**微服务架构设计**

|  |  |
| --- | --- |
| 版本 | V0.01 |
| 公司： | 杭州一共享科技有限公司 |
| 创建人 | 刘勇俊 |
| 联系电话 |  |
| 机密等级 |  |
|  |  |
|  |  |

版本历史

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 开始日期 | 结束日期 | 作者 | 审核 | 内容 |
| V0.01 | 20191223 | 2019122314 | 刘勇俊 |  | 第一 二三四五六 七八九章 初稿 |
| V0.01 | 20200113 | 20200113 | 刘勇俊 |  | 第九章 第十章 修改 |
| V0.02 | 20200113 | 20200324 | 刘勇俊 |  | 第八章 第九章 修改目录结构  第十一章 增加网关接口 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. **缘由**

公司的现有业务使用的php 加jqury实现的,属于典型的传统架构,现有系统存在如下的弊病:

1. 前后端耦合在一起,不利于前端和后端责任分工,不利于开发,常出现后端等待前端出UI的情况.
2. 后端功能没有明确划分模块,耦合在一起, 代表后端开发人员必须要下载或者掌握全部框架代码, 后端人员之间分工不明确,不利于项目开发.
3. 后期使用 c 开发部分动态链接库,供golang调用,以达到加解密系统. 屏蔽代码外泄风险.
4. **目的**
5. 需要使用新的技术,实现前后端分离.提高开发效率.
6. 引入新的技术架构实现后端功能的解耦合.
7. **选型**
8. 语言选型

通常项目中使用较多的语言是java php golang.

在安全方便,golang是编译型语言,直接编译成可执行文件,java 是编译成字节码类型的中间代码,介乎编译型和解释性语言之间,php 是纯编译型语言,在安全工作做的不好的网站,时常会爆出php源码,但即使安全再差,java也只是显示些字节信息,golang 则完全不会显示. 故语言层次上 安全性 golang > java >php

在性能方面,go 语言的性能是比肩java, 超越php的.

在功能java 拥有更好的类和继承机制,更适合做大型的项目. 但难度比php和go都要高.

Golang是多核时代为并发而生的语言,在并发上处理性能,比java和php都要优越,gorutine 机制,一台机器可以有上万的协程.

Golang 天生语言简洁,公司大部分开发人员是php,转golang成本很小,又能够兼顾性能和安全,故选择golang.

1. 传统架构与微服务

仅仅使用前后端分离,可以使前端分离出来,后端代码依然会有比较大的耦合,而且不利于其他新的项目接入现有系统,

使用微服务,将后端项目划分不同的功能模块,后端人员的开发解耦合,各小组之间的工作变得独立,项目也将有更好的扩展性.

故需要选择golang 微服务做为现有的架构.

1. Go 微服务

Go-kits: 开发团队为3个国外的大牛,github 有15K starts. 没有很好的开发文档,有比较完善的example ,但是是全英文文档.

其实现是使用,写中间键来添加微服务组件,使用的装饰者模式,继承原服务,实现endpoint结构, 每加一个功能需要写一个中间键.来实现.

Go-chassis :华为基于server mesh 实现的微服务架构,拥有完善的中文文档和中文example, 其架构使用sidecar 模式.实现了数据控制和数据逻辑的解耦合, 通过可配置的ymal 文件,动态配置微服务空间, 另外其微服务中的负载均衡组件实现ab 部署功能.其sidecar 部分代码是使用责任链模式做的架构.

根据公司现有团队情况,故Go-chassis 符合当前公司的选型..

1. 团队

公司团队一直使用php,虽然php转golang不会有太大问题, 但存在团队成员是否愿意转型的问题?

1. **拓扑图**

初稿 :



