mochki

April 3, 2017

Windo Server & API

I chose to do an apparently unconventional project as most of the other students chose to work on projects at a lower networking level and the bulk of mine was at an application level. I saw nothing in the assignment details that specified against such and I thought it would be a great opportunity to learn some valuable skills and make something anyone could interface. I have finished what I have outlined I would for my final project and have included the files and a readme file for execution, though it takes some setup. I will leave the details in there.

I learned a lot about newer technologies that are sort of taking the world by storm. Node.js is being widely adopted by enthusiasts and corporations alike and I often see jobs for 'full stack javascript developers' base on the Node.js stack. I was able to learn and get comfortable in the Node.js ecosystem as well as improve my JavaScript and start using ES6 standards and emerging technologies there. That includes frameworks such as Express and Mongoose (with MongoDB). Though technically separate from my final project, I wanted to get this running on AWS so I spent a decent amount of time learning about that, which will again be a helpful skill. I hope to have pushed this onto my instances running there so that my API can actually be interfaced with at windoapp.xyz. I also spent a lot of time in WebStorm (an IDE based around web development) and Postman (a program to test an API), both of which I knew little about previously.

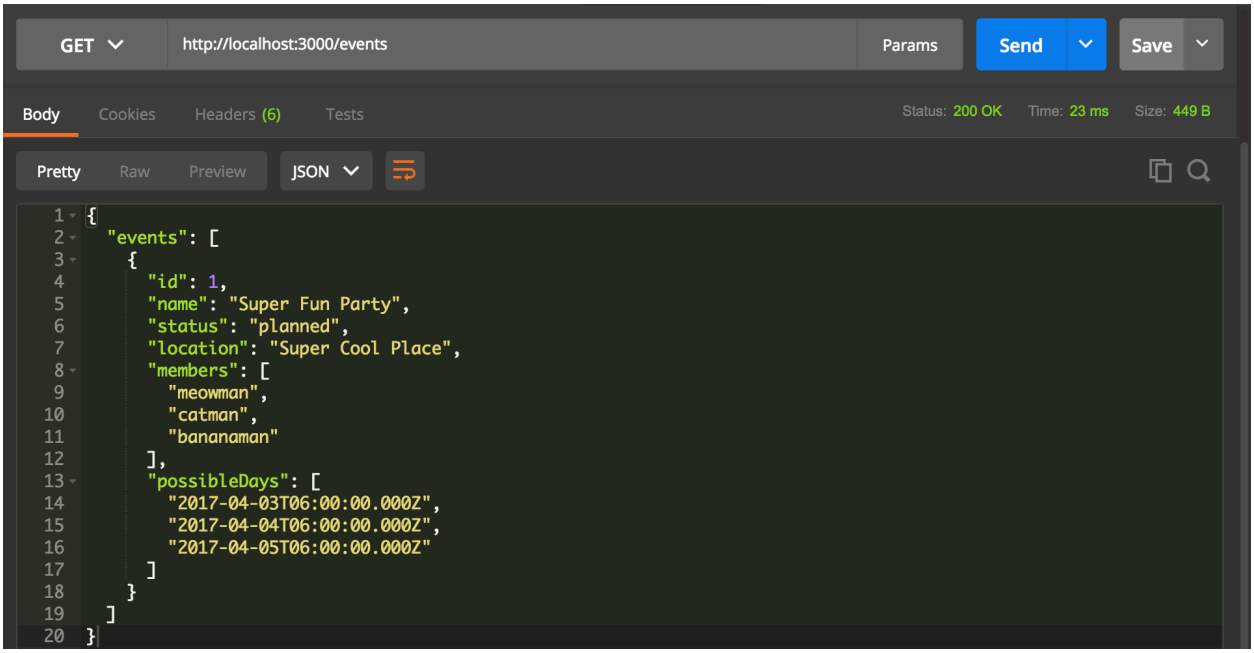
It's hard to sum up how much time I've spent altogether working on this project but I do know I spent a solid 12 hours this last week sitting in front of my computer working in WebStorm. I would estimate a solid 4-5 more hours of actual work on the project scattered throughout the semester.

I would definitely note the size requirements of the project to future students. Perhaps I missed a little at the beginning of the semester, but I did anticipate this to be a far larger project and most other student's projects were smaller than I expected. But the real advice I have for students is to pick a project they would have fun with. It makes working on a project that much easier.

