My Tiny URL Design Document

Mark Aronov

Table of contents

Introduction

The purpose of this document is to outline the technical design of the URL shortening tool and provide an overview for its implementation.
This document describes the implementation details of 'My Tiny URL'.
The software consists of two major components.
First the back end that is made to store the redirections and serve the user stats.
And the second component which is made to serve the user for input and output.

Design Overview

Description of Problem

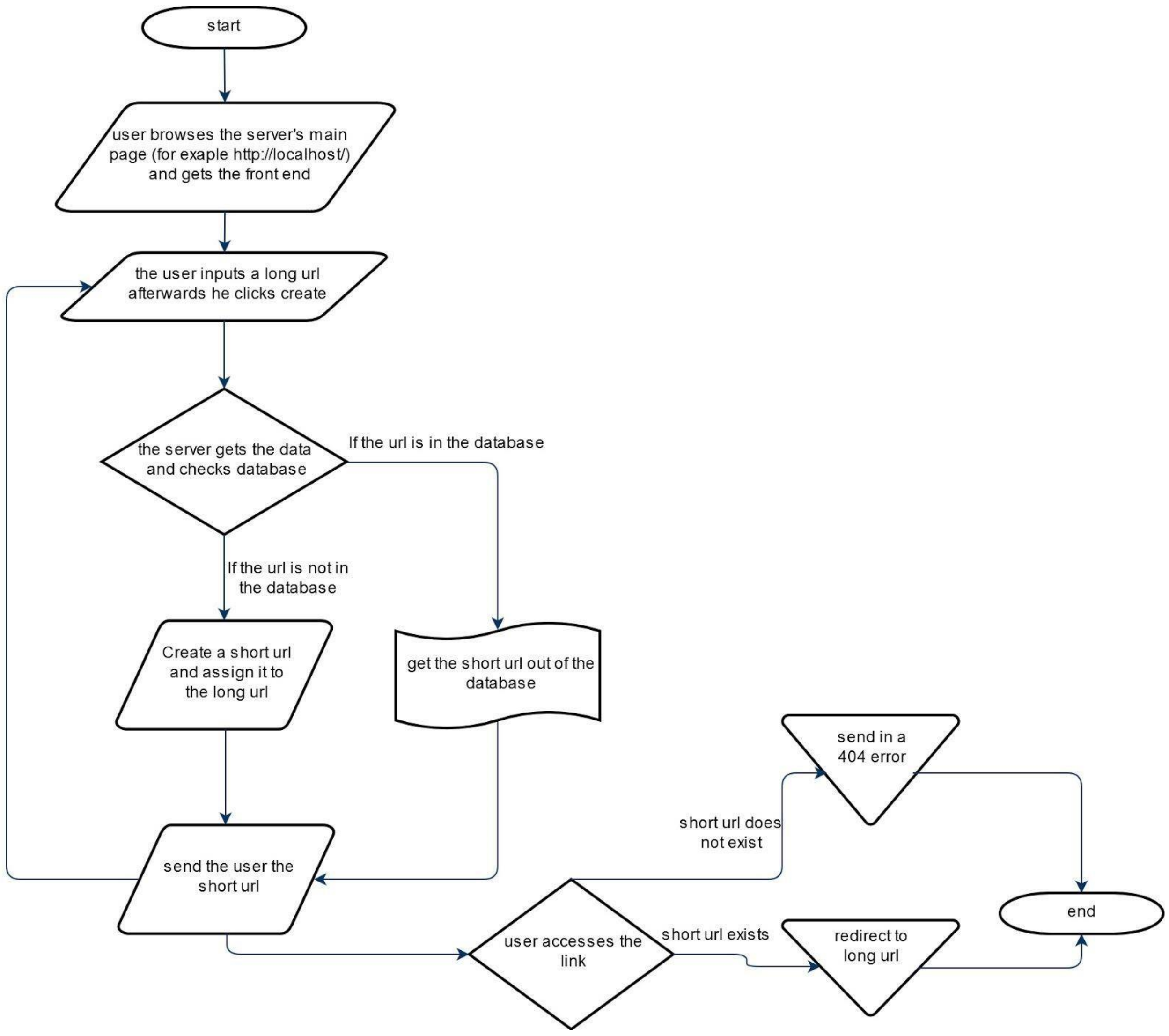
Often, regular URLs end up being hundreds of characters long and might contain complex characters and such.
These URLs are difficult to memorize or share with others.
URL shortening services are made to solve this issue by using URL redirections.

Technologies Used

For the backend I used python 3 with the SQLite database and Flask web framework which runs a server in a Linux distribution Ubuntu 18.04
As for the front-end I used JavaScript with HTML and CSS

System Operation

The following graph shows a typical sequence of events that occur during a URL creation and usage.


start
user browses the server's main page (for exaple http://localhost/) and gets the front end
the user inputs a long url afterwards he clicks create
the server gets the data and checks database
If the url is in the database
If the url is not in the database
Create a short url and assign it to the long url
get the short url out of the database
send the user the short url
user accesses the link
short url does not exist
send in a 404 error
short url exists
redirect to long url
end

Server Interface

The following tables show in detail the components of the server

main.py

make_short_url()	This Function creates a random 7 char length URL address for the short URL. The reason I choose 7-character long string is because of the amount of string combinations. The alphabet characters with the number digits make up to 62 options (26 uppercase chars + 26 lowercase chars + 10 digits) for a character and it being 7 char long string the number of possibilities is 62^7 (3.5216146e+12 i.e., too much) Input - None Output - a string that contains the short URL
index()	For the main page of the server, we send to the user the index.html file which is the front end for the user Input - None Output - index.html the front end
fetch_long_url(short_url)	The function gets a short URL as a parameter with it we check if there's already made redirection. If there is, we pull out the database and send it if it’s invalid or a non-excising then we send a 404 error to the user. Input - short_url: the URL to work with Output - a redirect to the original URL OR a 404 error
receive_from_user()	With this function we receive the data from the front end in a JSON form, we convert the JSON data into a string and check if the long URL is found in the database if found we send the already made shorten URL if not, we make one and assign it to the long URL then we return to the user the short URL Input - None Output - shorten URL
stats()	Bonus 1 , we send in the stats about the number of redirections, number of successful and unsuccessful redirections in last day hour and minute Input - None Output - String that contains the number of redirections

database.py

database_init()	The initializer that checks if there's the tables already and if not, we create them
Add_to_database (full_URL, short_URL)	The functions that add a new redirection to the database if it does not exist Input - full_URL, short_URL Output - None
Get_from_database (long_url, short_URL)	Gets the redirection from the database, either the short URL or the long URL Input - full_URL, short_URL Output - the URL from the database
Update_visit (full_URL, visit_type)	Updates each redirection to the statistics database If the visit_type is 1 we update the visit count of the redirection in the database and update the stats as well Input - full_URL, visit_type Output - None
get_amount_of_urls()	Gets the amount of redirections Input - None Output - String that shows the number of redirections
Get_redirection_stats (visit_type)	Gets the number of redirections and errors that were made in the last-minute hour and day Input - visit_type (1 for regular 0 for errors) Output - String that shows the number of redirections

Front-end Interface

The front end is made with a simple combination of JavaScript HTML and CSS in which the .js file only contains 2 functions **window.onload function** - which waits for the window to load so that the input could be identified for extraction **function handleSend()** - which checks if the input is empty or not, if empty an error will display and if not, a fetch request will be made with the data being the input in JSON, the fetch request will also receive data from the server, but this time in plain text because of the simplicity of the short URL (for not having complex characters)