# 🚝 Build an API using Express

> This week's project is to start your backend journey by creating an API using Express. Your API should have at least a couple of RESTful endpoints which return either an array of data, or a single item.

Technigo/project-express-api
Replace this readme with your own information about your project. Start by briefly describing the assignment in a sentence or two. Keep it short and to the point. Describe how you approached to problem, and what tools and techniques you used to
https://github.com/Technigo/project-express-api

Since we're not using databases yet, you'll need to use a hard-coded set of data, stored as a JSON file in your project. The API endpoints you create will use this file along with array methods such as `.find()`, `.filter()`, and `.slice()` to select, filter, or limit the items returned in your responses.

It's up to you to decide what sort of data you'd like to 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 (used in this week's codealong)
- 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.

## What you will learn 🧠

- How to build an API in Node using Express
- How to create routes in Express
- Practice data manipulation in JavaScript - selecting, filtering, and limiting arrays

## 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`

## Example routes 🧭

RESTful routing can take a little time to get used to. Try to think of what the plural term for your data is and use that as a starting point. For example, if you're building an API using the Spotify data in the repo, each item in the data is a track - or song. So a pluralized route for this would be `/songs`. Your routes could then look like this:

`/songs` - Returns an array of songs

`/songs/5` - Returns a single song whose ID is '5'

## 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 🆘

Ask for help and share your knowledge about this project with the 'project-express-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 via the 1:1 sessions.

## 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.

#### 🔵 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 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)

- On routes which return a single item, handle when the item doesn't exist and return some useful data in the response.
- Accept filters via query parameters to filter the data you return from endpoints which return an array of data.
- Create some empty/dummy endpoints which could contain more complex operations in the future. Find good names for them (think back to the labs)

#### ⚫ Black Level (Advanced Goals)

- Build a frontend which uses your API in some way to show the data in a nice way (use the react-starter template to get up and running fast).
- If your dataset is large, try implementing 'pages' using `.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'.
- Create useful documentation for your endpoints. What's a good way to present this documentation? What if it changes in the future? Are there any npm packages that could help with this?

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