_nology

TALENT IN **TECH**NICOLOUR

Node Project

Overview

You're going to build a local node API that can connect to a local MySQL database!

This will solidify the concepts we've covered so far, built using these technologies:

- Node.js
- Express
- MySQL

Todo's

The API will have several different endpoints and will be able to **CRUD** to a local database.

The project is based around managing personal tasks. A Todo list 😍.

If you are confident and have your own idea create it! Just make sure to cover the stages below. 😎

Setup

Create your Github Repository:

- Your project should resemble the project structure to the side.
- `node_modules` should be ignored
- `your-*.js` files should match whatever you decided your project to based on.

```
|- controllers/
|  |- your-controller.js
|- db/
|  |- index.js
|- models/
|  |- your-model.js
|- routes/
|  |- your-routes.js
|- .gitignore
|- index.js
|- package.json
|- package-lock.json
|- README.md
|_ node_modules
```

This is the basic skeleton structure if you want to add more to it, go ahead! 📚

Data Structure

I would advise you to think of your data structure first, think that your....

- Todo will need to have the following to complete the challenges below.
  - id, title, priority
- What else could a Todo have?
  - textBody, completionTime, createdBy, createdOn etc...
- Have an idea first of what your data will look like before you start coding.

Stage 1:

Skeleton

Once you have the project setup and have an idea of the data you are modelling. Start by setting up your API.

- Create your `index.js`
  - You will need to configure your express app.
  - Will you need to add any dependencies?
- Set up your DB:
  - Set up and connect to your db in `db/index.js`
  - Create your data model in `models`
- Set up your `routes/your-route.js` and `controller/your-controller.js`
  - Create a `GET` request to `/todos`
  - This will get all of your Todo's from your database.

Stage 2:

CRUD

Time to add more endpoints to your API so you can implement different HTTP request methods.

- Create a endpoint for `GET /todos/:todoId`
  - This should return the todo with a matching id.
- Create a endpoint for `DELETE /todos/:todoId`
  - This should delete the todo with a matching id and return the remaining todo's.
- Create a endpoint for `POST /todos/`
  - This will take a new todo in the body of the request and create a new todo in the database.
- Create a endpoint for `PUT /todos/:todoId`
  - This will take a new todo in the body of the request and update the todo that matches the id in the db.

Extension:

Params

Time to start customizing the results you get back from your `GET /students` request. Update your controller to start to use query's.

- Search for todos that match the given query text
  - `/students?title=searchtext`
- Return all of the todo's ordered by there priority either ascending or descending
  - `/students?priority=asc`
  - `/students?priority=desc`

In what other creative ways can you query your data... Pagination? By Date? 🔍

Ready, Steady, Go!!!

We'll review how everyone is progressing with this challenge, but for now....

Good luck!