Edinburgh Napier UNIVERSITY

Coursework Assessment 2
CRUD Blog

Oliver Thornewill von Essen
40210534@napier.ac.uk
Edinburgh Napier University
Web Technologies (SET08101)

9 April 2018

Abstract

The goal of this project is to design and implement a simple blog platform. It will compromise server and client elements.

The client element must present a user interface, enabling at least one user to add a new blog post, to edit or view an existing blog post, and to delete an existing blog post. The server element will persist data related to the blog, will serve up the user interface, and will also provide a create, read, update, delete (CRUD) API that the client element will utilize in providing the blog's features.

This project must select and apply appropriate technologies aiding it to reach the goal. Specifically HTML, CSS & JavaScript must be used for the client interface, while Node.JS must be used on the server. Additional libraries including Angular.JS, Express, JSON Web Tokens, and BcryptJS have been used to further extend the blog's functionality.

Contents

1 Introduction

Introduction: A representational state transfer(REST) architectural website is "a set of coordinated constraints on designing components within a distributed hypermedia system"[1]. The scope of this project is to build a RESTful blog application hosted through Node.JS that will allow at least one user to Create, Read, Update, Delete(CRUD) blog posts. Among other constraints, the focus for this platform is that it is stateless and cache-able. Stateless constraint means that data will persist even in the event of a server restart, and cache-able means that data will not be recalculated where there are no changes. There will be a noSQL data store - MongoDB - including separate collections for blog posts and user accounts. The use of the data store enables a stateless mode for the application.

2 Software Design

2.1 The approach to creating this website

This blog application is going to make use of Node.JS modules which extend the functionality of HTML and therefore the application's functionality. This blog application is going to make use of dynamic web pages. Instead of a server simply waiting for GET requests to display static HTML, dynamic web pages allow the manipulation of data / files on the server such as POST or UPDATE requests. The client can trigger a function to be ran on the server who will generate a response to be sent back to the client [2]

2.2 Node Libraries Used

2.2.1 Angular.JS

AngularJS is a "structural framework for dynamic web apps". It enables templates to be created using HTML and further extents HTML's syntax allowing one to express the blog's components in a clear way. Not every app is suitable to be using Angular.JS as flexibility gets diminished through a higher level of abstraction (breaking down of components) to the developer. However, as AngularJS was produced for CRUD applications, it is suitable for this blog platform.[3]

2.2.2 Express

"Express is a light framework for Node.JS providing a robust set of features for web and mobile applications"[4]. Features include middle wares when responding to HTTP requests, routing tables based on the URL, dynamically render HTML pages [5].

2.2.3 JSON Web Tokens

JSON Web Tokens (JWT) is "an open standard (RFC 7519)[6] that defines a compact and self-contained way for securely transmitting information between parties as a JSON object". The use of digital signatures (public/private keys with RSA) ensure the verification of transferred information [7].

2.2.4 BcryptJS

The node module BcryptJS will be used in order to secure user account passwords. It does this through hashing, allowing one to avoid storing user account passwords in plaintext. Added security is implemented through the incorporation of salts when generating a password hash. This adds a layer of security, in the event of a database breach it is less likely for a rainbow table attack to be successful [8]

2.3 Sketches of design

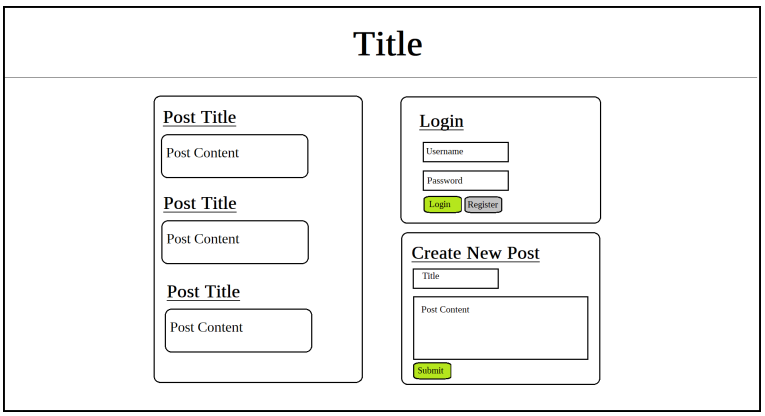

Figure 1: Initial sketch for home page

Everything should be based in one page. The borders around different sections make the different functionalities clear. The negative of this approach is that the page quickly becomes over populated and visually less appealing.

