Assignment 1.2 - RESTful Service - Lab Report

Katerina Chinnappan, Rohit Shaw, Kjell Zijlemaker

Design & Approach

We have chosen the Python micro-framework *Flask* to implement the RESTful service for this part of the assignment. The user can perform four HTTP-bound operations: get an overview of the IDs, get redirected by filling in ID bound to the URL, add new URLs for the new shortened URL, edit them and delete them. The server itself is designed to respond to these four operations. It keeps the shortened URLs in memory for the user to operate on and also checks if the URLs are in correct format before saving them into memory.

Implementation

REST Service - `REST.py`

The REST service is built with Python, in particular with the Flask package which can be used to serve as a REST server. It consists of four operations which all correspond to the HTTP operations: `GET`, `POST`, `PUT` and `DELETE`. All resources are pointed to the root of the server '/' and can be requested by arguments placed in the URL and `POST`.

For keeping the URLs and binding them to an unique ID, a dictionary has been used, combined with the Nanoid package has been used for Python. This package helps creating unique IDs which also can be expanded or reduced in size.

To check if a correct URL has been given by the user, a regex check has been used in accordance with the `GET`, `PUT` and `POST` operations. This regex will check if the URL is beginning with `http://` or `https://`. Furthermore, it will not allow any ports to be given but will allow IP addresses and upper-casing as browser will automatically reduce to lowercase.

The `GET` operation is programmed to respond to two requests: one with a given ID and one without. The first will give the redirection to the bounded URL and the latter will give all the IDs which are present in the dictionary. The `DELETE` operation is straightforward as it checks if the ID which has been given exists. If so, it will be removed from the dictionary.

The `PUT` operation is used to edit a given ID to a new URL. The server checks if a correct URL and ID have been given and will then update the corresponding URL bound to this ID. Finally, the `POST` operation creates new ID's and binds them with the given URL, giving back the newly created ID.

Client Web Page - `index.html`

The client has been built with HTML and JavaScript (jQuery). The web page consists of three fields where the user can add a new URL, see all URLs with saved ID's, edit these URLs and remove/edit them. Finally, the user can also insert the new ID to be redirected to the original URL.

The HTML consist of three forms where the user can enter the URL to be shortened, to get redirected to the URL from an ID, and to edit a particular URL. Finally, two buttons are also present which can be used for going to the edit form of a particular URL or to remove an ID.

The JavaScript consists of all four HTTP operations which are also present in the REST server and will do requests with AJAX. Because the server will check the URLS sent by the user, the client side does not check for valid URL inputs. The AJAX responses will check for error codes given by the server, which will then result in a error message suitable for this error code.