**Documentation du Backend**

**Description du Projet**

Ce projet est une API Django REST Framework qui gère des catégories et des tâches (todo). Les fonctionnalités principales incluent la gestion des catégories et des tâches avec des opérations CRUD complètes.

**1. Modèles**

**Category**

Représente une catégorie associée à des tâches.

* **Champs :**
  + name (CharField) : Nom de la catégorie, unique.
* **Méthodes :**
  + \_\_str\_\_ : Retourne le nom de la catégorie.

**Todo**

Représente une tâche individuelle.

* **Champs :**
  + title (CharField) : Titre de la tâche.
  + completed (BooleanField) : Statut de complétion (par défaut : False).
  + created\_at (DateTimeField) : Date de création (auto générée).
  + category (ForeignKey) : Référence à une catégorie (relation plusieurs-à-un).
* **Méthodes :**
  + \_\_str\_\_ : Retourne le titre de la tâche.

**2. Sérialiseurs**

**CategorySerializer**

Sérialiseur pour le modèle Category.

* **Meta :**
  + Modèle : Category.
  + Champs : Tous les champs (\_\_all\_\_).

**TodoSerializer**

Sérialiseur pour le modèle Todo.

* **Champs :**
  + id : Identifiant de la tâche.
  + title : Titre de la tâche.
  + completed : Statut de complétion.
  + created\_at : Date de création.
  + category : Sérialiseur imbriqué pour afficher les détails de la catégorie.
  + category\_id : Clé primaire de la catégorie (écriture uniquement).

**3. Vues**

**CategoryViewSet**

VueSet pour les catégories.

* **QuerySet :**
  + Toutes les catégories.
* **Serializer :**
  + CategorySerializer.

**TodoViewSet**

VueSet pour les tâches.

* **QuerySet :**
  + Toutes les tâches.
* **Serializer :**
  + TodoSerializer.

**4. URL**

Les routes sont définies dans urls.py en utilisant un routeur DefaultRouter.

**Endpoints :**

* api/categories/ : Endpoints pour gérer les catégories.
* api/todos/ : Endpoints pour gérer les tâches.

**5. Paramètres (settings.py)**

* **Applications installées :**
  + django.contrib.\* : Applications par défaut.
  + rest\_framework : Django REST Framework.
  + corsheaders : Gestion des en-têtes CORS.
  + todos : Application locale.
* **Middleware :**
  + Gestion de la sécurité, sessions, CORS, authentification, etc.
* **Base de données :**
  + Utilisation de SQLite par défaut.
* **Origines autorisées pour CORS :**
  + http://localhost:3000 (pour le développement).

**7. Fonctionnalités de l'API**

**Endpoints des catégories :**

* **GET** : Liste toutes les catégories.
* **POST** : Crée une nouvelle catégorie.
* **PUT/PATCH** : Met à jour une catégorie existante.
* **DELETE** : Supprime une catégorie.

**Endpoints des tâches :**

* **GET** : Liste toutes les tâches.
* **POST** : Crée une nouvelle tâche.
* **PUT/PATCH** : Met à jour une tâche existante.
* **DELETE** : Supprime une tâche.

**Documentation du Frontend**

**1. Description du projet**

Le frontend est une application React utilisant Redux pour gérer l'état. Elle consomme une API backend (Django REST Framework) pour gérer des **catégories** et des **tâches (todos)**. Le projet est structuré pour faciliter la création, la mise à jour, et la suppression de ces entités.

**2. Structure du Projet**

**Composants Principaux :**

1. **Main.jsx :**
   * Composant principal qui regroupe l'interface utilisateur pour :
     + Afficher les listes de catégories et tâches.
     + Ajouter ou modifier des catégories et tâches.
     + Supprimer des éléments.
2. **App.js :**
   * Fait le routage global et configure l'application React.

**Services HTTP :**

1. **categories.services.js :**
   * Contient des fonctions pour interagir avec l'API des catégories (CRUD).
2. **todos.services.js :**
   * Contient des fonctions pour interagir avec l'API des tâches (CRUD).
3. **http-common.js :**
   * Configure les requêtes HTTP avec Axios.

**Actions Redux :**

1. **Catégories :**
   * fetchCategories, createCategory, updateCategory, deleteCategory.
2. **Tâches :**
   * fetchTodos, createTodo, updateTodo, deleteTodo.

**Store Redux :**

* Défini dans store.js, il intègre les reducers des catégories et des tâches.

**3. Fonctionnalités**

**Catégories**

* **Lister les catégories :**
  + Récupère les catégories du backend et les affiche dans une liste.
* **Créer une catégorie :**
  + Ouvre une boîte de dialogue pour entrer un nom de catégorie.
  + Envoie une requête POST au backend.
* **Modifier une catégorie :**
  + Ouvre une boîte de dialogue préremplie avec les informations existantes.
  + Envoie une requête PUT au backend.
* **Supprimer une catégorie :**
  + Supprime une catégorie spécifique via une requête DELETE.

**Tâches**

* **Lister les tâches :**
  + Récupère toutes les tâches et affiche :
    - Le titre.
    - Le statut (complété ou non).
    - La catégorie associée.
* **Créer une tâche :**
  + Ouvre une boîte de dialogue avec :
    - Un champ pour le titre.
    - Une liste déroulante pour sélectionner une catégorie.
    - Une case à cocher pour marquer la tâche comme complétée.
  + Envoie une requête POST.
* **Modifier une tâche :**
  + Identique à la création mais préremplie avec les données existantes.
  + Envoie une requête PUT.
* **Supprimer une tâche :**
  + Supprime une tâche via une requête DELETE.

**4. Points Clés de Main.jsx**

**États Locaux :**

* **Catégories :**
  + categories : Liste des catégories.
  + category : Objet catégorie pour création/modification.
* **Tâches :**
  + todos : Liste des tâches.
  + todo : Objet tâche pour création/modification.
* **Autres :**
  + message : Affiche des retours utilisateur (succès/erreur).
  + openCategoryModal, openTodoModal : Contrôle des modales.

**Utilisation de Redux :**

* useDispatch : Pour dispatcher des actions Redux.
* useSelector : Pour récupérer des données du store Redux.

**5. Fonctionnement des Modales**

**Modale de Catégorie :**

* Gère la création et la mise à jour des catégories.
* Champs :
  + Nom de la catégorie.

**Modale de Tâche :**

* Gère la création et la mise à jour des tâches.
* Champs :
  + Titre.
  + Catégorie (liste déroulante).
  + Statut (complété ou non).

**6. Services HTTP**

Les services HTTP (dans categories.services.js et todos.services.js) gèrent les requêtes vers l'API Django REST Framework. Exemple :

**Categories**