2.4 The requirements to be fulfilled

- At least one user:
  - Administrator User
    - Create posts
    - Update posts
    - Delete posts
  - Every User
    - Read posts
    - Comment on blog posts
    - Search through blog posts
- Consistent visual design such that the pages look similar despite varying in content.

3 Implementation

The produced blog application includes functionality for user registration and authentication as well as CRUD blog functionality. Unauthorized users are able to read the list of posted blog posts, however in order to create/update/delete a blog the user must be authorized. Authorization is enabled through the use of JWT allowing the browser to store an authentication key locally.

The use of AngularJS has made it possible to break down the application into different components. Among other examples, the login and registration forms, navigation bar, footer are all their own component. This has the advantage to make the application more modular which enables simpler addition of new functionalities. It has another clear advantage that code is only edited in one place. For example the navigation bar or page footer must only be edited once and the changes are applied to all pages within the application.

Instead of having the different functionalities in one page as proposed in Figure 1, a decision change was made to avoid this - separating the functionalities into their own pages. This made it possible to have a more streamlined appearance and also enables easier navigation with smaller window sizes for users with smaller displays. [9] [10]

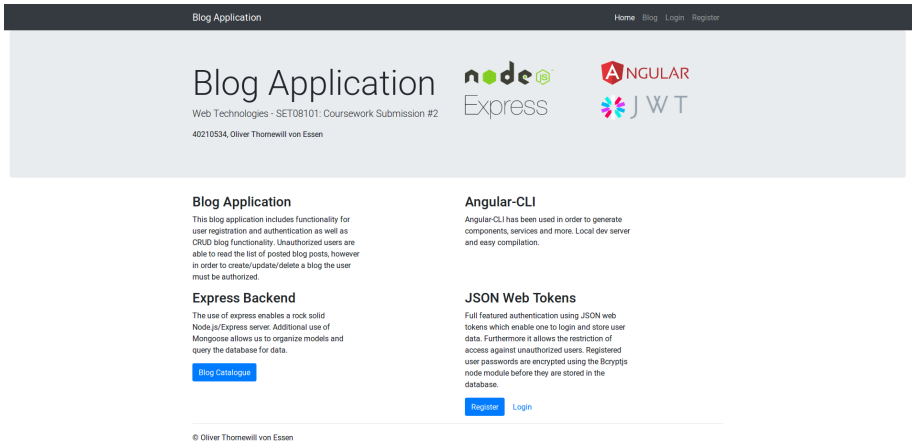

Figure 2: Front page

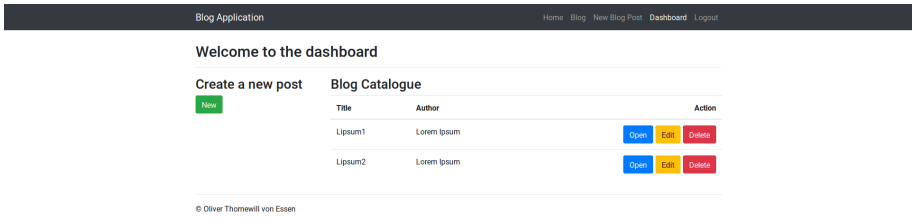

Figure 3: Dashboard

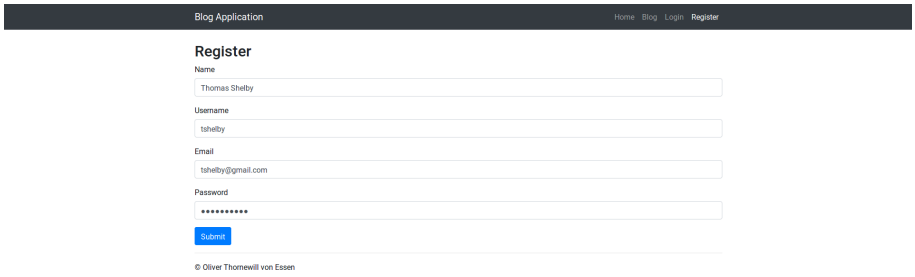

Figure 4: Register form

4 Critical Evaluation

4.1 Checklist against plan:

- Users
  - ☑ Register account
  - ☑ Authenticate to log in
- Blog Posts (persisting through server restarts)
  - ☑ Read posts
  - ☑ Create posts after authentication
  - ☑ Update posts after authentication
  - ☑ Delete posts after authentication
  - ☐ Comment on blog posts
  - ☐ Search through blog posts

- Consistent visual design such that the pages look similar despite varying in content.
  - ☑ Navigation bar
    - ☑ Changes state based on if user is authenticated or not. For example, logout button only shows when user is authenticated.
  - ☑ Footer

The evidence suggests that the implemented application has gone beyond what the initial requirements were, demonstrating knowledge in user registration and authentication. This went beyond the scope as the initial requirements asked for an administrator user and non elevated permission users. The current implementation was able to make use of JWT and BcryptJS to create a secure authentication application.

