API DOCUMENTATION

Documentation de l'API Backend

Cette documentation décrit les endpoints et les méthodes de l'API backend pour gérer les catégories de notre application. Pour des soucis de longueur, nous allons voir comment nous gérons l'API des catégories comme exemple.

Endpoint

`/api/category`

Ce endpoint permet de récupérer, mettre à jour et supprimer une catégorie spécifique par son `categoryId`.

Méthodes

GET

Description

Récupère les détails d'une catégorie spécifique.

Requête

- **Méthode HTTP** : `GET`
- **URL** : `/api/category?categoryId={categoryId}`

Paramètres de la requête

- **categoryId** (query) : ID de la catégorie à récupérer (obligatoire).

Réponse

- **200 OK** : Retourne les détails de la catégorie.

```
Json
{
  "id": 1,
  "displayName": "Example Category",
  "uniqueSlug": "example-category",
  "displayRank": 1
}
```

- **404 Not Found** : Si la catégorie n'est pas trouvée.

```
json
{
  "error": "Category not found"
}
```

- **500 Internal Server Error** : En cas d'erreur serveur.

```
json
{
  "error": "Detailed error message"
}
```

PATCH

Description

Met à jour les détails d'une catégorie spécifique.

Requête

- **Méthode HTTP** : PATCH
- **URL** : `/api/category?categoryId={categoryId}`
- **Corps de la requête** : JSON contenant les champs à mettre à jour.

```
json
{
  "displayName": "New Category Name",
  "uniqueSlug": "new-category-slug",
  "displayRank": 2
}
```

Paramètres de la requête

- **categoryId** (query) : ID de la catégorie à mettre à jour (obligatoire).

Réponse

- **200 OK** : Retourne les détails de la catégorie mise à jour.

```
json
{
  "id": 1,
  "displayName": "New Category Name",
  "uniqueSlug": "new-category-slug",
  "displayRank": 2
}
```

- **500 Internal Server Error** : En cas d'erreur serveur.

```
json
{
  "error": "Detailed error message"
}
```

DELETE

Description

Supprime une catégorie spécifique.

Requête

- **Méthode HTTP** : `DELETE`
- **URL** : `/api/category?categoryId={categoryId}`

Paramètres de la requête

- **categoryId** (query) : ID de la catégorie à supprimer (obligatoire).

Réponse

- **204 No Content** : Si la suppression est réussie.
- **500 Internal Server Error** : En cas d'erreur serveur.

```
json
{
  "error": "Detailed error message"
}
```

Erreurs et Méthodes non autorisées

Requête

- **Méthode HTTP** : Autres méthodes que `GET`, `PATCH`, `DELETE`

Réponse

- **405 Method Not Allowed** : Si la méthode HTTP n'est pas autorisée.

`Method {req.method} Not Allowed`

Exemple de Code

Voici le code de l'API :

```
import { prisma } from "@/back/db"

export default async function handler(req, res) {
  const { categoryId } = req.query

  if (req.method === "GET") {
    try {
      const category = await prisma.category.findUnique({
        where: { id: parseInt(categoryId, 10) },
      })

      if (!category) {
        return res.status(404).json({ error: "Category not found"
})
      }

      res.status(200).json(category)
    } catch (error) {
      res.status(500).json({ error: error.message })
    }
  } else if (req.method === "PATCH") {
    const { displayName, uniqueSlug, displayRank } = req.body

    try {
      const updatedCategory = await prisma.category.update({
        where: { id: parseInt(categoryId, 10) },
        data: { displayName, uniqueSlug, displayRank },
      })
      res.status(200).json(updatedCategory)
    } catch (error) {
      res.status(500).json({ error: error.message })
    }
  } else if (req.method === "DELETE") {
    try {
      await prisma.category.delete({
        where: { id: parseInt(categoryId, 10) },
      })
      res.status(204).send()
    } catch (error) {
      res.status(500).json({ error: error.message })
    }
  } else {
    res.setHeader("Allow", ["GET", "PATCH", "DELETE"])
    res.status(405).end(`Method ${req.method} Not Allowed`)
  }
}
```

Cette documentation vous permet de comprendre comment utiliser les différents endpoints de l'API pour gérer les catégories de notre application.