FACEIT
CHALLENGE YOUR GAME

TASK

To write a small microservice to manage access to Users, the service should be implemented in Go - as this is primary language that we use at FACEIT for backend development.

REQUIREMENTS

A user must be stored using the following schema:

```
{
   "id":
"d2a7924e-765f-4949-bc4c-219c956d0f8b",
   "first_name": "Alice",
   "last_name": "Bob",
   "nickname": "AB123",
   "password": "supersecurepassword",
   "email": "alice@bob.com",
   "country": "UK",
   "created_at": "2019-10-12T07:20:50.52Z",
   "updated_at": "2019-10-12T07:20:50.52Z"
}
```

The service must allow you to:

- Add a new User
- Modify an existing User
- Remove a User
- Return a paginated list of Users, allowing for filtering by certain criteria (e.g. all Users with the country "UK")

The service must:

- Provide an HTTP/RPC API
- Use a sensible storage mechanism for the Users
- Have the ability to notify other interested services of changes to User entities
- Have meaningful logs
- Be well documented
- Have a health check

The service must NOT:

- Provide login or authentication functionality

It is up to you what technologies and patterns you use to implement these features, but you will be assessed on these choices and we expect you to be confident in explaining your choice. We encourage the use of local alternatives or stubs (for instance a database containerised and linked to your service through docker-compose).

Notes

Remember that we want to test your understanding of these concepts, not how well you write boilerplate code. If your solution is becoming overly complex, simply explain what would have been implemented and prepare for follow-up questions in the technical interview.

Please also provide a README.md that contains:

- Instructions to start the application on localhost (dockerised applications are preferred)
- An explanation of the choices taken and assumptions made during development
- Possible extensions or improvements to the service (focusing on scalability and deployment to production)

We expect to be able to run the tests, build the application and run it locally.

WHAT YOU WILL BE SCORED ON

Coding Skills:

- Is your code respecting fields and access modifiers?
- Is your code respecting single responsibility principles?

Application Structure:

- Have you applied the correct division of the layers?
- Do you have the correct dependencies between layers?

Framework Usage:

- Have you applied the correct usage of framework features?

REST endpoints:

- Is your URL structure correct?
- Have you used HTTP verbs?

Asynchronous communication:

- Is it asynchronous?

Testing:

- Are you happy with your test coverage?