What have I got myself into?

At Lüp we like to do things differently from other technology companies.

Introduction

This test has been designed to measure your ability to work with our technology stack, and to produce software to the standard that is expected by our team and customers.

How we will assess you

We'll assessing you for the following the following criteria:

- Your ability to solve the solution (duh)
- Your approach to solving the problem
    - Does it follow best practices?
    - Is it efficient?
    - Is there any edge cases where it might fail?
- Your Git commits
    - Make regular Git commits so that we can see your thought process
- The ease which a future developer will be able to maintain your code.
- The time you take to solve the problem.
    - Email the solution back as soon as you complete it so we know how long it took you.
    - Bonus for solving it and emailing the solution back within three hours.
    - If you're constrained on time we'd rather you complete one task thoroughly than all tasks partially
- Your ability to follow instructions closely (don't add any additional functionality!)

The problem

We provide event services to stacks of events in Australia and New Zealand. In fact, some weekends our system services over 100,000 unique visitors for these events. For many of those events we provide SMSs to visitors to remind them to register or other functions. Those SMSs often have URLs in them.

It's common that the use of standard URLs forces the message over multiple SMSs where a shortened URL would have allowed the message to fit in just one SMS. If you do the maths, that makes a significant cost difference!

At the time when we researched existing URL-shortener services like Google URL Shortener API and Bitly we were disappointed to discover that they proved unreliable or unviable (high cost) for large volumes of URLs. Therefore we decided to produce our own basic URL shortener.

Your task is to build a simple URL shortener that can be run at scale to solve this problem.

Your solution must be:

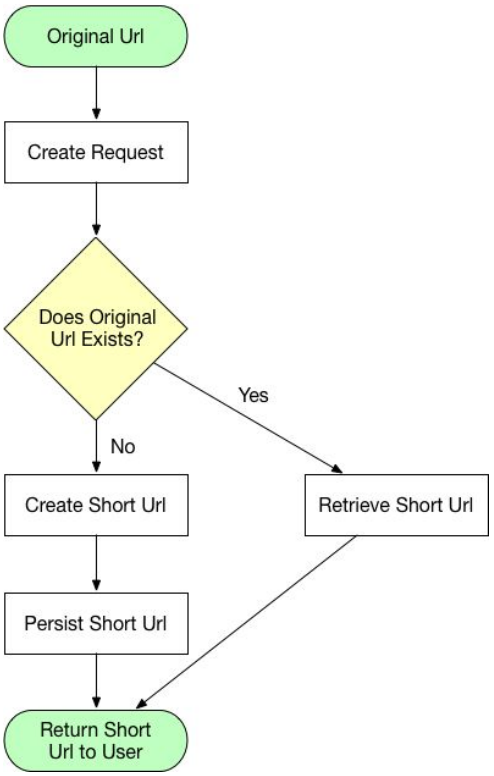
- Written in .NET Core 2
- Deployable to Azure App Services
- Persist data through Azure Table Storage
- Captured in Git, with regular commits

To assist you in completing this task our "Standard" development environment has already been setup, along with a skeleton solution that includes:

- A skeleton application
- Credentials for Table Storage.
- Build toolchain (build.ps1 / build.sh)

Task #1: Ability to create a shortened URL via an API call

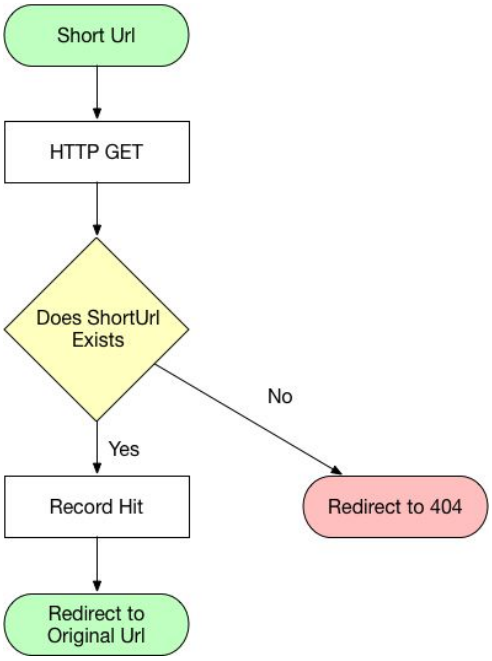
When our core system has a URL that needs to be shortened it will make an API call to this URL shortening service containing a full URL. This service must confirm that the request is sane, before creating and returning a shortened URL. URLs which are not sane (prefixed with “http://”, “https://” or “ftp://”) must be rejected in an intelligible way. Also, if the same URL is shortened multiple times the service must return the same shortened URL.



Authentication is out of scope - it is safe to assume that any service calling this endpoint is authorised.

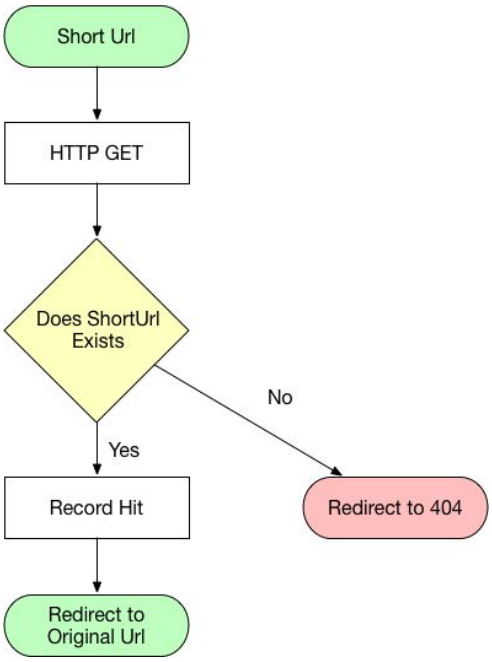
Task #2: Redirect user to original URL

After a URL has been shortened (as per Task #1) it will be SMS'd to an event attendee. Eventually (hopefully) they will tap on the link, calling the given URL. When this happens this service must redirect that request to the original URL. Attempts to access an invalid short URL must give a intelegable error.



Task #3: Record the number of hits for each short URL

The system must count the number of short-URL hits so that we can later analyise the usage of short-URLs.



Note that providing the number of hits to an administrative user is outside of scope.

Final word

Lüp isn't scared of failure, we embrace it. What does that mean for you? It means it's okay if you are unable to complete the task within the target three hours.

Good luck!