CSCI 330 Project Report

Isaac Boaz

November 28, 2023

1 Introduction

From the source code, we can observe that the front-end has no knowledge of the back-end implementation. For all it cares, the back-end could be throwing away the data and returning random values. The benefit in this case is our back-end could be a flat-file system, a mongodb database, an SQL server, or any other data system. Rather than being tightly coupled with implementation, the front-end interacts with the back-end using a RESTful API, with PUT, GET, DELETE, and POST requests.

Task Blueprint		
Field	Type	Description
_id?	string / null	The id of the task
text	string	The text of the task
day	string	The day the task is due
reminder	boolean	Whether or not to remind the user

- **CREATE** - *POST /tasks* Inserts a new **Task**
  Returns A JSON object under the result key with the **Task** with *_id* populated.

- **READ** - *GET /tasks/(id)* Fetches a specific **task** or (if unspecified) all **tasks**
  Returns A JSON array under the result key, if ID was specified then the task that matches it will be in the array, if no task matches, it will be empty.

- **UPDATE** - *PUT /tasks/<id>* Toggles the given task's reminder boolean
  Returns A JSON object under the result key with the updated **Task**.

- **DELETE** - *DELETE /tasks/<id>* Deletes the given task
  Returns A JSON object with result mapped to a boolean.

2 Implementation

The workflow for the front-end app is simple.

- When the application loads:
  1. The application sends a GET request to the back-end for all tasks.
  2. The application generates a list of tasks from the response.
- When a user wishes to make a task:
  1. The application gathers the necessary information. (Text, Day, Reminder)
  2. The application sends a POST request to the back-end with the information.
  3. The application maintains an internal list of tasks, and adds the new task to the list.
- When a user double clicks a task:
  1. The application sends a PUT request to the back-end with the task's ID.
  2. The application maintains an internal list of tasks, and toggles the reminder boolean of the task.
  3. The application updates the UI to reflect the change.
- When a user wishes to delete a task:
  1. The application sends a DELETE request to the back-end with the task's ID.
  2. The application maintains an internal list of tasks, and removes the task from the list.
  3. The application updates the UI to reflect the change.

3 Differences

- **Error Handling**: The MongoDB implementation will safely reject invalid requests (null / missing text or malformed data types) whereas the MySQL implementation will internally crash and requires the user to restart the server.
- **Metadata**: MongoDB returns additional metadata about each task, such as the *createdAt*, *updatedAt*, and $\_\_v$ fields.
- **Boolean Types**: MongoDB has a boolean type, whereas MySQL does not. MySQL uses an integer type to represent boolean values.
- **Security**: One additional consideration is that each type of backend handles injection / malicious input differently.
- **IDs**: MongoDB returns a 24-character ID per task created, whereas MySQL returns a 36-character UUID.