1. **Go 安装**
2. **Go-chassis开发调试**

项目设置

1. Go 环境变量

Win 系统执行:

set GOARCH=amd64

set GOBIN=D:\gopro\bin

set GOCACHE=D:\gopro\gocache

set GOEXE=.exe

set GOFLAGS=

set GOHOSTARCH=amd64

set GOHOSTOS=windows

set GOOS=windows

set GOPATH=D:\gopro

set GOPROXY=https://goproxy.io

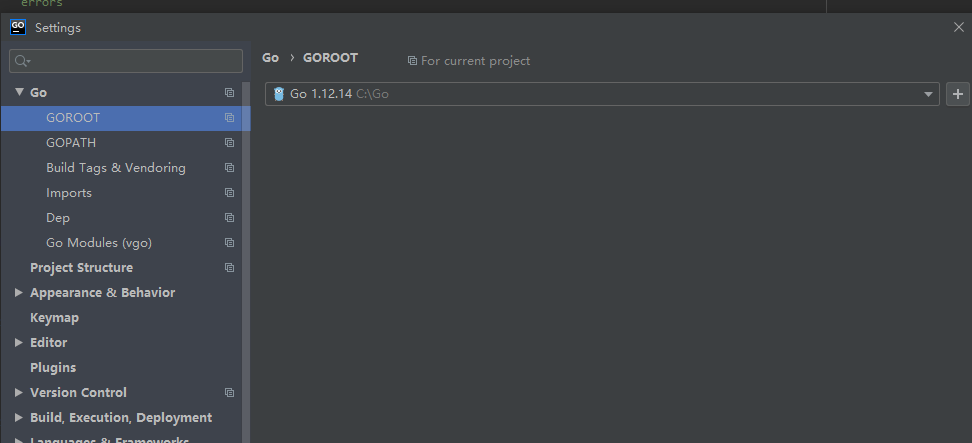
set GORACE=

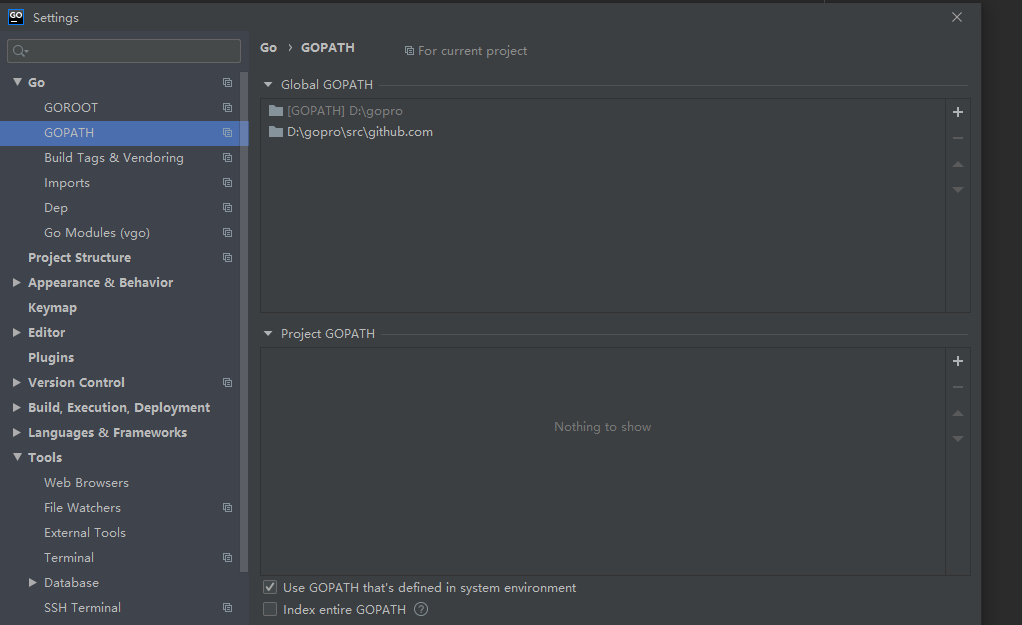
set GOROOT=C:\Go

set GOTMPDIR=

set GOTOOLDIR=C:\Go\pkg\tool\windows\_amd64

1. 下载goland,并且安装好.
2. 进入 GOPATH 目录下面 Git clone https://github.com/superufo/go-chassis.git
3. Goland 工程设置





1. 查看env | findstr "GO111MODULE"

设置 GO111MODULE=auto

Demo 运行

1. 安装docker .
2. 进入项目下面的example
3. 运行 docker-compose up
4. 进入项目下面的example\rest\server, 撰写 run.bat

