Back End 2

For todays lab you will be building a realty application. Your application should be able to add a new house, including an address, a price, and an image url. You should be able to see all the houses in the database, as well as edit the prices (+- $10,000). You should also be able to delete houses from the database.

Attempt to build out the application by following the instructions below. If you are stuck, take a look at the solution code and see if you can work out what is going on. If you cannot, make sure to reach out in the queue.

Instructions

Part 1 - Setup

In your server folder, create a file called **index.js**.
Setup the basic functionality of your index.js file (express, cors, json, app.listen).
Create another file called **controller.js** and set a module up with the following function signatures: `getHouses`, `deleteHouse`, `createHouse`, `updateHouse`.

Part 2 - Setting Up Endpoints

Create a variable that requires your controller file.
Setup an endpoint to get all houses, making sure to call you controllers `getHouses` function.
Setup an endpoint to create a house, making sure to call you controllers `createHouse` function.
Setup an endpoint to update a house, making sure to call you controllers `updateHouse` function. Note: It should require an id param in order to allow for you to update the correct house.
Setup an endpoint to delete a house, making sure to call you controllers `deleteHouse` function. Note: It should require an id param in order to allow for you to delete the correct house.

Part 3 - Adding Endpoint Functionality

Outside of `module.exports`, create a variable that requires your db.
Outside of `module.exports`, create a variable to keep track of your upcoming house id, and set it equal to `4` (we already have 3 houses in the database with id's 1, 2, and 3).
Build out the functionality of your `getHouses` function. It should send all the houses in the houses database to the front-end.
Build out the functionality of your `deleteHouse` function. It should find the index of the house in the houses database whose id correlates to the id passed in as a parameter on your endpoint (I recommend you use the `findIndex` method). Once you have the index, you can delete that house from the database using the `splice` method. Once done, send the remaining houses to the front-end so that the view can be updated.
Build out the functionality of your `createHouse` function. It should create a new house object with the following properties: `id`, `address`, `price`, and `imageURL`. All of those values should come from `req.body`, except for `id`, which will come from the variable you created above to keep track of your upcoming house id. Once you have created the new house object, add it to your houses database using the `push` method. Then send all your houses to the front-end so that the view can be updated to include your new house.
Don't forget to increment your variable tracking your upcoming house id.
Build out the functionality for your `updateHouse` function. It should capture the `id` from your endpoints params so that you know which house to update. It should also capture `type` off of `req.body`. Type is a string and could either be 'plus' or 'minus'. Next, find the index of your house in the houses array by iterating through the houses array and locating the house with the correct id (I recommend the `findIndex` method). Once you have the index of the house you should be updating, use a sequence of conditional checks to see if the type is 'minus' or 'plus', and then decrease or increase the price of the hosue by $10,000. Once complete, send all the houses to the front-end so you can update the views.

Submit

Initialize your work as a git repo and push the repo to your github account.