IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 7, Issue 2, April - May, 2019
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

Workflow Management System Implementation using Golang Microservices

Snehal Shyam Nrupnarayan
Sipna College of Engineering and Technology
Amrawati, India

Seema Rathod
Sipna College of Egineering and Technology
Amrawati, India

***Abstract*— This paper explains the implementation of APIs through microser- vices architecture in recently popular programming language GO. The basic approach behind this enactment is to construct a hypothesis to prove the us- age of GO to be advantageous and efficient in terms of server-side coding or API development. This paper will illustrate the process for API development in GO as well as it's performance comparison with Nodejs.**

Keywords*— *Go ,microservice ,api , performance ,Gokit

I. INTRODUCTION

Project development is a planned activity with continuous integration and deployment of functional patches. A typical software product goes through a complete development cycle which involves Designing, Development, Testing and finally Deployment. As far from coding perspective, modularity is a key to cover all possible scenarios, same applies when the whole integrated product is considered. The project or a product is broken down into different components and modules at an architectural level. Designing, Development and Testing of all these components are done and then the final integration is done which again goes through rigorous testing. This in turn indicates that project development is an interdependent process of different components.

Project development is a planned activity with continuous integration and deployment of functional patches. A typical software product goes through a complete development cycle which involves Designing, Development, Testing and finally Deployment. As far from coding perspective, modularity is a key to cover all possible scenarios, same applies when the whole integrated product is considered. The project or a product is broken down into different com- ponents and modules at an architectural level. Designing, Development and Testing of all these components are done and then the final integration is done which again goes through rigorous testing. This in turn indicates that

II. BASIC API MODEL FOR "PROPELLER" THROUGH MICRO-SERVICE ARCHITECTURE

The backend API structure basically operates over four micro-services. As per the already known advantages of microservices, these four services run independent of each other and thus provide a continuous request handling capability even if one of the service fails.

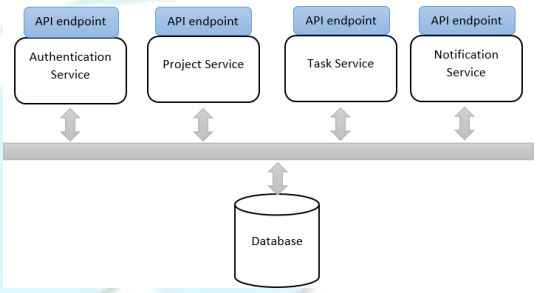

Fig 1. Architecture for Propeller

The functioning of these services are explained as follows:

a) **Authentication Service:** This service particularly works for user iden- tification or profile creation for "Propeller" portal.

b) **Project Service:** This service works at creating project, listing user specific projects, and adding members to project.

c) **Task Service:** This service functions to create, update tasks in projects.These are the key granular components that project members work over

b)**Notiftcation Service:** This service is used to notify user associated events in the portal such task assignment, registration to the portal, through mail.

III. GO KIT

Go is a great general-purpose language, but microservices require a certain amount of specialized support. RPC safety, system observability, infrastructure integration, even program design — Go kit fills in the gaps left by the standard library, and makes Go a first-class language for writing microservices in any organization. Gokit provides abstraction for the development of microservices through commands which can be run on the terminal. It also eases out the deployment of the microservices, which is relatively complex part of API development. It also facilitates the dockerized version of deployment

.

IV. PROGRAMMING FLOW FOR A GOKIT MICROSERVICES

1. **kit new service <service-name>:**
   This command creates a boilerplate for the microservice to be developed.This eases out the development to be done from scratch by auto-generating the folder structure and code prototyping.
2. **kit generate service <service-name>:**
   This command extends the boilerplate by actually integrating the logical modules such as service singleton, functions to handle requests and default middleware.

- **service-name/pkg/service/service.go** -> This file specifically contains the logic the logic that runs behind any api endpoint.
- **service-name/pkg/handler/handler.go** -> This file contains the endpoint to function mapping with defined parameters for any API endpoint.
- **service-name/cmd/service/service.go** -> This file defines the necessary configuration such listen and debug ports for the service.
- **service-name/cmd/main.go** -> The service is made up through executing this file which imports the service singleton and keeps the server listening.

V. PERFORMANCE COMPARISON

By developing similar api in nodejs and Go, on comparison it can be concluded that Go on an average is 50% faster than a similar nodejs service.

Table 1: API response time comparison

API	Go response time	NodeJs response time
/get-all-project	88ms	196ms
/log-in	51ms	231ms

The main reason behind Go's speedup is it's simplified compilation process, statically typed language model and internal usage of go-routines to get extensive concurrency in code execution.

VI. CONCLUSION

Thus, a workflow management system is implemented through micro service architecture in Golang. With the performance comparison, it can be successfully proved that Go is much better in terms of efficiency and speedup as compared to the conventional preferences for API development. With further advancements , Go will only get better at development and should get larger footprint in upcoming development.

VII. REFERENCES

[1] Jose Ignacio Ferna´ndez-Villamor, Carlos A. Iglesias, Mercedes Garijo,"MICROSERVICES: LIGHTWEIGHT SERVICE DESCRIPTIONS FOR REST ARCHITECTURAL STYLE"

[2] "Concurrency in Go and Java: Performance Analysis" by Naohiro Togashi Software Engineering Lab University of Aizu Aizuwakamatsu, Fukushima, Japan , Vitaly Klyuev Senior Associate Professor University of Aizu Aizuwakamatsu, Fukushima, [978-1-4799-4808-6 /14/ ©2014 IEEE]

[3] Fahim Shariar Shoumik,Md. Ibna Masum Millat Talukder,Ahmed Imtiaz Jami,Neeaz Wahed Protik,Md. Moinul Hoque," Scalable Micro-Service based Approach to FHIR Server with Golang and No-SQL", 2017 20th International Conference of Computer and Information Technology (ICCIT), 22-24 December, 2017

[4] G.Alonso ,D. Agrawal ,A. El Abbadi,,C. Mohan ," Functionality and Limitations of Current Workflow Management Systems"

[5] A. Koschel, I. Astrova and J. Dötterl, "Making the move to microservice architecture,"2017 International Conference on Information Society (i-Society), Dublin,2017,pp. 74-79. doi: 10.23919/i-Society.2017.8354675

[6] E. Djogic, S. Ribic and D. Donko, "Monolithic to microservices redesign of event driven integration platform," 2018 4 1st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),Opatija, 2018, pp. 1411-1414. Doi: 10.23919/MIPRO.2018.8400254

[7] ” An Introduction to Programming in Go” Copyright c 2012 by Caleb Doxsey

[8] The Go Programming Language Official Website, http://golang.org

[9]” Node.js vs Golang: Battle of the Next-Gen Languages” article by Jacob Nicholson