Slyce Worktest

Your task is to design and implement data schemas and corresponding HTTP server endpoints for a professional athlete hosting a Question & Answer session with fans.

Using an HTTP server and data store of your choosing, please implement the REST-like endpoints listed below.

Each task details a minimum set of required data attributes, add more as you deem necessary.

The routes in the task list are only suggestions, modify them however you see fit.

Be prepared to discuss any benefits, limitations, and tradeoffs of your design after the exercise.

This webpage is meant to help you test your implementation, feel free to make modifications in order to get it talking with your server.

By default, this webpage expects an HTTP server running on **http://localhost:8080** that accepts and returns all data in JSON format.

It also attempts to make requests to the example routes, so if you make changes to the routes, you will also need to change the JavaScript on the page code.

Feel free to ask us any questions you may have.

When you are finished, please email your source code (including your version of this HTML file if you made any modifications) along with directions for launching the server and data store to dian@slyce.io (mailto:dian@slyce.io) or give us a link to your GitHub.

Tasks

*Note: All data models are expected to have some sort of **id** field. All user models must have at least a **name** attribute.

- 1. Create a QA session. QA sessions must have a **host** user, a **start_time**, and an **end_time**.

  `POST /qa`

- 2. Retrieve a QA session.

  `GET /qa/:qa_id`

- 3. Ask a question. Questions are within the context of a single QA session and must contain **text** and an **asked_by** user.

  `POST /question/:qa_id`

- 4. Answer a question. Can assume one answer per question. An answer can allow for either **text** and/or **image_url** as a response. It must also have an **answered_by** user.

  `POST /answer/:question_id`

- 5. Retrieve a list of questions for a given QA session. Allow for the option of filtering answered vs. non-answered questions.

  Each question in the result list must contain all relevant expanded models. IE: the asked_by user, the answer (if there was one), and the answered_by user.

  `GET /qa/:qa_id/questions`

Testing

Form fields are sent to the server as JSON in POST requests. EX: the host_name and start_time fields are sent as {host_name: value, start_time: value}. ID fields may be sent as part of the URL depending on the endpoint. EX: qa_id = 1234 is included in the URL as /qa/1234

If you see 'Access control...' type errors in the browser developer console, read the comments in the JavaScript source code of this HTML page regarding 'CORS'.

1. Create Q&A Session

host_name | start_date | end_date | POST

2. Get Q&A Session

qa_id | GET

3. Ask a question

qa_id | text | asked_by_name | POST

4. Answer a question

question_id | text | image_url

answered_by_name | POST

5. Get Questions

qa_id GET