STORD URL Shortener Exercise

The goal of this exercise is to create a URL shortener web application in the same vein as bitly, TinyURL, or the now defunct Google URL Shortener. Since we are an Elixir shop, we prefer submissions in Elixir, but you can also implement your solution using other languages (please let us know ahead of time if you use another language, so that we can make sure we have someone on staff qualified to assess it).

The core functionality of the application should be expressed through your own original code.

Although this project is small in scope, it is a great opportunity for you to show off your skills in: attention to detail, testing, CI/CD, design and development. Show us what you got!

Keep in mind that this is a primarily a backend position, so there's no need for a complicated SPA. Plain old fashioned server side rendered HTML with minimal styling is fine.

Application Requirements

- When navigating to the root path (e.g. `http://localhost:8080/`) of the app in a browser a user should be presented with a form that allows them to paste in a URL (e.g. `https://www.google.com/search?q=url+shortener&oq=google+u&aqs=chrome.0.69i59j69i60l3j0j69i57.1069j0j7&sourceid=chrome&ie=UTF-8`).
- When a user submits the form they should be presented with a simplified URL of the form `http://localhost:8080/{slug}` (e.g. `http://localhost:8080/h40Xg2`). The format and method of generation of the slug is up to your discretion.
- When a user navigates to a shortened URL that they have been provided by the app (e.g. `http://localhost:8080/h40Xg2`) they should be redirected to the original URL that yielded that short URL (e.g `https://www.google.com/search?q=url+shortener&oq=google+u&aqs=chrome.0.69i59j69i60l3j0j69i57.1069j0j7&sourceid=chrome&ie=UTF-8`).
- Only allow valid URLs (e.g., start with `http(s)://{domain}/`)
- When a user navigates to `/stats` they should be presented with a basic table that lists each shortened url, its corresponding original url, and the number of

times a user has visited the shortened url. From this page, the user should also be able to download a CSV of the data contained in the table.
- Don't worry about logins or any kind of user authentication or authorization (but we might ask some questions about how you would do this in the next round).
- Your submission should be able to handle 5 requests per second to the submission form, and 25 requests per second to the redirect endpoint, so choose a tech stack and design with this in mind. We will be testing these on current developer laptops (apple M1 Max or core i5+, 16 GB+ RAM).
- You should use a persistent data store (Postgres is encouraged, but not required).

Deliverables

- Implement your solution, including test cases for your application code.
- Please include instructions in `README.md` on how to set up, run, and test your application.
- You may also include any notes for our engineering team that you would like regarding your approach and assumptions you have made in the readme.
- E-mail the point of contact that sent you this exercise and include a link to a public GitHub (or GitLab) repository.
    - Note: if your github account is monitored or the like, then simply create a new github account for this submission.

Evaluation

To ensure consistency in the evaluation, our Engineering team uses a rubric to score your submission, which will be evaluated along the following four criteria by the Reviewer:

1. **Completeness** - Does your submission meet the **Application Requirements** and **Deliverables** specified above?
2. **Testing** - evaluate your use of test coverage to allow for iterative development
3. **Ease of setup** - how easy was it for the Reviewer to setup and run your app?
    a. hint: Try to emulate what it would be like to run a fresh install of your app following your own instructions in `README.md`. If you are not sure how to do that, then simply have someone that you know try the setup --following

your instructions-- on their machine to see if the install, tests, and application all run smoothly

   b. hint: in most cases an experienced developer should be able to get your app up and running in under 15 minutes

4. **Technical design** - Separation of concerns, adherence to certain 12 factor App principles, knowledge of backend frameworks, security concerns, performance, etc. Please be ready to speak to your technical choices during the face-to-face (video) interview, as well as how your design might need to change if the requirements change.

Notes:

- "How long should this take?" Best rule of thumb, "turn this in when you are proud of your work"
    - There's no time limit (e.g., within four days) on when you need to turn this exercise into us. You have a job, family and life that has many demands on your time, and we get that.
    - Just see us as your accountability partner here. You tell us when you think you can get it done, and we'll follow up with you around that time if we haven't heard from you yet. And we can set a new goal on when you can get it done if need be.
- The good news is that we will not subject you to a code exercise on a whiteboard in any subsequent interviews! You will discuss your code submission with the same engineer that reviewed it, which I think is pretty cool.
- Please show us your strengths for **CI/CD (ease of setup), coding, testing, user experience, technical design, and attention to detail! Please show that you are taking this exercise as seriously as we do. For example, your Reviewer will spend 1.5 to 2 hours reviewing your submission.**
    - Please try to keep your solution to 1 repo (if you use docker, then one docker compose file) just so it is a little faster to grade.

Sincerely, I want to thank you for the time that you are spending with us as a candidate with STORD. It is very much appreciated and we know it is a big ask. We hope to have you join our great team members that have made it through the same diligent process.