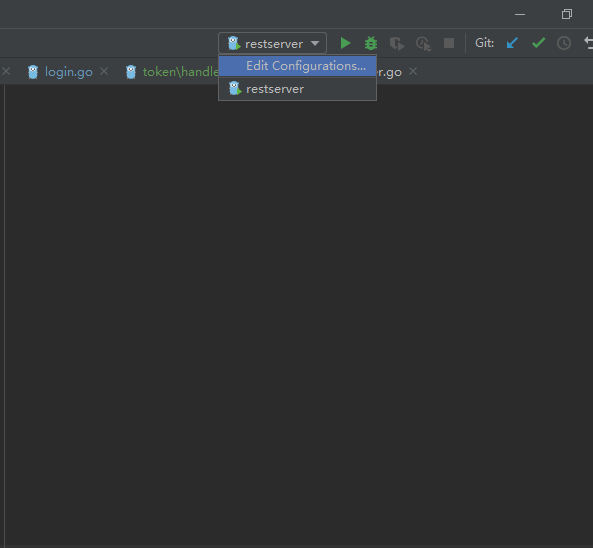
在 命令行运行 run.bat win. 看到相关的信息.

@echo off  
echo %1  
  
set *os*=%1  
set *ChassisConfDir*=D:\\gopro\\src\\github.com\\go-chassis\\go-chassis\\examples\\rest\\server\\conf  
echo *%ChassisConfDir%*set *CHASSIS\_HOME*=D:\\gopro\\src\\github.com\\go-chassis\\go-chassis\\examples\\rest\\server  
echo *%CHASSIS\_HOME%*@echo off  
if /i "*%os%*"=="linux" (  
 SET *CGO\_ENABLED*=0  
 SET *GOOS*=linux  
 SET *GOARCH*=amd64  
 go build -o server main.go  
 server  
) else (  
 SET *CGO\_ENABLED*=0  
 SET *GOOS*=windows  
 SET *GOARCH*=amd64  
 go build -o server.exe main.go  
 server.exe  
)  
  
echo success

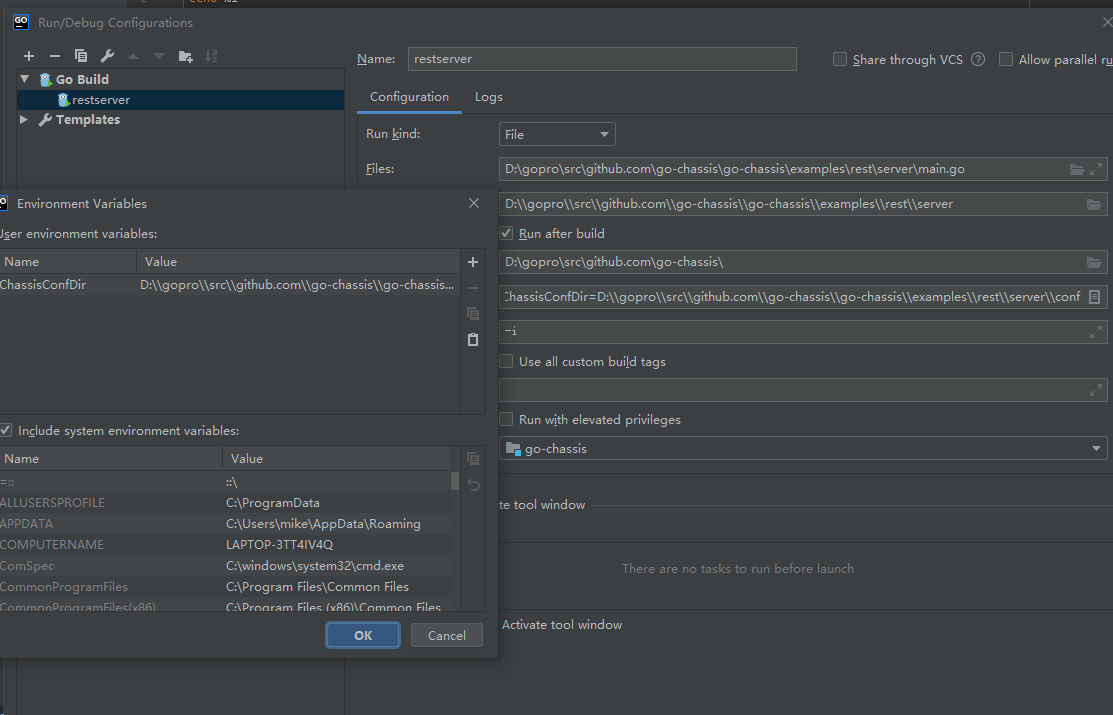
1. 进入项目下面的example\rest\client 建立同样的 run.bat 运行run.bat 即ok

Demo debug

1. 安装docker .
2. 进入项目下面的example
3. 运行 docker-compose up
4. 在 工具栏



配置微服务目录如下:



然后点击 “ ” debug 即可. 这样server 端 debug 即运行起来.