I would give myself a solid 90% on the project. There's definitely more I could've done and the nature of using the Node.js stack is that you can have a web server running in mere seconds so it takes a little away from the difficulty of my project but I actually wanted to understand what my server was doing.

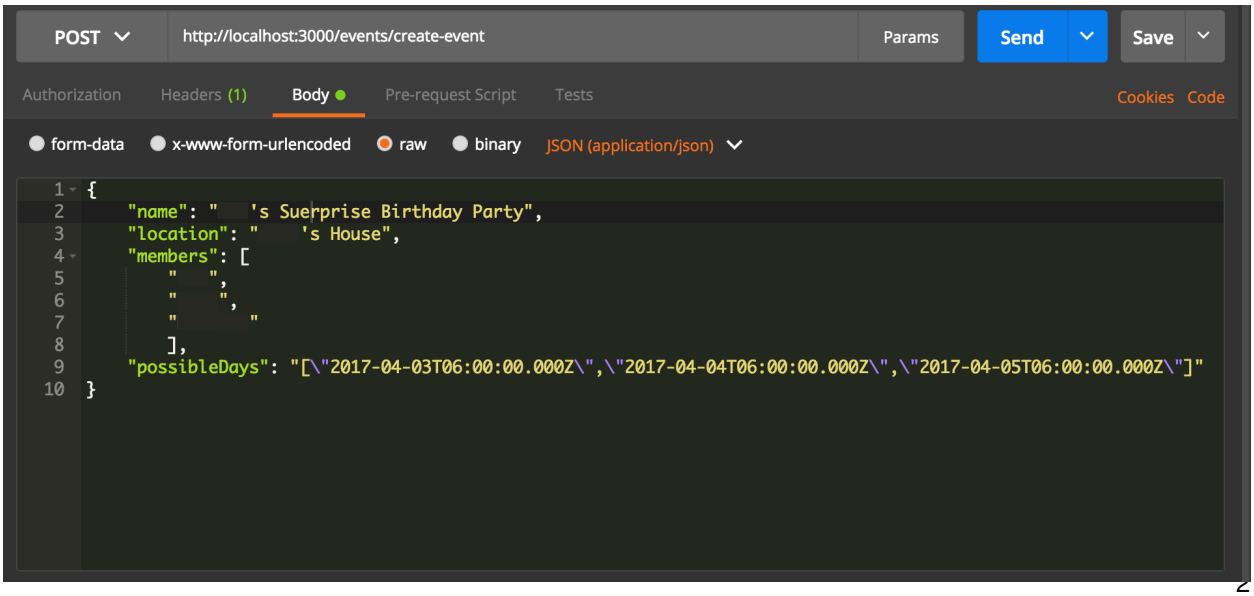
I will include some screenshots of using my api, just in case the server can't be set up or you didn't want to install node and npm on your computer. Once the server is running, I used Postman to make all the queries, sending just JSON data. These are also the effective 'API Docs' defining what fields are required with what data.

GET /events - we receive all events in a JSON



POST /events/create-event - to create a new event:

"name" : String
"location" : String
"members" : JSON.stringify(Array of Strings)
"possibleDays" : JSON.stringify(Array of Dates)



We now have the following in our database:

GET /events/:eventID - to get specific details of an event

replace ":eventID" with the ID

POST /events/:eventID - to change the following specific details of an event

"id" : Number
"name" : String
"location" : String

GET /events/:eventID/time-submit - effectively identical to /events/:eventID, just returns event information in preparation to send a time-submission

