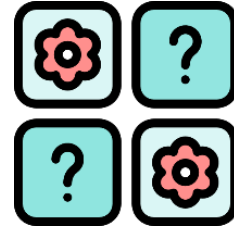


Jeu de mémoire

Seul ou en équipe de 2. À partir du prototype *react-drf-tp4-partial*, créez un jeu complet qui supporte toutes les fonctionnalités ciblées.

Votre application **React** doit être codée en **TypeScript** et utiliser **Material UI** pour l'entièreté du visuel (aucune balise html). Votre service de données **Python Django**, doit utiliser **Django REST Framework**, ainsi qu'une base de données **SQLite**.



*À vous de proposer un contenu, une disposition et un visuel intéressant pour votre application.
Inspirez-vous des exemples des différents composants sur le site de Material UI.*

1. Les améliorations du prototype existant (React)

- Utiliser plutôt les images fournies au lieu des emojis.** Lors de l'initialisation de la partie, faites une sélection aléatoire parmi les 36 images.
- Permettre de modifier la couleur du dos des cartes.**
 - Proposez 3 choix de couleur.
- Jouer une partie :** 4x4 par défaut.
 - Niveaux de jeu disponibles : 4x3, 4x4, 5x4, 6x4, 6x5 ou 6x6
- Ajouter une option "Recommencer"** lorsque la partie est terminée.

2. Le service de données (Django REST Framework)

Créez d'abord un projet **Django** de base dans lequel vous installerez les *packages* nécessaires dans votre environnement virtuel. Référez-vous au projet « démo ».

Vous ajouterez par la suite les **modèles**, les **serializers**, les **vues** et les **routes** nécessaires selon les différentes fonctionnalités supportées par votre application.

3. La création d'un profil de joueur

En mode « invité », un visiteur peut :


- **Créer un nouveau profil** (/signup)
- **Se connecter** (/login)

En mode « connecté », un joueur peut :



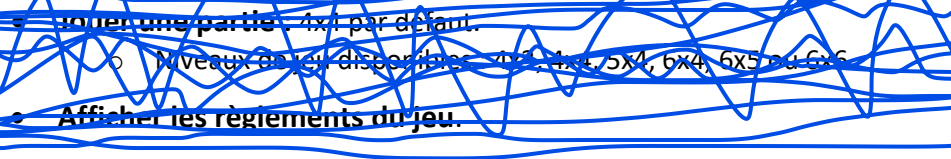
4. Les fonctionnalités

En mode « invité », un visiteur pourra :



- **Afficher les évaluations laissées par les joueurs enregistrés.**
 - En ordre décroissant de date (la plus récente en premier).
 - La date + heure, le nom de l'utilisateur, la note et le commentaire.

En mode « connecté », un joueur pourra :

- 
- **Afficher les évaluations laissées par les joueurs enregistrés.**
 - En ordre décroissant de date (la plus récente en premier).
 - La date + heure, le nom de l'utilisateur, la note et le commentaire.
 - **Afficher l'historique de ses dernières parties.**
 - Au choix du joueur : 20, 50 ou 100 (20 par défaut).
 - En ordre décroissant de date (la plus récente en premier).
 - La date + heure, le niveau, la durée et le nombre de tentatives.
 - Si la partie n'a pas été terminée, afficher "non complétée".

- **Laisser une évaluation du jeu.**
 - Une note sur 5 et un commentaire.
 - Permettre les demi-notes.
 - Utiliser le composant **Rating** : ★★★★★
 - Une seule évaluation par joueur.
- **Modifier son évaluation.**
 - La note sur 5 et/ou le commentaire.
 - Ce qui mettra à jour la date d'évaluation.
- **Supprimer son évaluation.**
 - Utiliser le composant **Dialog** pour demander une confirmation.

Vous êtes libre d'ajouter des fonctionnalités supplémentaires de votre choix. Ça sera bien sûr pris en considération lors de la correction.

Pour les équipes de 2, vous devez obligatoirement ajouter des fonctionnalités supplémentaires. Valider avec moi pour vous assurer que l'ampleur correspond aux attentes.

5. Déploiement de votre API et votre application



API : Django REST Framework

Référez-vous au tutoriel <https://clients.vtinyhosting.com/knowledgebase/20/>

- Archivez/Zippez préalablement le code source de votre API. **Ne pas inclure votre environnement virtuel!**
- Étape 1** du tutoriel : Connectez-vous à votre compte **cPanel**
- Étape 2** du tutoriel : Créez une application **Python 3.12**. Assurez-vous de sélectionner le sous-domaine déjà créé : **apirest**
- Via le **Gestionnaire de fichiers cPanel**, téléversez le ZIP contenant votre code source dans le dossier de votre **App Python** précédemment créée et dézippez-le. Si les fichiers sont dans un sous-répertoire, déplacez-les directement dans le répertoire de votre **App Python** et effacez ce sous-répertoire. Supprimez ensuite le ZIP téléchargé.

e) Modifiez le fichier **settings.py** de votre API :

```
CORS_ORIGIN_WHITELIST = (  
    'https://react.w4-XXX.vtynhosting.site',  
)
```

f) **Étape 3** du tutoriel (Configurez le projet Django) (sauf « **d** » qui est déjà fait).

g) Redémarrer l'**App Python** via cPanel

h) Testez votre API via votre navigateur et/ou un logiciel comme *Postman*.

Application web : React & Material UI

a) Assurez-vous que le contenu de votre fichier **.env.production** est correct.

b) Compilez votre application *React* (qui va créer un dossier **dist**) :

```
npm run build
```

c) Archivez/Zippez le contenu du dossier **dist**.

d) Via le **Gestionnaire de fichiers cPanel** :

- Téléversez le ZIP contenant votre code source dans le dossier cible de votre sous-domaine **react** (dans "**public_react**" possiblement) et dézippez-le. Si les fichiers sont dans un sous-répertoire, déplacez-les directement dans le répertoire cible et effacez ce sous-répertoire. Supprimez ensuite le ZIP téléchargé.
- Ajoutez ou modifiez le fichier « .htaccess ». Assurez-vous que les fichiers cachés sont visibles : bouton « **Paramètres** » en haut à droite. Voici le contenu du fichier « .htaccess » :

```
<IfModule mod_rewrite.c>  
    RewriteEngine On  
    RewriteBase /  
    RewriteRule ^index\.html$ - [L]  
    RewriteCond %{REQUEST_FILENAME} !-f  
    RewriteCond %{REQUEST_FILENAME} !-d  
    RewriteCond %{REQUEST_FILENAME} !-l  
    RewriteRule . /index.html [L]  
</IfModule>
```

a) Testez l'entièreté de votre site web!

Une partie « Extra » vous sera partagée bientôt...