1. 进入项目下面的example\rest\client 建立同样的 run.bat 运行run.bat 即ok, golang 会自动定位 断点处. 调试者可以用 F5 F6 F7 来执行相关的操作. 也可以在下面窗口看调试信息.
2. <http://127.0.0.1:30103/#!/sc/topology> 打开这个链接既可以查看服务的情况.
3. <http://127.0.0.1:9411/zipkin/> 打开这个可以查看链路追踪情况

grpc go get -u github.com/golang/protobuf/protoc-gen-go

1. **Go-chassis 源码分析**
2. 注册



1. 初始化流程



1. 服务注册结构
2. 服务注册结构关系



1. 服务注册结构

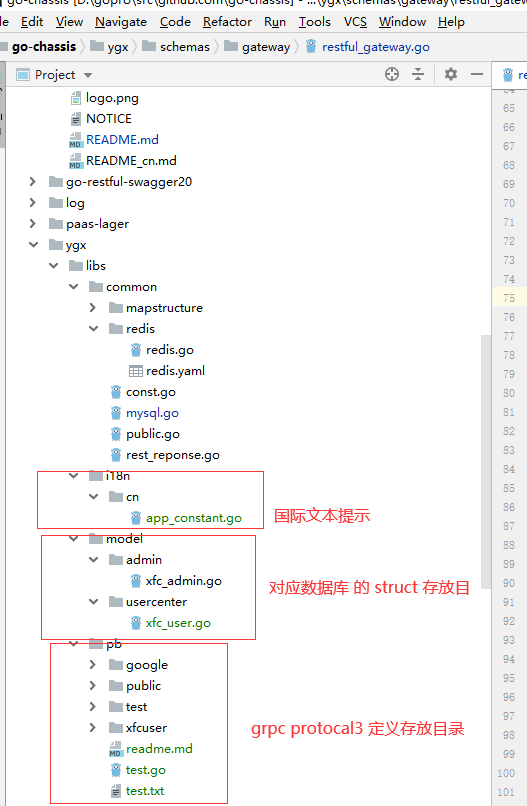


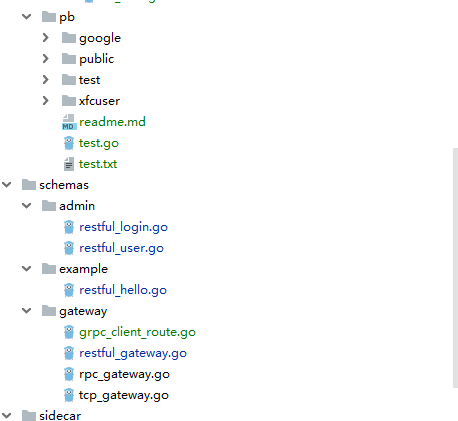
1. chassis结构

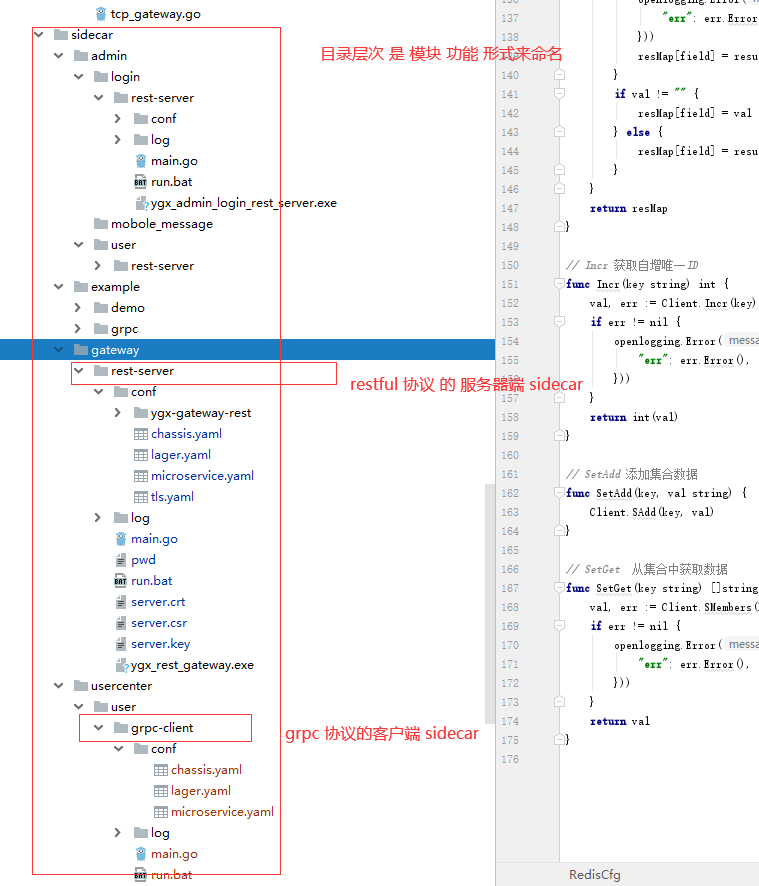


1. **壹共享源码结构**
2. 目录结构

此目录结构作为起始的第一版,不排除后期更好的方案.







schemas 存放所有的微服务, 微服务分组, 现在已经写好了admin 登陆和获取admin详情,分别放在admin 文件夹下面.

common文件夹存放微服务中使用的公共组件和变量.如数据库链接,返回结构的定义等.

model 文件夹存放分组的数据库映射表结构, 物流表放物流表一组,admin 存放账号信息为一组.