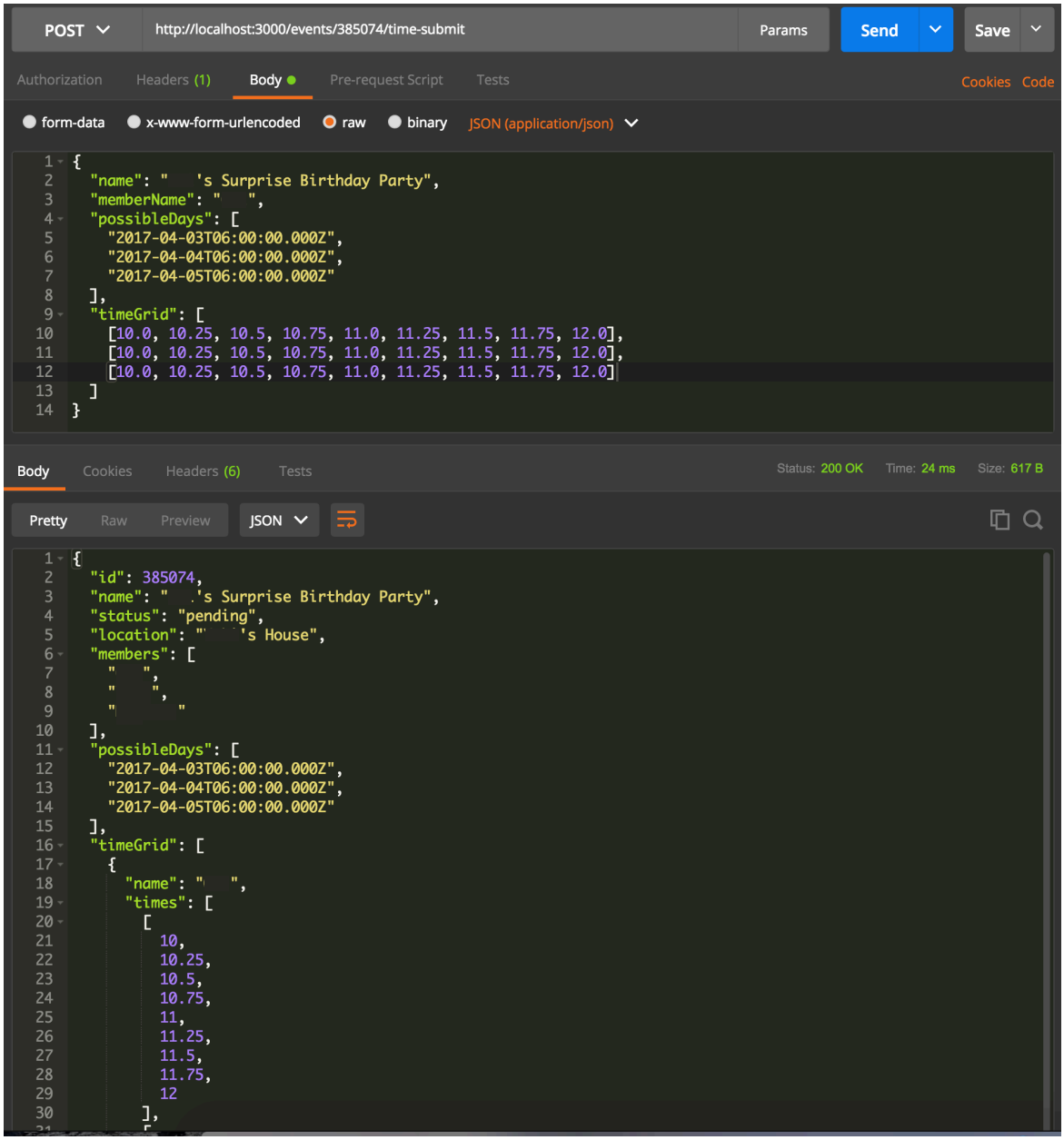
replace ":eventID" with the ID

POST /events/:eventID/time-submit - Each user needs to send in their available times. A lot of information is required to be sent to ensure the proper event it being submitted to.

"name" : String
"memberName" : String
"possibleDays" : JSON.stringify(Array of Dates)
"timeGrid" : [Arrays of available times]

This requires a little bit of explanation. The API requires that the memberName who is submitting this is one of the 'members' listed in the event in the database, otherwise we get an error.

The other tricky one is ‘timeGrid.’ This will be an array of arrays, one array for each possible days. That means the number of arrays must equal the number of days listed, otherwise the API will throw an error. What goes in these arrays are just numbers that go from 0 to 23.75. 6.25 refers to the 6:15 to 6:30 time slot. 6.75 refers to the 6:45-7:00 time slot. We use a 24-hour clock for simplicity’s sake. Here are some examples submission:



Once all users have submitted their available times, the status on the event changes from ‘pending’ to ‘planned’ and a ‘time’ key is added to the event the shows all the possible times. The event as a whole is returned at this endpoint.

ELSE - all other HTTP methods to any other endpoint returns a JSON with an error message.