QuickLinks Design Document

Contact on Github at @MasterChief-John-117

What is this?

A simple, lightweight link shortening API for use on a personal web server

Principles

1. All primary endpoints MUST accept GET requests to ensure web browsers can be used to interact with it

Goals

1. Be simple and easy to use
2. Provide fastest possible response times
3. Utilize native HTTP features for redirection as to be non-browser-reliant
4. Provide simple analytics information upon request

Non-Goals

1. Provide a graphical front-end
2. Allow for IP/Client blocking or require authentication
    a. These can be implemented at the webserver level and should not be implemented here

Requirements

Use cases

1. Adding a new URL via browser
    1.1. User visits website to add link
    1.2. The website gives him the new shortened link in a form that does NOT redirect along with the unique ID
2. Visiting a shortlink
    2.1. The user types the shortlink into the address bar
    2.2. If the link exists they are redirected via a 301 to the long link
    2.3. If the link does not exist the api should handled the 404, not the web server
3. Retrieving analytic information
    3.1. The user visits example.com/analytics/UID
    3.2. The if the UID is invalid it should return a 400 Bad Request along with formatted information about the error
    3.3. If the UID is valid it should return the stored analytic information along with the shortlink and longlinks in plaintext

Minimal Viable Product

The MVP should be able to add and return links. Analytics are not included

Other Requirements

1. The same URL added multiple times should not have the same shortlink
2. Two URLs added at the same time should not have the same shortlink

Design

Algorithms

1. Link generation
   1.1. Identical links should NEVER be generated
   1.2. The link should be alphanumeric and both rememberable and easy to distribute
   1.3. The link should be three to five English words randomly selected from a words list
   1.4. The first letter of each word should be capitalized
   1.5. After generation uniqueness should be checked

Data Structures

ShortLinks should be stored with the original URL, analytical information, and a unique string different from the shortlink that can be used to retrieve analytics

Analytic information should include total page views and unique visitors

Databases

The database should be automatically set up and usable on any system without any admin interaction. LiteDB (https://github.com/mbdavid/LiteDB) is embedded and NoSQL, and tests have shown as fast if not faster than SQLite for querying and single-record insertion, so it will be used. The words for creating links will be stored in the database as well in another table.

Other Notes

Word list is compiles from the medium and long words in https://github.com/first20hours/google-10000-english. The short words will not be included