WEB APPLICATION DEVELOPMENT USING NODEJS - 502070

LAB SESSION 5

By Mai Van Manh

OBJECTIVES

1. Review and consolidate knowledge about Rest API.
2. Utilize the http and node-fetch/fetch modules to make Rest API calls within NodeJS.
3. Explore the concept of Cross-Origin Resource Sharing (CORS).
4. Familiarize oneself with the concept of middleware and create a custom middleware for reading request bodies.
5. Process data in HTML forms using express-form and express-validator.
6. Implement flash messages for inter-route data transmission.
7. Mitigate DDOS attacks using rate limiting mechanisms (express-rate-limit).
8. Continuously apply previously acquired knowledge in areas such as Node.js, Express, etc., for the ongoing development of web applications.

ASSIGNMENT DESCRIPTION

Utilize Node.js and modules such as http, express, ejs, etc., to create a webpage displaying a list of users (with a user interface similar to last week's assignment). When clicking on a user, the webpage should transition to display detailed information about that user. Additionally, the webpage should offer functions like adding a new user, updating user information, and deleting users. Modal dialogs should be displayed for confirmation before updating or deleting data.

DATA REQUIREMENTS

- The webpage (on the Node.js side) won't directly store data but will use data from a provided Rest API.

- The Rest API is available at https://web-nodejs-502070-wiolshzi6q-uc.a.run.app/. Students should access the link above to view the description of the API endpoints.
- When users need to view user lists, add users, update users, or delete users, the Node.js backend needs to forward requests to the respective API endpoints below to perform corresponding functions.
- Technically, our Node.js now functions as a "forwarder," receiving requests from users and then forwarding these requests to the REST API. It receives the results and reports them back to the users of the webpage.
-

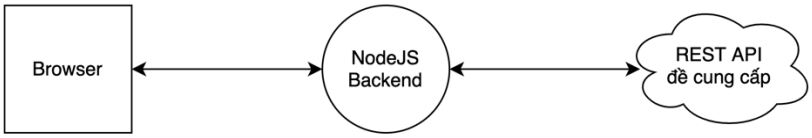
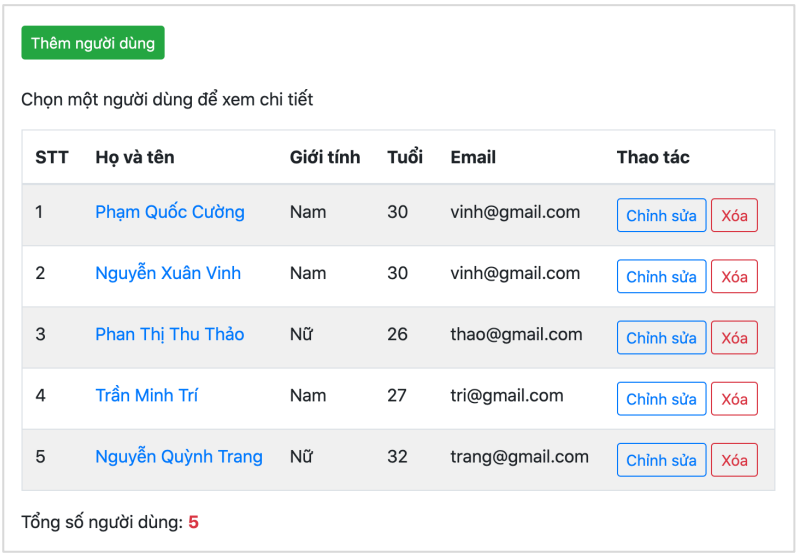


Illustration of components in the application's operational mechanism

The front-end can directly use ajax/fetch to call the REST API and fetch data without going through the Node.js backend. However, this can only be achieved if appropriate Cross-Origin Resource Sharing mechanisms are in place. By default, ajax/fetch can only read server-side data if the client and server share the same domain. Ajax/fetch requests from the front-end to a server with a different domain will be blocked by most browsers due to security reasons. In contrast, the Node.js backend is not controlled by this mechanism as it is the back-end, and therefore, it can freely read data from any API.



The homepage interface

Additional requirements:

- All data in HTML forms should be validated on the server side using middleware like express-form, express-validator, etc.
- In previous exercises, body-parser was used to read data from the request body. In this assignment, students should write their own body parser (middleware) without using externally installed modules.
- Use flash messages to transmit notifications between routes in the Express app.
- Utilize the rate-limiting feature in Express to prevent DDOS attacks.
- If users access any paths not described above, redirect them to the /error path while displaying a 404 not found error interface.