4.2 Possible Improvements:

4.2.1 Search bar

In a situation where there are many blog posts on the platform, there is currently no way to search or sort the posts. At a lower scale implementation of the blog platform this is not so much of an issue to scroll through 10 posts to find a specific post. However, if the platform had many more posts a searching mechanism would be required.

4.2.2 Commenting

Expanding on the search bar possible improvement, a system that would allow users to comment on the separate blog posts would be a great extension

4.2.3 Sorting

Currently there is no way to sort the blog posts. For example an extension could be sorting based on the most recently updated or alternatively, alphabetically by the Author / Title.

4.2.4 Automatically add author upon creation of blog post

The application in its current form does not implement the author on the creation of a new blog post. This application trusts that the users will be honest when implementing the author.

4.2.5 Permissions

The application has no permissions beyond authenticated and not authenticated. There is no separation between an administrator or an editor. Anyone can make an account and edit/delete other people's contents.

4.2.6 Mobile website

The application does not scale well visually for mobile displays. Typically a mobile version of websites makes it easier for the user to navigate without the use of a mouse using a touch screen.

4.2.7 Allowing users upload files

Users cannot upload files along with their blog posts. For example it is not allowed to post an image or alternatively a PDF document, restricting the usability of the blog application.

4.2.8 Security Audit

There has been no security audit on the authentication application. It is currently unknown if the text boxes can be manipulated by the user to perform malicious attacks. It is unsure if an attacker can alter the local storage in order to illegally obtain a valid authentication token.

5 Personal Evaluation

This section will take the opportunity to not only reflect on the project. As a whole I feel that this project was very demanding and the following is going to discuss not only what I learned but also the challenges I faced, the methods I used to overcome these challenges, and how I feel that I performed.

5.1 Lessons Learned

Through the use of HTML, Node.JS including Node Modules, I was able to make an RESTful blog platform from scratch. I learned how to use AngularJS and Express in order to create a dynamic website. I learned how to implement Bootstrap stylesheets enabling me to use classes that are scalable according to the user's window size. Successfully using MongoDB noSQL as a data store permits the blog posts to be stateless. Further extending lessons learned, I was able to implement a user registration and authentication aspect to the application. This was done using BcryptJS alongside with JWT.

5.2 The Challenges that I Faced

Before starting this project, I had no experience with Node.JS, login systems, or data stores and such had a steep challenge to understand what modules are needed and what are not. During my research of implemented systems, I found it difficult to activate it into the application that I had built as were using different modules, and without a redesign I could not use it. For example I was using AngularJS while the previously implemented system was using Handlebars. Instead I researched AngularJS implementations. I broke down the application into two tools, an authentication and a blog tool. Once these were complete it was difficult to fuse the tools together to make a functioning application, as errors were coming up which I did not understand. For example, I was having an HTTP:404 error even though the page was successfully able to render.

5.3 The Methods that I used to Overcome the Challenges

I researched previously implemented login systems or data stores in order to see what structure is required. I found that this did not ultimately bring me to the solution that I needed because while it was working on the platform demonstrated, I could not implement it into what I had previously created without a serious re-design. In order to overcome this, I broke down the application from one into two tools, an authentication and a blog tool. This allowed me to isolate variables and made troubleshooting easier while building a working tool.

5.4 Evaluation of my Performance

I realize that there is still a long list of improvements to be made, and surely there are more improvements that I have not come across during my brainstorming. I think that my performance has been adequate for the requirements of this project. While there are improvements to be made, there is a time constraint that I must adhere to and it cannot be adjusted. I feel that I have covered the main bases in regards to a blog platform.

References

[1] S. Wells, "Web-apps and restful design," Mar. 2018.

[2] S. Wells, "Dynamic sites using node.js," Mar. 2018.

[3] AngularJS, "What is angularjs?."

[4] "Express - node.js web application framework documentation."

[5] "Node.js express framework," Jan 2018.

[6] "Json web token."

[7] "Json web tokens introduction."

[8] "Npm, bcryptjs documentation."

[9] B. Traversy Media, "Mean stack front to back," Feb 2017.

[10] D. J., "Tutorial building crud app from scratch using mean stack (angular 2)," Aug 2017.