Route backend

api.localhost/graphql

Résolveurs et Mutations par Entité

Authentification

Mutations

`register(email: String!, password: String!): AuthPayload!`

- **Description** : Enregistre un nouvel utilisateur avec une authentification classique (email et mot de passe).
- **Résultat** : Renvoie un jeton d'authentification (token) et les informations de l'utilisateur (user).

`login(email: String!, password: String!): AuthPayload!`

- **Description** : Authentifie un utilisateur existant avec son email et son mot de passe.
- **Résultat** : Renvoie un jeton d'authentification et les informations de l'utilisateur.

`oauthLogin(provider: String!, providerUserId: String!, email: String, username: String, avatar: String): AuthPayload!`

- **Description** : Authentifie un utilisateur via OAuth (par exemple, Google). Crée un nouvel utilisateur si nécessaire.
- **Résultat** : Renvoie un jeton d'authentification et les informations de l'utilisateur.

Types

AuthPayload

Champs :

- `token: String!` : Le jeton JWT pour l'authentification future.
- `user: User!` : Les informations de l'utilisateur authentifié.

Utilisateur (User)

Queries

`me: User`

- **Description** : Renvoie les informations de l'utilisateur actuellement authentifié.
- **Utilisation** : Permet à un utilisateur de récupérer ses propres informations.

`user(id: ID!): User`

- **Description** : Renvoie les informations d'un utilisateur spécifique par son identifiant.
- **Utilisation** : Afficher le profil d'un autre utilisateur.

```
users(filter: UserFilter, skip: Int, take: Int): [User!]
```

- **Description** : Renvoie une liste d'utilisateurs avec des options de filtrage, de pagination.
- **Utilisation** : Parcourir ou rechercher des utilisateurs.

Mutations

```
updateUser(username: String, avatar: String): User!
```

- **Description** : Met à jour les informations de l'utilisateur actuellement authentifié.
- **Utilisation** : Permet à l'utilisateur de modifier son profil.

```
deleteUser: Boolean!
```

- **Description** : Supprime le compte de l'utilisateur actuellement authentifié.
- **Utilisation** : Permet à l'utilisateur de supprimer son compte.

Mot (Word)

Queries

```
word(id: ID!): Word
```

- **Description** : Renvoie les informations d'un mot spécifique.
- **Utilisation** : Principalement pour des fins administratives.

```
words(filter: WordFilter, skip: Int, take: Int): [Word!]
```

- **Description** : Renvoie une liste de mots avec des options de filtrage et de pagination.
- **Utilisation** : Gérer les mots disponibles dans le jeu (administrateur).

Mutations

```
createWord(wordText: String!): Word!
```

- **Description** : Ajoute un nouveau mot à la liste des mots disponibles.
- **Utilisation** : Permet aux administrateurs d'ajouter de nouveaux mots.

```
deleteWord(id: ID!): Boolean!
```

- **Description** : Supprime un mot de la liste.
- **Utilisation** : Gérer les mots (administrateur).

Jeu (Game)

Queries

`game(id: ID!): Game`

- **Description** : Renvoie les informations d'un jeu spécifique.
- **Utilisation** : Afficher les détails d'un jeu.

`games(filter: GameFilter, skip: Int, take: Int): [Game!]`

- **Description** : Renvoie une liste de jeux avec des options de filtrage et de pagination.
- **Utilisation** : Permet aux utilisateurs de trouver des jeux à rejoindre.

Mutations

`createGame(maxPlayers: Int): Game!`

- **Description** : Crée un nouveau jeu avec un nombre maximum de joueurs.
- **Utilisation** : Permet à un utilisateur de créer un nouveau jeu.

`startGame(gameId: ID!): Game!`

- **Description** : Démarre un jeu existant.
- **Utilisation** : Utilisé par le créateur du jeu pour lancer la partie une fois que tous les joueurs ont rejoint.

`joinGame(gameId: ID!): GameSession!`

- **Description** : Permet à un utilisateur de rejoindre un jeu existant.
- **Utilisation** : Les utilisateurs utilisent cette mutation pour participer à un jeu.

`leaveGame(gameId: ID!): Boolean!`

- **Description** : Permet à un utilisateur de quitter un jeu avant qu'il ne commence.
- **Utilisation** : Gérer la participation au jeu.

Session de Jeu (GameSession)

Queries

`gameSession(id: ID!): GameSession`

- **Description** : Renvoie les informations d'une session de jeu spécifique.
- **Utilisation** : Afficher l'état actuel de la session de jeu pour un joueur.

`myGameSessions(status: String): [GameSession!]`

- **Description** : Renvoie les sessions de jeu de l'utilisateur authentifié, avec possibilité de filtrer par statut.
- **Utilisation** : Permet à l'utilisateur de voir ses parties en cours ou passées.

Mutations

`makeGuess(gameSessionId: ID!, guessWordText: String!): Guess!`

- **Description** : Enregistre une tentative de mot pour une session de jeu donnée.
- **Utilisation** : Utilisé par les joueurs pour soumettre leurs tentatives.

`endGameSession(gameSessionId: ID!): Boolean!`

- **Description** : Termine une session de jeu pour le joueur.
- **Utilisation** : Permet à un joueur de quitter une partie en cours.

Tentative (Guess)

Queries

`guesses(gameSessionId: ID!): [Guess!]`

- **Description** : Renvoie la liste des tentatives effectuées dans une session de jeu.
- **Utilisation** : Afficher l'historique des tentatives pour un joueur.

Souscriptions (Subscriptions)

Pour les fonctionnalités en temps réel, notamment pour le mode multijoueur :

Souscriptions pour les Jeux

`gameUpdated(gameId: ID!): Game`

- **Description** : Notifie les clients lorsque des changements surviennent dans un jeu spécifique (par exemple, changement de statut, nouveaux joueurs).
- **Utilisation** : Mettre à jour l'interface des joueurs en temps réel.

Souscriptions pour les Sessions de Jeu

`gameSessionUpdated(gameSessionId: ID!): GameSession`

- **Description** : Notifie le joueur lorsque sa session de jeu est mise à jour (par exemple, nouvelle tentative enregistrée).
- **Utilisation** : Actualiser l'état de la partie pour le joueur.

Souscriptions pour les Tentatives

`guessMade(gameSessionId: ID!): Guess`

- **Description** : Notifie le joueur lorsqu'une nouvelle tentative est enregistrée dans sa session de jeu.
- **Utilisation** : Mettre à jour l'affichage des tentatives en temps réel.