CentraleSupélec

Architectures Applicatives
TP 1 & 3

Micro-Service in Javascript with NodeJS

Students:
Daniel Stulberg Huf
Tomas Gonzalez Villarroel

Supervisor:
Benoît VALIRON

March 13, 2024

1 Overview

This micro-service serves as a simple messaging platform where users can write, send, and see messages. The server maintains a list of messages, the usernames associated with each message, and timestamps for when each message was sent. Clients can interact with the server through predefined HTTP endpoints to perform actions such as posting a new message, fetching all the messages, or deleting a specific message. The client-side application provides a user interface for these interactions. User can input and send messages, view a dynamically updated list of posted messages along with the sender's username and the message's timestamp, and even fetch a programming joke using an external API. The messages are displayed in reverse chronological order, with the most recent message appearing at the top of the page. The application uses AJAX calls for server communication, enabling asynchronous data exchange without needing to reload the web page.

2 Links

- **Replit project:**
  1. Server side: https://replit.com/join/thxysruptz-danielhuf
  2. Client side: https://replit.com/join/epjpbpurwa-danielhuf
- **GitHub repo**: https://github.com/danielhuf/cs-architectures-applicatives/tree/main/TP1-3
- **Render service**: https://meme-chat-client.onrender.com

Note: Since the service was deployed on a free instance of Render, the first request to the server might be delayed by 50 seconds or more.

3 Extensions

3.1 Usernames and timestamps

A feature has been introduced that allows adding a username to each posted message. If a user does not specify a username, "Anonymous" is automatically assigned and sent to the server. Along with the username, the corresponding timestamp of the message post is also sent. When messages are retrieved from the server, each timestamp is displayed alongside its corresponding message, enabling to trace back when each message was sent.

In order to keep compatibility with other students' APIs, the requirements of using a GET request and sending the message content through the URL weren't changed. In consequence, the username and timestamp were sent by adding a custom header to the request (`Username` and `Datetime`). This ensures compatibility while adding a new feature.

Additionally, when fetching all messages, all usernames and timestamps are sent in the header in the format of a stringified list. Messages received by the server that have no Username header are saved with the Anonymous username, while messages with no Datetime header will simply save the current time that the message was received, which ensures compatibility and generalization with regards to other students' client pages that may not send any headers in their requests.

3.2 Image API

To add more customization in a simplified way, a user image was added to each message. This image is obtained by using an API provided by Boring Avatars, which returns a unique avatar image generated by a given string. In this case, the username was the provided string, which gives the effect of each user having a unique profile picture.

3.3 Joke API

Another addition to the service on the client side was suggesting the user to send a programming joke, which is generated by JokeAPI. To do that, we use ‘fetch‘ to send a GET request to the API and we also filter jokes to the programming category, as well as specifying some flags. Once the request is completed, we process the response by reading its body to completion as plain text. After obtaining the joke text, we take this text and set it as the value of the ´textarea´ element representing the textual content of the message to be the sent.