



Project: Mongo API

It's time to level-up your APIs and start using a database to store and retrieve data from and use that data to produce a RESTful API.

Technigo/project-mongo-api

Replace this readme with your own information about your project. Start by briefly describing the assignment in a

 <https://github.com/Technigo/project-mongo-api>



If you followed the last project, you were tasked with using a set of data (either one you made yourself, one we provided, or one you downloaded from a site such as [Kaggle](#).)

This project follows on from that, but this time you should model your database using Mongoose models, and persist your data in the database.

As before, it's up to you to decide what sort of data you'd like to store in your database and return from your API endpoints. In the repository, we've included some datasets you can use if you'd like:

- Golden globes nominations 2010-2019
- 500 book reviews
- Avocado sales and average prices from a selection of US states
- 1375 Netflix titles and some data about them
- 50 popular Spotify tracks and some data about the music type

If you'd like to build your own data set - feel free! You could write it yourself, or find data on a site such as [Kaggle](#). We used Kaggle to get the datasets above by downloading CSV files from datasets, renaming columns to be more JSON friendly, and then using a service to [convert from CSV to JSON](#).

In order to get all this data into your database, you'll need to write some code to generate the data - see the 'Making seed data' section further down for tips on how to do this.

Once you have the data stored, you will need to write appropriate RESTful endpoints to return the data and make use of [Mongoose Queries](#) to find and return the correct data given the route and any filter params passed.

What you will learn 🧠

- How to model data in Mongoose
- How to fetch items from a Mongo database using Mongoose
- How to seed large amounts of data to a database

How to get started 💪

1. Fork [the repo](#)
2. Clone the repo into your projects folder on your computer
3. Open up VS Code and start coding!
4. Install dependencies: `npm install`
5. Start the development server: `npm run dev`

Making seed data 🐡

[Seeding a database](#) is a process in which an initial set of data is provided to a database when it is being installed or set up. In the videos for this week, we showed a way to generate a small amount of data, but in this project, some of the JSON files have thousands of rows, so we don't want to have it all in our server file.

Instead, we can load the JSON and iterate over it to generate a lot of entries in our database. Let's say we had a file called `people.json`, and a `Person` model and the person model has exactly the same property names as the objects in the JSON file. Then we could write something like this in our server file to seed the database from the JSON:

```
import data from './people.json' const Person =
mongoose.model('Person', { // Properties defined here match the
keys from the people.json file }) if (process.env.RESET_DB) { const
seedDatabase = async () => { await People.deleteMany({})
data.forEach((personData) => { new Person(personData).save() }) }
seedDatabase() }
```

This whole process can be a little tricky and relies on your models having the same keys. Give it a try and if you run into issues, don't hesitate to ask your team, ask in slack, or post your code on stack overflow to get some help!

How to hand in the code 🎯

- When you're finished with the project, push your code to GitHub with these commands:




```
git add . git commit -m "your commit message" git push origin
master
```

- Navigate to your repo and create a Pull Request into the Technigo repo
- Wait for the code review

How to get help SOS

Ask for help and share your knowledge about this project with the 'project-mongo-api' tag on [Stack Overflow](#). Talk to your team on Slack and help each other out. Do some research about your problem, you are surely not the first one with this problem, Google is your friend 😊. And you can of course also reach out to your teachers.

Requirements

Your project should fulfill the  **Blue Level** and all of the **General Requirements**. Use the  **Red Level** and  **Black Level** to push your knowledge to the next level!

General Requirements


- Contribute by helping others with this project on Stack Overflow.
- If selected; demo your solution for your team.
- Code follows Technigo's code guidelines:

 [Copy of Guidelines for how to write good code](#)

- Your API should be deployed to Heroku or similar hosting service.
- Your database should be deployed using mongo cloud or similar.

Blue Level (Minimum Requirements)

- Your API should have at least 2 routes. Try to push yourself to do more, though!
- A minimum of one endpoint to return a **collection** of results (array of elements)
- A minimum of one endpoint to return a **single** result (single element).
- Your API should make use of Mongoose models to model your data and use these models to fetch data from the database.
- Your API should be RESTful

 Make sure you've committed and pushed a version of your project before starting with the intermediary and advanced goals.

Red Level (Intermediary Goals)

The Red Level requirements from the previous project can be applied here as well.

- If you are doing any sort of manipulation after retrieving the data from the database. Try using mongoose to do these operations instead.
- Accept filters via query parameters to filter (via mongoose) the data you return from endpoints which return an array of data.

● Black Level (Advanced Goals)

The Black Level requirements from the previous project can be applied here as well.

- Try implementing 'pages' using `.skip()` and `.limit()` (instead of `.slice()`) to return only a selection of results from the array. You could then use a query parameter to allow the client to ask for the next 'page'.
- Try to create an endpoint that uses mongoose's aggregate function which allows you to use the MongoDB aggregate pipeline. This is super-useful when doing more complex operations on database data, and a lot faster!

💡 🚩 Don't forget to add, commit and push the changes to GitHub when you're done. 🏁