RESTFUL SERVICE

Yi Rong
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
y.rong@student.vu.nl

Hongyu Wu
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
wu2@student.vu.nl

Jiahui Xiong
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
xiong2@student.vu.nl

1. ABSTRACT

This report provides detailed documentation to the group assignment1 of the course Web Services and Cloud-Based Systems. In this assignment, we implement a URL shorten service based on RESTFUL API using spring boot.

2. IMPLEMENTATION

The main function of our restful service is to process the input URL and return a shortened URL which can redirect the user to the original page when clicked. Besides, the validation of the input URL will be checked before uploaded and the uniqueness of the shortened URL is maintained, which means that one shortened URL will only be mapped to one long URL.

Additionally, the generated shortened URL will be stored in the databases where the edit and delete operation can be completed manually. To monitor the access of the URL we generate, usage of the shortened URL will be presented in form of a graph. To be noticed, every request in the service will return the status code accordingly based on RESTFUL protocol.

2.1 RESTFUL SERVICE

According to the RESTFUL protocol, each service would return a status code accordingly, which is shown in Table 1.

As for the PUT request, the first version of our implementation is to return a web page, on which the user could update the URL that is mapped to the input id. But during the communication with our teaching assistant, we were informed that there must be update when respond to the PUT request, which means that our first approach is not proper. So we made another attempt, which is to include the new URL in the request body, and the server will directly change the original URL to the new one and return a status code 200.

2.2 ID GENERATION ALGORITHM

To generate shortened URL, we adopt the base62 encoding scheme which consists of the capital letters A-Z, the lowercase letters a-z, and the numbers 0–9. Our algorithm is to randomly selects 6 characters from the character set base62 and use it as a short identifier. The result of our algorithm is shown in Figure 1:

Table 1: Restful Service Table

HTTP Methods	SERVICE	STATUS CODE
/ - POST	Shorten long URL from the request body and return the shortened URL	201
/:id - PUT	Mapping the corresponding id to a new URL from the request body	200
/:id - DELETE	Remove URLs from the database according to the input id.	204
/id - GET	Receive the id and accordingly redirect the user to the original page	301
/:id-DELETE,id-GET,id-PUT,DELETE	Delete,edit or enter the id that does not exist.	404
/:id-PUT,POST	The submitted id does not conform to the right format.	400

Figure 1: Shorten Example

As shown above, the input long URL is shortened to *9QZtad*. The advantage of this design is that it produces a service that can handle a wide range of demands due to the diversity of random combinations of characters in the character set. Noticeably, to ensure its uniqueness, we first do a query on the database to check whether the shortened URL already existed. If so, we will regenerate a new id to do the mapping.

3. ANSWER TO THE RESEARCH QUESTION

In order to allow multiple users to operate on the service to shorten the URL, We can implement multi-user access to the URL shorten service by requesting that users create an account, then using the account login. We can accomplish this by developing a unique authentication system or by using a third-party authentication service (like *OAuth*). A token with a specific identifier linked to the user's account should be given to them after they have successfully authenticated. The database also needs some modifications. Tables in the database should contain data on users, URLs, and their relationships. Only the user-created URL mappings are accessible for creation, deletion, and viewing. This can be achieved by verifying the user's unique identifier linked to each URL mapping in the database and allowing them access to only those mappings that are linked to it.

4. THE BONUS PART

4.1 Data Base

In the current task we adopt *mysql* as our database to store the URLs. The database operations for the URL shortening service in Spring Boot is implemented by Spring Data JPA which include creating new mappings when users input long URLs, updating existing mappings when users edit URLs, and deleting mappings when users request deletion of shortened URLs(as shown in **??**). After each operation is done, the service will return the status code accordingly as we introduce in the previous sections. The data schema can be seen as follows:

Table 2: Database Schema

Column Name	Data Type	Description
id	INT	Primary key for the URL shortening
long_url	VARCHAR(255)	Original long URL to be shortened
short_url	VARCHAR(255)	Shortened URL generated by the service
daily visits count	LONG	seven other Long fields(mon to sun) to store the visiting numbers of the shortened url

4.2 URL usage analyze

As part of our URL shortening service, we also implemented a function to track the usage of shortened URLs. When a shortened URL is visited, the system records the access in a database for tracking purposes. To visualize the usage data, we utilize Echarts, a popular JavaScript charting library, to generate bar plots that display the frequency of access for each shortened URL, providing valuable insights into the popularity and engagement of the shortened URLs (as presented in figure 2).

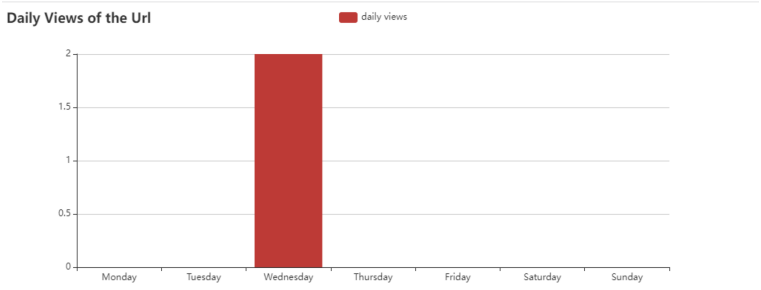

Figure 2: daily access count for shorten URLs

5. INDIVIDUAL CONTRIBUTION

	main contributor	inferior contributor
RESTFUL service	YI RONG	HONGYU WU, JIAHUI XIONG
URL Shortener	JIAHUI XIONG	YI RONG, HONGYU WU
URL Validation	HONGYU WU	YI RONG, JIAHUI XIONG
Data Base	YI RONG	JIAHUI XIONG, HONGYU WU