Lista de endPoints que debería tener mi endPoint

Crear Tarea usando como parámetro Todos los campos (esto incluye las foráneas user y categoría)

Actualizar Tarea esto va a ser una actualización parcial de: (Nombre y/o descripción)
Solo estado o solo mover a otra categoría obtenida por ID de entre las categorías que posea el usuario.

Más fácil sería crear endPoints separados. Uno para actualizar name y/o description otro para actualizar el estado que simplemente lo cambie de 1 a 0 o de 0 a 1 cada vez que se llame con un ternario y ya está (puesto que el usuario también podrá descompletar tareas)

Finalmente otro endpoint para actualizar solo la categoría.

Se necesitará un controlador para obtener todas las tareas por categoría. Esto incluye las tareas con categoría null que deberán mostrarse en el apartado de tareas sin categoría. Este debería llamarse varias veces

También sería conveniente tener un controlador para obtener tarea específica por ID, porque seguramente vendrá bien para checkear privacidad y porque este tipo de controlador siempre viene bien, por ejemplo en este caso no solo para comprobar privacidad sino que si se muestran los detalles de la tarea más en profundidad.

Finalmente, un controlador para borrar la tarea.

En resumen:

getById

getByCategory

create

updateDetails

updateState

updateCategory

delete.

Al preguntar por mi planteamiento obtengo las siguientes respuestas.

getByCategory: Debe filtrar por el estado. También Esto podemos hacerlo mediante queryparams para no manchar excesivamente el endpoint.

Consideraciones de las que me he dado cuenta:
Las tareas se deben borrar en cascada.

Se debe indicar categoría -1 y luego hacer conversión a null para buscar sin categoría en el endpoint, pues getByCategory, como endpoint que precisa un número en el cuerpo, no puede recibir un null, así que podemos usar -1 que es una id que nunca existirá. (ya que son incrementativas)

Apuntes:
? Lista Final de Endpoints

Base URL: `/api/tasks`

Método	Endpoint	Descripción	Params/Body
`POST`	`/`	Crea nueva tarea (con categoría y usuario).	`{ name, description, category_id, user_id }`
`GET`	`/:id`	Obtiene detalles de una tarea específica (validando propiedad).	`id` en URL
`GET`	`/category/:category_id`	Filtra tareas por categoría (`-1` = sin categoría). Opcional: filtrar por estado.	`category_id` en URL, `?state=` (opcional: `0` o `1`)
`PATCH`	`/:id/details`	Actualiza **nombre y/o descripción**.	`{ name?, description? }` en body
`PATCH`	`/:id/state`	Alterna el estado (`0` ↔ `1`) con un toggle.	-
`PATCH`	`/:id/category`	Cambia la categoría (validando que pertenezca al usuario).	`{ category_id }` en body (acepta `null` para quitar categoría)
`DELETE`	`/:id`	Elimina la tarea (en cascada si es necesario).	`id` en URL

? Explicación de Decisiones Clave

1. **`GET /category/:category_id`**
   - Usa `-1` para tareas sin categoría (conversión interna a `null`).
   - Filtro opcional por estado vía query param (`?state=0`).

   http

- `GET /api/tasks/category/-1?state=0  // Tareas sin categoría + incompletas`

- **Actualizaciones separadas** (`/details`, `/state`, `/category`):
  - Más intuitivo que un `PATCH` genérico con lógica condicional compleja.
  - Ejemplo para toggle de estado:

    typescript

  - •
  - `@Patch(':id/state')`

```
async toggleState(@Param('id') id: number) {
  await this.taskService.toggleState(id);
  return { success: true };
}
```

- **Seguridad en `updateCategory`**:
  - Valida que el `category_id` pertenezca al usuario o sea `null`:

    typescript

```
if (category_id !== null &&
!userCategories.includes(category_id)) {
  throw new ForbiddenException('Categoría no válida');
}
```

- **Delete en cascada**:
  - Configúralo en tu entidad (TypeORM/Prisma):

    typescript

```
@ManyToOne(() => User, { onDelete: 'CASCADE' })
user: User;
```