySQL. Le jeu permet aux utilisateurs de lancer des dés, enregistrer les scores, et terminer des sessions de jeu. Les scores sont affichés dans une page dédiée.

Prérequis

- Node.js et npm
- MySQL
- Un navigateur web moderne

Installation

14. **Clonez le dépôt :**

```
Bash
Copier le code
```

```
git clone https://github.com/KevinOlinga/dice-game.git
cd dice-game
```

15. **Installez les dépendances pour le backend :**

```
Bash
cd dice_game_backend
npm install
```

16. **Installez les dépendances pour le frontend :**

```
Bash
cd ../dice_game
npm install
```

17. **Configurez la base de données MySQL :**

    - Assurez-vous que MySQL est installé et en cours d'exécution. (MySQL 8.4 de préférence)
    - Importez la base de données :

```
Bash

mysql -u root -p < database/database_backup.sql
```

Configuration

18. **Configurer les variables d'environnement pour le backend :**

Créez un fichier `.env` à la racine du dossier `dice_game_backend` et ajoutez-y les variables suivantes :

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=votre-mot-de-passe
DB_NAME=dice_game
PORT=3001
```

19. **Configurer les routes de l'API :**

Les routes sont définies dans `dice_game_backend/app.js`. Assurez-vous que les routes de l'API sont correctement configurées.

Utilisation

20. **Démarrez le serveur backend :**

```
Bash
cd dice_game_backend
node app.js
```

21. **Démarrez l'application frontend :**

```
Bash
cd ../dice_game
npm start
```

22. **Accédez à l'application dans votre navigateur :**

http://localhost:3000

Structure du Projet

```
/dice-game
│
├── /dice_game_backend
│   ├── /node-modules
│
│   ├── .env
│   ├── app.js
│   ├── package.json
│   └── ...

│
├── /dice_game_frontend
│   ├── /src
│   │   ├── /components
│   │   │   ├── DiceGame.js
│   │   │   ├── Score.js
```

```
│   │   │   ├── Header.js
│   │   │   └── Footer.js
│   │   ├── /assets
│   │   │   ├── images
│   │   │   │   ├── dice-face-1.png
│   │   │   │   ├── dice-face-2.png
│   │   │   │   ├── dice-face-3.png
│   │   │   │   ├── dice-face-4.png
│   │   │   │   ├── dice-face-5.png
│   │   │   │   └── dice-face-6.png
│   │   ├── App.js
│   │   ├── index.js
│   └── /public
│       ├── index.html
│       └── ...
│
├── /database
│   │──dice_game.sql
└── README.md
```

Endpoint API

- **Créer une session :**

```
POST /sessions
```

- **Créer un jeu :**

```
POST /games
```

- **Terminer une session :**

```
PUT /sessions/:sessionId
```

- **Récupérer les détails d'un joueur :**

```
GET /players/:name
```

Composants

23. **DiceGame.js :**
    - Gère le jeu de dés.
    - Affiche le nom du joueur, la valeur actuelle du dé, et les boutons pour lancer le dé ou passer le tour.
    - Enregistre les scores dans `localStorage` et met à jour la base de données via des appels API.
24. **Score.js :**
    - Affiche les scores des différentes sessions de jeu.
    - Récupère les données de `localStorage` et les affiche dans une liste.
25. **Header.js :**
    - Affiche l'en-tête avec le menu de navigation.
    - Comprend des liens vers différentes sections de l'application.
26. **Footer.js :**
    - Affiche le pied de page de l'application.
    - Contient des informations supplémentaires ou des liens pertinents.

Fonctionnalités

- **Lancer un dé et enregistrer le score :** Le joueur peut lancer un dé, et le score est enregistré pour la session en cours.
- **Passer son tour :** Le joueur peut passer son tour, enregistrant un score de 0.
- **Terminer une session :** Le joueur peut terminer une session, et les scores de toutes les parties de cette session sont affichés.
- **Voir les scores de toutes les sessions terminées :** Les scores sont stockés et peuvent être consultés ultérieurement.

Fonctionnalités en Cours de Développement

Les fonctionnalités suivantes sont encore en cours de développement conformément au projet 3ICP BXL 1 :

- L’espace d’administration pour configurer les parties.
- Les tests unitaires
- La virtualisation avec VM ou dockers.

Projet réalisé par Kevin, Yamina et Norbert.