sidecar 目录存放微服务的配置和驱动程序.

里面按业务划分分别存放不同业务的微服务程序.

预定所有微服务均为restful 通信. 请求均为json 数据格式. 返回的响应为

{

“status”:

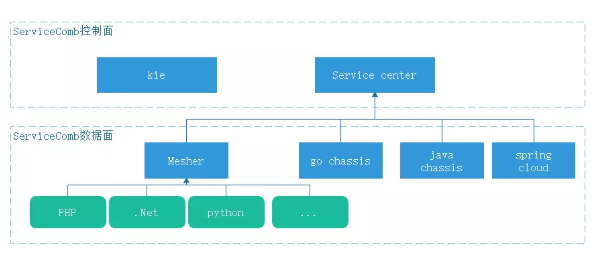
“data”: {}

“message”:””

}

1. 壹共享授权
2. **集成微服务**

本系统使用了 serviceComb 可以使用Mesher（go 工具）持续集成其他语言作为微服务。使用Mesher对其他语言可以做到零侵入成为微服务。



* 1. 与php微服务集成

使用 [**apache-servicecomb-mesher**](http://mirror.bit.edu.cn/apache/servicecomb/servicecomb-mesher/1.6.3/apache-servicecomb-mesher-1.6.3-src.tar.gz) 组件做sidecar

Demo:

详见 ygx/vm/php-microserver 源码

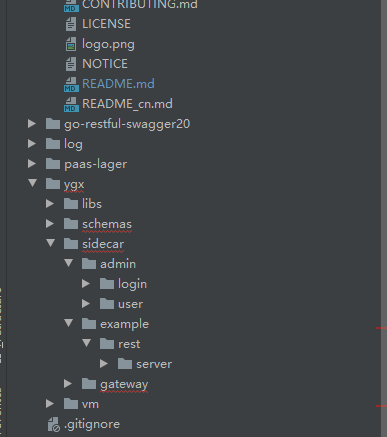
调用ygx/schemes/example/restful\_hello.go CallPhp方法

* 1. 与java微服务集成

Demo: 暂无

1. **壹共享开发约成**
   1. 微服务命名

为方便知道微服务功能查询和存放位置,以存放的文件层次(项目\_模块\_功能),间隔”\_”,来命名,如:

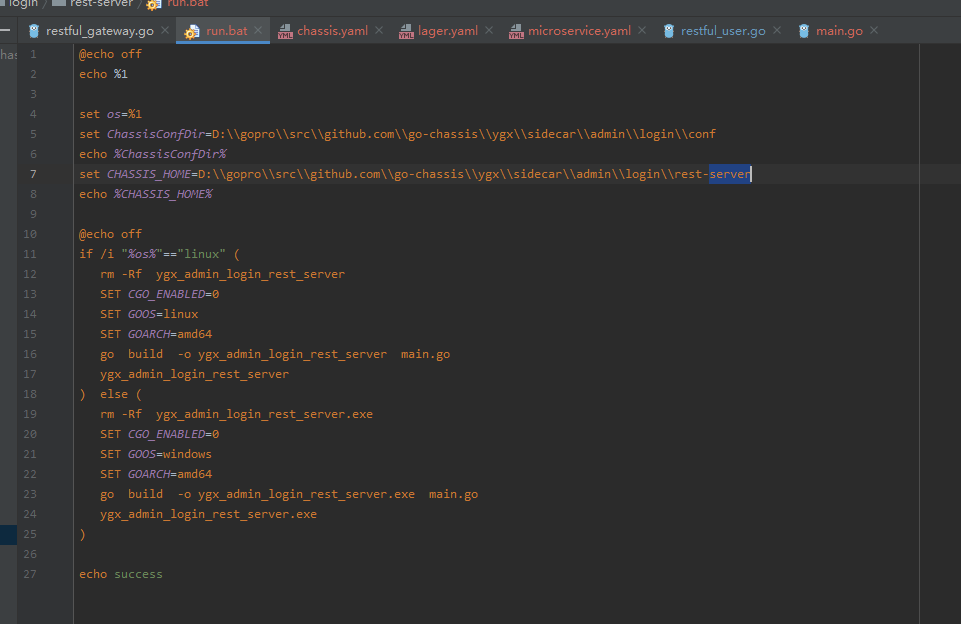


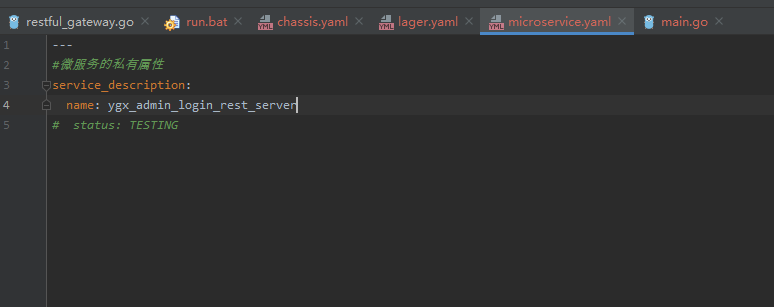
如ygx/sidecar/admin/user/rest-server 表示壹共享工程的后台的用户基于rest的api 接口.

编译后的可执行文件统一为: ygx\_admin\_user\_rest\_server 或者win 为ygx\_admin\_user\_rest\_server.exe

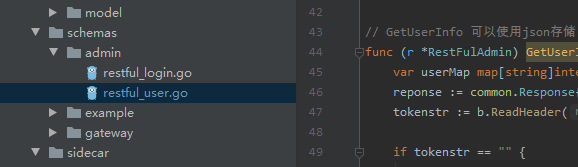
如grpc协议,可以使用ygx/sidecar/admin/user/grpc-server

ygx\_admin\_user\_grpc\_server 或者win 为ygx\_admin\_user\_ grpc \_server.exe



按上面规则, microservice.yaml中配置的微服务命名定义为如下 (请忽略sidecar文件夹): 

* 1. Shcmes 下命名



以模块文件夹命名为协议 rest 协议restful\_功能.go 如果为grpc 协议则命名为grpc\_功能.go 如 grpc\_user.go

* 1. 端口分配

提前规划好端口端分配：

网关微服务端 端口号区间： 23600 – 23900

公用微服务端 端口号区间:

app 前端调用微服务端口号区间：

后台 调用微服务端口号区间：

1. **壹共享微服务接口**

接口定义请查看 <http://192.168.11.137:4999/web/#/1?page_id=1> 此处不在提供

1. **登陆接口**

|  |  |
| --- | --- |
|  |  |
| 开发人员 | 刘勇俊 |
| 维护人员 |  |
| 微服务名称 | 登陆接口 |
| 存放地址 | ygx/sidecar/admin/login/server/\*  ygx/schemas/admin/restful\_login.go |
| 依赖微服务 | 无 |
| 请求测试url | http://127.0.0.1:5003/login |
| 正式路径: |  |
| 请求方式 | Post |
| 请求数据格式 | Json |
| 请求参数 | UserName 用户名  Password 密码  例如:  {  "UserName": "admin",  "Password": "a4abfed2ab6f5bf97b2c56434e2f0695"  } |
| 请求数据头 | 无 |
| 返回数据 | Status 0 成功 1 失败  Data user返回的token  Message 返回的信息  例如:  {      "status": 0,      "Data": {          "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  eyJleHAiOjE1Nzg2MjY2NjIsImlzcyI6IllneCBsaW1pdGVkIiwibm  JmIjoxNTc4MDIwODYyLCJ1c2VyX2lkIjowLCJ1c2VyX25hbWUiOiIiL  CJlbWFpbCI6IiIsIm1vYmlsZSI6IiIsInJvbGVfaWQiOjB9.h7RwTY  UFyNC0s\_6fPZ0hjjRhEpyvqvLNqL1PDYe3tW8"      },      "message": ""  } |

1. **获取admin 用户信息**

|  |  |
| --- | --- |
|  |  |
| 开发人员 | 刘勇俊 |
| 维护人员 |  |
| 微服务名称 | 登陆接口 |
| 存放地址 | ygx/sidecar/admin/user/server/\*  ygx/schemas/admin/restful\_user.go |
| 依赖微服务 | 无 |
| 请求测试url | http://127.0.0.1:5004/getInfo |
| 正式路径: |  |
| 请求方式 | Post |
| 请求数据格式 | Json |
| 请求参数 | 无 |
| 请求数据头 | Token  例如:  Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  eyJleHAiOjE1Nzg2MjY2NjIsImlzcyI6IllneCBsaW1pdGVkIiwibm  JmIjoxNTc4MDIwODYyLCJ1c2VyX2lkIjowLCJ1c2VyX25hbWUiOiIiL  CJlbWFpbCI6IiIsIm1vYmlsZSI6IiIsInJvbGVfaWQiOjB9.h7RwTY  UFyNC0s\_6fPZ0hjjRhEpyvqvLNqL1PDYe3tW8 |
| 返回数据 | Status 0 成功 1 失败  Data token 返回的token  Message 返回的信息  例如:  {  "status": 0,  "Data": {  User\_info:{  "admin\_id": 1,  "user\_name": "admin",  "dep\_id": 0,  "pos\_id": 0,  "mobile": "13549748887",  "email": "admin@ioa365.com",  "password": "519475228fe35ad067744465c42a19b2",  "ec\_salt": "",  "add\_time": 1513675064,  "last\_login": 0,  "last\_ip": "",  "nav\_list": "",  "lang\_type": "",  "agency\_id": 0,  "suppliers\_id": 0,  "todo\_list": "",  "role\_id": 1,  "is\_lock": 1,  "is\_sales": 0,  "is\_govsales": 0  }  },  "message": "获取用户信息成功"  } |

1. Restful 后台的网关接口 （原则上一个网关 内部通信只使用 rest 或 grpc 中一种）

|  |  |
| --- | --- |
|  |  |
| 开发人员 | 刘勇俊 |
| 维护人员 |  |
| 微服务名称 | 后台的网关接口 |
| 存放地址 | ygx/sidecar/gateway/rest-server/\*  ygx/schemas/gateway/restful\_gate.go |
| 依赖微服务 | 无 |
| 请求测试url | https://127.0.0.1:23600/ygx/app |
| 正式路径: |  |
| 请求方式 | Post |
| 请求数据格式 | Json |
| 请求参数 | {"email":"admin@ioa365.com"} |
| 请求数据头 | Token  例如:  Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  eyJleHAiOjE1Nzg2MjY2NjIsImlzcyI6IllneCBsaW1pdGVkIiwibm  JmIjoxNTc4MDIwODYyLCJ1c2VyX2lkIjowLCJ1c2VyX25hbWUiOiIiL  CJlbWFpbCI6IiIsIm1vYmlsZSI6IiIsInJvbGVfaWQiOjB9.h7RwTY  UFyNC0s\_6fPZ0hjjRhEpyvqvLNqL1PDYe3tW8  Content-Type application/json  Service 协议：method：微服务名称: 微服务里面的方法：版本  协议支持 rest 和 grpc 例如：  rest:post:ygx\_admin\_user\_rest\_server/updateUser:0.0.1  grpc:ygx\_usercenter\_user\_grpc-server/UserLogin:0.0.1 |
| 返回数据 | Status 0 成功 1 失败  Data token 返回的token  Message 返回的信息  例如:  {      "status": 0,      "data": {          "user\_info": {              "add\_time": 1513675064,              "admin\_id": 3,              "agency\_id": 0,              "dep\_id": 0,              "ec\_salt": "",              "email": "admin@ioa365.com",              "is\_govsales": 0,              "is\_lock": 1,              "is\_sales": 0,              "lang\_type": "",              "last\_ip": "",              "last\_login": 0,              "mobile": "13549748887",              "nav\_list": "",              "password": "7b0879b7e88eeb2ce34bd36e10292a70",              "pos\_id": 0,              "role\_id": 1,              "suppliers\_id": 0,              "todo\_list": "",              "user\_name": "admin"          }      },      "message": "更新用户信息成功"  } |

