Especialización en Back End III

Flujo de un servicio en Go que se autoregistra en Eureka Server

Para implementar el flujo vamos a comunicarnos con la API de Eureka Server. Podemos ver el listado de operaciones disponibles ingresando en el siguiente enlace.

1. Registro

Para registrar una instancia de un microservicio debemos enviar una petición mediante POST al endpoint **/eureka/v2/apps/${appID}**, en donde "**appID**" es el nombre del microservicio. El body que debemos enviar debe tener la siguiente estructura:

```
{
   "instance": {
       "instanceId": "go-ms-1",
       "hostName": "localhost",
       "app": "go_ms",
       "vipAddress": "go_ms",
       "secureVipAddress": "go_ms",
       "ipAddr": "127.0.0.1",
       "status": "STARTING",
       "port": {
```

```
            "$": 8080,
            "@enabled": "true"
        },
        "securePort": {
            "$": 8443,
            "@enabled": "true"
        },
        "healthCheckUrl": "http: //localhost:8081/healthcheck",
        "statusPageUrl": "http://localhost:8081/status",
        "homePageUrl": "http://localhost:8081",
        "dataCenterInfo": {
             "@class":
"com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo",
            "name": "MyOwn"
        }
    }
}
```

Veamos un ejemplo:

```
func registerApp() {
```

```
    appId := "go_ms-1"

    appName := "go_ms"

    body := buildBody(appId, "STARTING")

    var buf bytes.Buffer

    error := json.NewEncoder(&buf).Encode(body)

    if error != nil {log.Fatal(error)}

    resp, err := http.Post("http://localhost:8761/eureka/apps/"+appName,
"application/json", &buf)

    if err != nil {log.Fatal(err)}

    defer resp.Body.Close()

    responseBody, parseErr := io.ReadAll(resp.Body)

    if parseErr != nil {log.Fatal(err)}

    fmt.Println(string(responseBody))

}
```

El método **buildBody** se encarga de crear un objeto como el que Eureka está esperando para registrar una nueva instancia de microservicio.

2. Actualizar estado

Una vez que el microservicio ya está listo para recibir peticiones, debemos enviar la misma petición que enviamos para registrar, pero cambiando el estado. Ahora el estado que vamos a enviar será **UP**. El body nos queda:

```
{

    "instance": {
```

```
"instanceId": "go-ms-1",
"hostName": "localhost",
"app": "go_ms",
"vipAddress": "go_ms",
"secureVipAddress": "go_ms",
"ipAddr": "127.0.0.1",
"status": "UP",
"port": {
    "$": 8081,
    "@enabled": "true"
},
"securePort": {
    "$": 8443,
    "@enabled": "true"
},
"healthCheckUrl": "http: //localhost:8081/healthcheck",
"statusPageUrl": "http://localhost:8081/status",
"homePageUrl": "http://localhost:8081",
"dataCenterInfo": {
```

```
            "@class":
"com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo",
            "name": "MyOwn"
        }
    }
}
```

Veamos un ejemplo:

```
func updateAppStatus(status string) {
    fmt.Println("update app")
    appId := "go_ms-1"
    appName := "go_ms"
    body := buildBody(appId, appName,status)
    var buf bytes.Buffer
    error := json.NewEncoder(&buf).Encode(body)
    if error != nil {log.Fatal(error)}
    resp, err := http.Post("http://localhost:8761/eureka/apps/"+appName,
"application/json", &buf)
    if err != nil {log.Fatal(err)}
    defer resp.Body.Close()
    responseBody, parseErr := io.ReadAll(resp.Body)
    if parseErr != nil {log.Fatal(err)}
```

```
    fmt.Println(string(responseBody))

}
```

3. Heartbeat

Eureka necesita recibir un “latido” de la instancia cada 30 segundos. Para enviar el latido, debemos hacerlo vía **PUT** al endpoint **/eureka/apps/{APP_NAME}/{INSTANCE_ID}**.

Veamos un ejemplo:

```
func sendHeartbeat(appName string, appId string) {

    req, err := http.NewRequest("PUT",
"http://localhost:8761/eureka/apps/"+appName+"/"+appId+"", nil)

    if err != nil {

        log.Fatal(err)

    }

    resp, err := http.DefaultClient.Do(req)

    if err != nil {

        log.Fatal(err)

    }

    fmt.Println(resp.Body)

}
```

4. Dar de baja

Si la instancia se está apagando, debemos actualizar su estado a **DOWN** y luego eliminarla del listado de instancias registradas en Eureka. Para hacer esto enviaremos una petición utilizando el método **DELETE** al endpoint **/eureka/apps/{APP_NAME}/{INSTANCE_ID}**.

Veamos un ejemplo:

```
func updateAppStatus(status string) {

  fmt.Println("update app")

  appId := "go_ms-1"

  appName := "go_ms"

  body := buildBody(appId, appName,status)

  var buf bytes.Buffer

  error := json.NewEncoder(&buf).Encode(body)

  if error != nil {log.Fatal(error)}

  resp, err := http.Post("http://localhost:8761/eureka/apps/"+appName,
"application/json", &buf)

  if err != nil {log.Fatal(err)}

  defer resp.Body.Close()

  responseBody, parseErr := io.ReadAll(resp.Body)

  if parseErr != nil {log.Fatal(err)}

  fmt.Println(string(responseBody))

}
```

```
func deleteApp(appName string, appId string) {
   fmt.Println("sending hearbeat")
   req, err := http.NewRequest("DELETE",
"http://localhost:8761/eureka/apps/"+appName+"/"+appId+"", nil)
   if err != nil {
       log.Fatal(err)
   }
   resp, err := http.DefaultClient.Do(req)
   if err != nil {
       log.Fatal(err)
   }
   fmt.Println(resp.Body)

}
```

Volvemos a usar el método **updateAppStatus**, pero esta vez el valor de "**status**" será "**DOWN**". Después, eliminamos la instancia del listado de aplicaciones registradas en Eureka.