Test Objectives

Create a simple in-memory cache with an HTTP interface

Interface

● HTTP POST /<key> with the value as UTF-8 body

● HTTP GET /<key> replies with the value as body or 404 if no such key exists

Think About

● Use idiomatic Go or Rust

● Stored key expires after 30 minutes

● The server is expected to handle a large load with an 80/20 ratio between reads and

writes

● Write it as if would would be expected to continue to support it in production for the

foreseeable future

Deliverables

* Go source file with source code
* HTML document with a simple form to input data

Test progress and personal opinion

Used Golang for programming the solution. Being fairly new to the system, referred the documentation on Golang website for reference

Objectives I could achieve:

* Set the environment for Go
* Implement web socket with and without interaction with DB (used MySQL & SQLite DB for testing)
* Start the web server and serve my html page
* Showing of response on browser after the user has input the data
* Random key generation function to generate a random key on the fly (I used UUID but thought the existing one is simple)
* Timer function to control start and stop
* Concurrency implementation of timer looping

Objectives I left figuring out

* URL redirection for GET & POST, I tried to index a page with both links then passing the random key value to HTML was not working, since the .tmpl template path detection was throwing much errors

Overall it was a nice experience to understand and manage a task in Go

Working hours on test: 8hrs including installations and download

Git link: https://github.com/cgmafia/go\_test