URL Shortener.

The URLShortener service works in this manner:

1. First thing it does is to shorten a given URL
2. Second task is to redirect that given short URL, to its original destination.

The URLShortener is triggered as the endpoint when a user makes a request to shorten a URL.

The Main class instantiates and starts the URLShortener. Once its task is done, it then closes it.

It locates a configuration resource file by name and then uses it to come up with the URLShortener instance.

The UrlShortener is responsible for shortening the long URL and redirecting the short URL to the long URL’s destination.

We use a hashing function, through an interface that we refer to as Shortener in this design.

We map and replace a long URL to a short URL which is getting generated by the service.

We need to avoid a case whereby there are no available short URLs being generated or if they get generated, there are collisions in that, same URLs were generated earlier.

For shortening, we use the sipHashFunction. This function, uses a Google library known as Guava that enables string shortening with a very minimum chance of collision – Since we need the service to shorten URLs and the short URLs to route to the long URL destinations, many users will shorten their URLs.

Now upon request of a short URL, the response redirects to the original long URL.

We need to store the records in a database thus we use a database (SQLite) to store the keys and values, short and long URLs, this is done by the storageProvider class.