Web Services and Cloud-Based Systems

Assignment 1.2
REST URL-Shortener

Daniela Cretu
11163372
Aida Pločo
11160381

April 8, 2016

Table of Content

1. Implementation report

We chose PHP as it seemed like the most straightforward choice, because it allows the parsing of parameters of the HTTP requests ($_GET, $_POST).

Since the assignment did not require to maintain states, we chose MySQL database for storing the URLs. The file “dbsettings.php” contains the necessary data for connecting to the database. The database has only one table, namely urls. This table has three columns, “id”, “url”, “short”. “Id” represents the id in which it is saved in the database, and has further no value to us. “Url” is the original URL provided by the user, while “short” represents the unique identifier which is created by the function unique_id() in the “rest.php” file. This function currently creates unique identifiers with length=5, because it seemed as an appropriate length, however this can be altered through the code. All of the columns require the inputs to be unique, for “id” and “short” it is obvious why this is necessary, however for “url” we decided upon this so that there would not be duplicate urls, because this would unnecessarily clog up the database and the number of unique identifiers.

The original way to pass parameters in PHP is as follows: localhost/rest.php?short=abc12, however this can be fixed by creating a .htaccess file through which we redirect all calls received as localhost/abc12 to localhost/rest.php?short=abc12.

The “index.php” file contains a simple form which has a text field and a button. The user copies an URL, which first needs to be validated. If it passes the test, the user receives the unique identifier to that URL, which can then be accessed as mentioned above, and this results in a redirect to the actual page of the URL. Otherwise, the user receives “Invalid URL”.

2. Answer to required question

Since we are using this URL-Shortener locally, having multiple users using it locally will not create any conflicts, since it is not one big shared database, but instead separate databases for each user. However if we were to deploy this on some remote server, the current code would not be able to differentiate between users. This does not create problems for GET and POST, since it will just retrieve the unique identifier, whether the URL is already in the database or not, making it irrelevant which user actually added the URL. But for PUT or DELETE this would create enormous problems, since it would allow for one user to edit/delete an URL, while another user might still be needing it. The simple way is to allow users only to use GET and POST, which is in essence what all URL-Shorteners do, this way we do not have to change anything in the code. If we really must enable users for some reason to edit/delete URLs, this can be done by adding another column called e.g. “user” thereby specifying which user applied which URL. Another reason why this might prove useful is because there is a limitation to the number of unique identifiers, due to the length of the unique_id() function and the fact that it can only take alphanumeric characters. If we were to differentiate between users, we could allow each user up to 36^5 combinations, instead of 36^5 in total.

3. How to run

First create a user and database in MySQL, these credentials are to be swapped with the ones we provided in “dbsettings.php”. The tables and columns will be created automatically through the HTTP requests.

For the .htaccess file to work, the rewrite_module needs to be activated in Apache (Apache -> Apache modules -> rewrite_module).

Run the server and access the “index.php” file through the browser. Paste any URL in the text field and click on “Shorten URL”. This will redirect to “rest.php” and will return a unique identifier. Copy this identifier and paste it **instead** of “rest.php” in the URL of the browser. This in turn redirects to the actual page of the given URL.

This way GET and POST methods have been tested. For PUT and DELETE it is possible to use other programs such as cURL[1]. In our case, we tested it using Postman[2].

1 https://curl.haxx.se/
2 https://www.getpostman.com/