Web Services and Cloud-Based Systems
Assignment 1.2

Felix Jose Farias Fueyo 12180424 felixfariasfueyo@hotmail.com
Daan Klijn 11393181 daanklijn0@gmail.com

Vrije Universiteit Amsterdam

RESTFUL service

Implementation Description

For this project it was needed to develop a URL-shortener. The URL-shortener should be able to store inputted URL's under a randomly genarated ID. The user should be able to update or delete the URL that corresponds to a certain ID. And whenever a user visits the "/ID" route he should be forwarded to the URL stored on this ID.

The URL-shortener service was developed using Flask, which is a microframework for python. This framework was chosen because of its simplicity when it comes to creating REST web services. For example, a web service that returns a simple string can already be written in 5 lines of python code.

The resources of a REST web services are stored at certain URI's and can be collected or modified using a range of HTTP requests. For the URL-shortener the paths and their corresponding methods can be found in the table below. For each of the paths a route is created in Flask and then for each of these routes their implementation is specified. When a GET requests is made to the '/' route a HTML file containing the graphical interface of the url-shortener will be passed back. In this graphical interface the user is able to perform all functionality that the application provides, also a list of urls and their corresponding ID is shown.

Fig. 1: Table containing the paths and the corresponding requests

Path and method	Parameters	Return value (HTTP code, value)
`/:id - GET`	`:id` – identificator of a URL	`301, value` `404`

`/:id - PUT`	`:id` – identificator of a URL	`200`
		`400, "error"`
		`404`

`/:id - DELETE`	`:id` – identificator of a URL	`204` `404`
`/ - GET`		`200, keys`
`/ - POST`	`:url` – URL to shorten	`201, id` `400, "error"`
`/ - DELETE`		`204`

URL shortener for multiple users

In the current situation the application could be used by multiple users. However, all of the user would share the same list of URL's and ID's. This might result in too many URL's being displayed in the graphical user interface which makes it impossible for the user to see which URL's are his. This problem could be solved by creating a system where the user can create an account and can use this account to store his URL's and corresponding ID's. Then, the user will only be able to see his own URL's, however if someone has got the ID he can still get the redirect. To implement this, one needs to add the process of logging in and signing up and a database where this kind of data is stored. After that is implemented, one needs to make sure that a session cookie is also passed around whenever the user want to see the graphical user interface so that he stays signed in while refreshing the page.