Backend take-home exercise

Prompt

Write a server that acts as a bit.ly-like URL shortener. The primary interface should be a **JSON API** that allows the following:

- Create a random short link for arbitrary URLs, e.g., bit.ly/2FhfhXh
- The same URL should always generate the same random shortlink
- Allow creating custom short links to arbitrary URLs, e.g., bit.ly/my-custom-link
- Provide a route for returning stats in a given short link, including:
    - When the short link was created
    - How many times the short link has been visited total
    - A histogram of number of visits to the short link per day
- Of course, the server itself should handle redirecting short links to the URLs it creates

Everything else is up to you: the routes for the API, what its parameters and return values are, how it handles errors. Of course, I will be happy to talk any of these things out, but you are empowered to make whatever design decisions you like (and ideally explain them!).

Submission

The goal should be to have a functioning server running locally on your computer (and ideally easy to run on someone else's computer). Code should be submitted as a **git repo**, with an initial commit when you start working.

We don't expect you to spend more than four hours on this problem, though we won't strictly time you. At the end it should be as close to deployable as possible. We'll go over it together to look at the code, and give you a chance to explain your design choices.

Expectations

Code

The final should be clean and easy to read, extensible, and narrowly scoped to the use-case. We don't expect any crazy fancy abstractions, or for you to predict what features we'd want to add, but the code shouldn't be painfully difficult to extend if we needed to either.

Architecture + scaling

You should be able to talk through your architecture + explain the issues you would expect to face as you scale your solution.

Comments

We don't expect this to be crazy documented, only commented as much as you think would be needed in a production, deadline-driven environment. (Which is to say explain things that might be confusing, but otherwise the code can speak for itself.)

Tests

Likewise we don't expect a full suite of unit, integration and acceptance test, nor do you need to practice test-drive-development, but the key requirements should be tested—and testable—in a reasonable way.

Language + technology

You can write the server in any language and using any technologies you choose (including your choice of database). There are no limitations on using open-source libraries, though be prepared to explain your choices.

Help

Feel free to use the internet or whatever resources you need. The code itself should, of course, be your own.

Extra credit

If you have extra time, implement any of the following features for extra credit. Or just be prepared to, like, talk about them:

- Deploy the server to an accessible URL
- Number of “unique” visitors to a given shortlink (you can define how we track visitors)
- A “global stats” endpoint, that aggregates some interesting stats across your URL-shortening platform (e.g., total number of links/visits per domain, histogram of total visits across the site)

Dasdadsasd