ProjectManager | MERN

Lógica de CRUD de proyectos

1. `.list()`: Listar todos los proyectos que pertenezcan al usuario autenticado

```
const projects = await Project.find().where("createdBy").equals(req.user);

return res.status(200).json({
    ok: true,
    projects
})
```

2. `.store()`: Crear un nuevo proyecto y vincularlo al usuario autenticado

```
const { name, description, client } = req.body;
if (
    [name, description, client].includes("") ||
    !name ||
    !description ||
    !client
) {
    throw createHttpError(
        400,
        "El nombre, la descripción y el cliente son datos obligatorios"
    );
}

if (!req.user) throw createHttpError(401, "Error de autenticación");

const project = new Project(req.body);

project.createdBy = req.user._id;
const projectStore = await project.save();

return res.status(201).json({
    ok: true,
    project: projectStore
})
```

3. `.detail()`: Traer un proyecto verificando que sea del usuario autenticado

```
const { id } = req.params;

if (!require("mongoose").Types.ObjectId.isValid(id)) throw createError(400, "No es
```

```
un ID válido");

const project = await Project.findById(id);

if (!project) {
    throw createError(404,"Proyecto no encontrado");
};

if (req.user._id.toString() !== project.createdBy.toString()) {
    throw createError(401,'No tenés la autorización para ver este proyecto');
};

return res.status(200).json({
    ok: true,
    msg: "Detalle del Proyecto",
    project,
});
```

4. `.update()`: Actualizar un proyecto

```
const { id } = req.params;
if (!require("mongoose").Types.ObjectId.isValid(id)) throw createError(400, "No es
un ID válido");

const project = await Project.findById(id);

if (!project) {
    throw createError(404, "Proyecto no encontrado");
}

if (req.user._id.toString() !== project.createdBy.toString()) {
throw createError(
    401,
    "No tenés la autorización para actualizar este proyecto"
    );
}

const { name, description, client, dataExpire } = req.body;

project.name = name || project.name;
project.description = description || project.description;
project.dateExpire = dateExpire || project.dateExpire;
project.client = client || project.client;

const projectUpdate = await project.save();

return res.status(201).json({
    ok: true,
    project: projectUpdate,
});
```

5. `.remove()`: Eliminar un proyecto

```
const { id } = req.params;
if (!require("mongoose").Types.ObjectId.isValid(id)) throw createError(400, "No es
un ID válido");

const project = await Project.findById(id);

if (!project) {
    throw createError(404, "Proyecto no encontrado");
}

if (req.user._id.toString() !== project.createdBy.toString()) {
    throw createError(
        401,
        "No tenés la autorización para eliminar este proyecto"
    );
}

await project.deleteOne();

return res.status(201).json({
    ok: true,
    msg: "Proyecto eliminado con éxito.",
});
```