Kumu Coding Challenge

Write your README as if it was for a production service. Include the following items:

- Description of the problem and solution.
- Installation Instructions.
- Reasoning behind your technical choices, including architecture.
- Trade-offs you might have made, anything you left out, or what you might do differently if you were to spend additional time on the project.
- Link to other code you're particularly proud of.
- Link to your resume or public profile.
- Link to the hosted application where applicable.

How we review

We do take into consideration your experience level.
We value quality over feature-completeness. It is fine to leave things aside provided you call them out in your project's README. The goal of this code sample is to help us identify what you consider production-ready code. You should consider this code ready for final review with your colleague, i.e. this would be the last step before deploying to production.

The aspects of your code we will assess include:

- **Architecture**
- **Clarity**: does the README clearly and concisely explain the problem and solution? Are technical trade offs explained?
- **Correctness**
- **Code quality**
- **Security**
- **Testing**
- **Technical choices**
- **Scalability**
- **Production-readiness**

Coding Challenge

Create a Golang project that has an API endpoint that takes a list of github usernames (up to a max of 10 names) and returns to the user a list of basic information for those users including:

- name
- login
- company
- number of followers
- number of public repos

- The returned users should be sorted alphabetically by name
- If some usernames cannot be found, this should not fail the other usernames that were requested
- Implement a caching layer that will store a user that has been retrieved from GitHub for 2 minutes. If a user's information has been cached, it should NOT hit Github again to retrieve the same user (if it is still in the cache window), instead it should return the user from the cache.
- Include the appropriate error handling and the appropriate frameworks to make the project more extensible.
- Use regular http calls to hit github's API, don't use any pre made github Golang libraries to integrate with github
- The API endpoint needed to get Github user information is https://api.github.com/users/{username}