1. restful前端的网关接口

|  |  |
| --- | --- |
|  |  |
| 开发人员 | 刘勇俊 |
| 维护人员 |  |
| 微服务名称 | 后台的网关接口 |
| 存放地址 | ygx/sidecar/gateway/rest-server/\*  ygx/schemas/gateway/restful\_gate.go |
| 依赖微服务 | 无 |
| 请求测试url | https://127.0.0.1:23600/ygx/admin |
| 正式路径: |  |
| 请求方式 | Post |
| 请求数据格式 | Json |
| 请求参数 | {"user\_name":"米奇", "passwd":  "519475228fe35ad067744465c42a19b2"} |
| 请求数据头 | Token  例如:  Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  eyJleHAiOjE1Nzg2MjY2NjIsImlzcyI6IllneCBsaW1pdGVkIiwibm  JmIjoxNTc4MDIwODYyLCJ1c2VyX2lkIjowLCJ1c2VyX25hbWUiOiIiL  CJlbWFpbCI6IiIsIm1vYmlsZSI6IiIsInJvbGVfaWQiOjB9.h7RwTY  UFyNC0s\_6fPZ0hjjRhEpyvqvLNqL1PDYe3tW8  Content-Type application/json  Service 协议：method：微服务名称: 微服务里面的方法：版本  协议支持 rest 和 grpc 例如：  rest:post:ygx\_admin\_user\_rest\_server/updateUser:0.0.1  grpc:ygx\_usercenter\_user\_grpc-server/UserLogin:0.0.1 |
| 返回数据 | Status 0 成功 1 失败  Data token 返回的token  Message 返回的信息  例如:  {  "status": 0,  "data": {  "userReply": {  "user\_id": 1,  "user\_name": "米奇",  "passwd": "519475228fe35ad067744465c42a19b2",  "nick": "米奇先生",  "mobile": "18699985000",  "email": "18699985000@139.com",  "post": "总监",  "team\_id": 1,  "team\_name": "one food",  "introduce": "介绍自己",  "department": "项目一组",  "department\_id": 1,  "balance": 50000,  "freeze\_balance": 500,  "real\_balance": 25000,  "role\_id": 1  },  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAi  OjE1ODUwMzU5NDYsImlhdCI6MTU4NDk0OTU0NiwiaXNzIjoiSGFuZ3  pob3UgWWd4IExpbWl0ZWQgQ29tcGFueSIsIm5iZiI6MTU4NDk0ODU0  Niwic3ViIjoiWWd4IFVzZXIgQXV0aCIsInVzZXJfaWQiOjEsIlBhc  3NXZCI6IjUxOTQ3NTIyOGZlMzVhZDA2Nzc0NDQ2NWM0MmExOWIyI  iwidXNlcl9uYW1lIjoi57Gz5aWHIiwiZW1haWwiOiIxODY5OTk4N  TAwMEAxMzkuY29tIiwibW9iaWxlIjoiMTg2OTk5ODUwMDAiLCJyb  2xlX2lkIjoxfQ.pIm5h0psHFOmSYJM81nhseEEVFpqU7a6uHQia  SR4DI"  },  "message": ""  } |

1. 127.0.0.1:5004/metrics
2. **Swagger**

如果是 restful 系统会自动生成 swagger, 拷贝微服务下面的schemas 下面yaml .可以到 <http://editor.swagger.io/> 网站，产看api 详情。

1. **todo**

tcp 网关接口

java微服务集成