Assignment 1.2: Web Service (REST)

Bernardo Rui Andrade
12531650
bernardo.andrade@tecnico.ulisboa.pt

João Almeida
12531693
joao.santos.almeida@tecnico.ulisboa.pt

April 7, 2019

Design and implementation description of the URL shortener

The architecture of our Web Service is based on a Client-Server model. In this case, the service is stateless meaning that the client will only be allowed to make independent requests to the server. The client reaches the server through a hardcoded url and has a previous knowledge of the available functions. .

The REST Web Service was implemented in java using Spring-boot framework and Maven. The server consists in three parts: URLshortener, Controller and Application.

The controller is responsible to answer to the remote REST requests. It was implemented using spring-boot and has a *@RestController* annotation such that it can be treated as one by spring-boot. In the controller are every method that the server provides. Every method has the annotation *@RequestMapping* that maps the URL and the type of the HTTP request to the proper method that will be responsible to handle the request. Then, the method calls a domain function in the URLshortener to treat the request, and depending on the response from the URLshortener sends the appropriate reply (success or error).

The URLshortener is a singleton java class that handles all the domain functions. This way, this instance can be shared between multiple clients if necessary and all the clients will have the same map URL to ID. It provides all the methods that can be called by the Controller and maintains a hashmap that maps URLs to IDs.

The Application is just a simple class with *@SpringBootApplication* annotation to tell sprinb-boot where is the main class to start the application.

In the client there is only the application with the main method that is responsible to execute a variety of request to the server that will test every type of response for every availabe function that the server provides.

Solution for making the Calculator service stateful

A simple solution to maintain state (assuming multiple users) is adding an unique Client token/identifier to each request (e.g. with a SOAP handler we can attach it to the SOAP header/body of the message) sent to the service and then the server keeps the result of the operations/state in a key-value data store (<k,v> = <Client UID, state>). The client in his future requests has to decide between methods that will use the previous result/state or methods that don't require the previous result (like the stateless methods already defined in the previous service implementation).