Conteinierizando uma API Go

Cleuton Sampaio

Docker é a plataforma mais popular para conteinerizar aplicações, e é possível fazer isso com uma **API Go** de maneira muito simples e fácil.

Para começar, o **footprint** é muito pequeno, pois não precisamos de nada de código-fonte. É até mais seguro! Precisamos apenas do código compilado.

Vamos ver como fazer isso da maneira mais simples possível. É claro que você pode utilizar **Docker Swarm** ou **Docker Compose** para isso, mas nosso objetivo final é instalarmos tudo em um cluster **K8S** (**Kubernetes**), portanto, não vale a pena enfeitar muito o pavão aqui.

Acesse o **código-fonte AQUI**!

Etapas

1. Compile o código gerando o binário:

```
cd code
go build -o ../api.bin cmd/main.go
```

2. Compile a imagem da API utilizando o binário gerado:

```
cd ..
docker build -t api:v001 .
```

3. Cria uma rede Docker e rode o Redis nela:

```
docker network create apinet
docker run -d -p 6379:6379 --name redisbase --network apinet redis
```

4. Execute o contêiner da aplicação e envie alguns requests:

```
docker run -d -p 8080:8080 --env API_DB_URL=redisbase:6379 --name apiserver
--network apinet api:v001
curl -i --header "Content-Type: application/json" --request POST --data
'{"data" : "save this", "onetime" : false}' http://localhost:8080/api/note
curl -i  http://localhost:8080/api/note/b61bc30d-8b2c-41e7-8df7-
36a262826f44
```

Atenção: Copie o id do resutado do POST para poder fazer o GET com ele.

O projeto

Blocopad é uma aplicação para salvar notas rápidas. Você salva uma nota e pode recuperá-la utilizando o código que foi devolvido. Inicialmente, as notas têm duração de apenas 24 horas. Sem registro, sem cadastro, sem cobrança. Futuramente, haverá planos especiais. Sua tarefa é construir a primeira versão da API do BlocoPad em Golang.

```
Recurso: nota
Método POST:
POST api/note
{"data" : <dados string>, "onetime" : true/false}
```

- "data": conteúdo string a ser preservado. Tamanho máximo: 32KB
- "onetime": Se é para apagar depois de consultada.
- Retorno: {"code": "UUID"} UUID é um string com 36 caracteres.

```
Método GET:
GET api/note/uuid
```

- Retorno Status 200: {"data": }

- Retorno Status 404: Not found

Como vamos armazenar as notas? Vamos utilizar o database **REDIS**. Para desenvolvimento, você pode utilizar a imagem Docker padrão do Redis.

Para acessar o Redis, estude o pacote Go-Redis:https://github.com/go-redis/redis Como vamos servir as notas?

Vamos utilizar o GorillaMux: https://github.com/gorilla/mux

Abordagem

O Redis permite criarmos um prazo de expiração para qualquer chave criada, portanto, deixaremos o controle de duração por conta do Redis. Temos muitas coisas que desconhecemos… Vamos começar pelo que conhecemos e isolar o que desconhecemos:

1. Criar wrappers para banco de dados Redis para isolarem a app do armazenamento;
2. Criar uma biblioteca Golang com funções para:
    1. Obter uma chave e retornar
    2. Salvar uma chave
    3. Apagar uma chave
3. Criar testes unitários injetando mocks para o banco de dados.
4. Implementar o acesso Redis nos wrappers.
5. Criar Wrapper com GorillaMux para gerar o servidor REST.
6. Criar teste de integração.

Para gerar um UUID vamos utilizar o pacote: https://pkg.go.dev/github.com/google/UUID#section-documentation

Para rodar os testes

Entre na pasta **code** rode o comando:

```
go test ./tests
```

Para executar os testes de integração, primeiramente suba um contêiner Redis, conforme mostrado no início. Depois:

```
go test ./it
```