



19floresa /
chatAPI



Code

Issues

Pull requests

Actions

Projects

Security

Insights

Settings



☆ **0** stars **0** forks **1** watching **1** Branch **0** Tags Activity

Private repository

main

1 Branch

0 Tags

Go to file

t

Go to file

+

Add file

Code

19floresa

modified readme; small bug

16 minutes ago

client

Incorporated SocketIO Changes to All ...

35 minutes ago

database

added information to set up db

3 hours ago

dict

added _id to specific users

19 minutes ago

middleware

Incorporated SocketIO Changes to All ...

35 minutes ago

node_modules

Created Chat Component Layout

3 weeks ago

routes

Incorporated SocketIO Changes to All ...

35 minutes ago

.gitignore

new thing to ignore

last week

README.md

modified readme; small bug

16 minutes ago

app.js

Incorporated SocketIO Changes to All ...

35 minutes ago

package-lock.json

Incorporated SocketIO Changes to All ...

35 minutes ago

package.json

Incorporated SocketIO Changes to All ...

35 minutes ago

README

chatAPI

0. set up database

- `mkdir data && cd data && mkdir db && cd ..`
- Assumption1: You have the most recent mongodb and tools installed and working.
- Assumption2: The mongo url is as follows: 'mongodb://localhost:27017
- Assumption3: A mongo database called `chat-db` exists.
- To find more information about the database set up then read the file in `./database/db.js`

1. Import Libraries

To import all the requires libraries you will need to call `npm i` in the following directories:

- `./`
- `./client/`

2. RUN db

- `mongod -dbpath data/db --port 27017`

2.5. Import data to db

- `cd dict`
- `mongoimport --db chat-db --collection users_info --file userData.json --jsonArray`
- `mongoimport --db chat-db --collection users_messages --file messageData.json --jsonArray`
- `cd ..`

3. Run backend code in './'

- `node app.js`

4. Run frontend code in './client/'

- `npm run start`
- In a different web browser tab open the same webpage for user2 (i.e <http://localhost:3000/>)
- If the url does not work then copy and paste the http link from the first tab

4.5. Test 1: Test imported data

- The current page should be log in page (i.e. <http://localhost:3000/>)
- Open another tab with the same webpage as above.
- Log in to user1 and user2 in each tab.
- user1: username: batman , password: 123
- user2: username: Ricky , password: 456
- In user1 tab, go to ricky and you should see the messages you imported
- In user2 tab, go to batman and you should see the messages you imported

5. Register a user with the GUI

- In the first Login screen click register in the bottom of the input boxes.
- Then, register 2 users with the username and password below.
- User1: 19floresas , password1: 123
- User2: xxx_cool_user_xxx , password2: 123
- Save both usernames and passwords!

6. Test 2. Log in with new users

- Go back to the login page.
- Enter data of user1 in the first tab.
- Repeat for user2.

7. Test 2.1. Communicate with user2

- In the web browser tab for user1, find user2.
- Type a message in the bottom of the webpage and send your message to user2.
- Repeat for user2.
- In both tabs, you should see user1 and user2 message.





Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors 2


-  **19floresa** Alexander Flores
-  **Coby-Do** Coby Do

Languages

JavaScript 91.8% HTML 5.3% CSS 2.9%


Suggested workflows

Based on your tech stack

**Webpack**


Configure

Build a NodeJS project with npm and webpack.

**Grunt**

Configure

Build a NodeJS project with npm and grunt.

**Publish Node.js Package**

Configure

Publishes a Node.js package to npm.

[More workflows](#)

[Dismiss suggestions](#)