Desarrollo Backend con NodeJS

Combo de modelos y manejos de archivos con Multer

course[It]

Combo de modelos

A veces para solicitar datos de un modelo, vamos a necesitar previamente contar con datos de otro modelo, por ejemplo:

- Imaginemos que tenemos dos collections, Users y Hobbies.
  - Cada User tiene como dato name y hobby
  - cada Hobby tiene como dato name y difficulty

Modelos

```
const mongoose = require("mongoose");

const userSchema = mongoose.Schema({
  name: {
    type: Number,
    required: true,
  },
  hobby: {
    type: String
  },
});

module.exports = mongoose.model("User", userSchema);
```

```
const mongoose = require("mongoose");

const hobbySchema = mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  difficulty: String,
  required: true
});

module.exports = mongoose.model("Hobby",
hobbySchema);
```

Imaginemos entonces que nosotros queremos devolver la siguiente frase desde un endpoint:

```
`El usuario ${user.name} tiene como hobby ${user.hobby}. Este es un hobby de dificultad ${hobby.difficulty}`
```

En una base de datos de tipo relacional podríamos obtener la data con un join, pero mongo no tiene relaciones entre collections.

Para lograr esto tenemos que hacer dos queries distintas, una a la collection users, y otra a la collection hobbies, todo esto desde un sólo controller...

En las rutas

```
const express = require("express");
const router = express.Router();
const UserController = require("./../controllers/userController");
const UserService = require("./../services/userService");
const HobbyService = require("./../services/hobbyService");
const UserInstance = new UserController(new UserService(), new HobbyService());
```

En el controller

```
class UserController {
  constructor(userService, hobbyService) {
    this.userService = userService;
    this.hobbyService = hobbyService;
  }

 async getUserById(req, res) {
    const { id } = req.params;
      const user = await this.userService.getUserById(id);
      const hobby = await this.hobbyService.getHobbyByName(user.hobby);
      const data = {
        name: user.name,
        hobby: user.hobby,
        difficulty: hobby.difficulty,
      };
      res.json(data);
  }
```

Services

```
const User = require("./../models/userModel");

class UserService {
  getUserById(id) {
    const query = User.findOne({ _id: id });
    return query;
  }
}

module.exports = UserService;
```

```
const Hobby = require("./../models/hobbyModel");

class HobbyService {
  getHobbyByName(name) {
    const query = Hobby.findOne({ name: name
}).exec();
    return query;
  }
}

module.exports = HobbyService;
```

Guardando archivos con Multer

Multer es una librería Multer que permite manejar el envío de archivos a nuestro servidor. Por ejemplo: imágenes o pdfs

- https://www.npmjs.com/package/multer
- npm install multer

Configurando Multer en index.js

```
const multer = require("multer");
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "./uploads");
  },
  filename: function (req, file, cb) {
    cb(null, file.fieldname + "-" + Date.now() + ".png");
  },
});

const upload = multer({ storage: storage });
```

Utilizando Multer en una ruta como middleware

- Middleware: cualquier función que se ejecuta antes de que recibamos la request y la modifica

```
router.post("/user", upload.single("avatar"), function (req, res) {
  console.log(req.body, req.file);
  return res.json({
    error: false,
    file: req.file,
    body: req.body,
  });
});
```