2024.1.8

- **做了**
    1. 容器和容器管理的学习
    2. go技术点:
        a. beego和gin的学习：beego简介
        b. gorm学习：GORM，另外，GORM 通过 `WithContext` 方法提供了 Context 支持
        c. gen：
            i. gorm/gen
            ii. GORM-GEN快速上手_gorm gen-CSDN博客
    3. go知识点:
        a. 接口interface底层：
            i. Go 语言中的 nil
            ii. 静态类型与动态类型_go语言静态类型
            iii. 只有接口变量的动态类型和其值都是nil，才能被认为是nil
        b. 控制结构-switch：任一分支的测试结果先为 true 时，该分支的代码会被执行

```
switch {
    case condition1:
        ...
    case condition2:
        ...
    default:
        ...
}
```

        c. 通道:

> - `make(chan string)` 表示创建了一个字符串类型的通道。
> - 使用 <- 运算符从通道接收值时，如果通道中没有值，接收操作将会被阻塞，直到通道中有值为止。这是无缓冲通道的行为。
> - 如果通道中有多个值，每次接收操作将从通道中取出一个值。换句话说，无缓冲通道保证发送和接收的同步性，每个值都是独立的。
> - 使用 `for result := range resultChan` 的形式是用于遍历通道的一种特殊语法

```
func main() {
    resultChan := make(chan string)
    go xxx(resultChan)
    for result := range resultChan {
        fmt.Println("Main: Received result:", result)
    }
}
```

- **计划**
- **遇到的问题**

- beego和gin有什么区别么？（另外，beego可以不使用模板引擎和视图）

> 1. **框架设计和哲学：**
>   - **Beego：** Beego 是一个全功能的 Web 框架，包含了很多内置的功能和工具，例如 ORM、模板引擎、国际化支持等。它设计上更注重提供一站式的解决方案，使得开发者可以在一个框架中获得各种功能。
>   - **Gin：** Gin 是一个轻量级的 Web 框架，它的设计目标是提供最小的框架和最快的性能。Gin 的哲学是尽量保持简单和快速，只包含了一些基本的功能，更多的功能可以通过中间件扩展。
> 2. **性能：**
>   - **Beego：** Beego 提供了许多功能，但相对而言可能在性能上不如 Gin 那么高效。
>   - **Gin：** Gin 被设计成非常轻量级，因此在性能方面通常较为出色。
> 3. **中间件的处理：**
>   - **Beego：** Beego 中间件的使用较为简单，它提供了一些内置的中间件，也支持自定义中间件。
>   - **Gin：** Gin 中间件的使用更为灵活，可以更细粒度地控制中间件的顺序和作用范围。
> 4. **路由定义：**
>   - **Beego：** Beego 的路由定义较为灵活，支持正则表达式等高级路由特性。
>   - **Gin：** Gin 采用了类似于 Express.js 的路由定义方式，使得路由配置更为直观和简单。
> 5. **模板引擎：**
>   - **Beego：** Beego 自带了模板引擎，支持自定义模板函数和模板布局。
>   - **Gin：** Gin 并没有自带模板引擎，而是鼓励使用现有的模板引擎或前端框架。
> 6. **社区和文档：**
>   - **Beego：** Beego 有一个较为活跃的社区，并且有相对完善的文档。
>   - **Gin：** Gin 也有不少用户，虽然相对较年轻，但在一些场景中已经取得了很好的应用。
>
> 综合考虑，选择使用 Beego 还是 Gin 取决于项目的需求和开发者的偏好。如果需要一个全功能的框架，并且愿意牺牲一些性能以换取更多的内置功能，那么 Beego 可能是一个不错的选择。如果更注重性能、简洁和更灵活的中间件管理，那么 Gin 可能更适合。

- **笔记**
  - Gorm基础教程
  - docker-compose的介绍：Docker三剑客之docker-compose
  - advantage of Docker with k8s：Docker不香吗？为什么还要使用K8s？
  - introduce of k8s：K8s是什么
  - k3s和k8s的区别：k3s和k8s的区别和优缺点_k3s和k8s的功能对比

2024.1.9

- **做了**

1. go知识点：
   a. 结构+接口：协程间的信道
   b. go的赋值写法

c. go的封装继承多态(结构&接口)：GO语言中封装，继承，和多态
d. go程是协作式的。这意味着在 Go 语言中，协程之间的切换是由程序员明确地进行的，而不是由操作系统强制执行的。

2. 负载均衡&高并发的了解

3. gen学习&实践

a. 自动**生成模型结构体和自定义sql语句**(方法)

i. 官网：GORM
ii. 简单使用：GORM Gen使用指南 (qq.com)
iii. 具体使用（但是排版差）：GORM-GEN快速上手_gorm gen-CSDN博客
iv. ⭐后端 - GEN 自动生成 GORM 模型结构体文件及使用示例 - 个人文章 - SegmentFault 思否
v. 详细讲解：GORM 强大的代码生成工具 —— gorm/gen - 掘金 (juejin.cn)

b. 以eatate项目一数据表为例，重构部分结构与方法

c. 自动生成代码的结构:

```
.
├── cmd
│   └── generate
│       └── main.go
├── dal
│   ├── model
│   │   └── user.gen.go
│   └── query
│       ├── gen.go
│       └── user.gen.go
├── go.mod
├── go.sum
└── main.go
```

4. 小结gorm和gen的区别，gen好在哪?

a. 少些点东西,哈哈哈，还能自动生成，美滋滋（缺点：可视化太差劲了）

- **计划**
  - 想要重构
  - 细看协程底层，彻底搞清楚协程工作原理
- **笔记**

1. go的:=

a. GO语言中=和:=的区别

b. go = :=

c. golang中var、make、new、:= 的使用

d. 10.1. 结构体定义

e.
```go
/*下面的都是等价的指针初始化写法*/
// 1
pers1 := new(Person)
// 2
pers2 := &Person{}
// 3
var pers3 *Person
```

```
pers3 = new(Person)
// 4
var pers4 *Person = new(Person)
// 5
var pers5 = new(Person)
// 6
var pers6 *Person
pers6 = &Person{}
/*初始化写法,注意不能用xxx.A xxx.B*/
type Interval struct {
    start int
    end   int
}
intr1 := Interval{0, 3}            // (A)
intr2 := Interval{end:0, start:3}  // (B)
intr3 := Interval{end:3}           // (C)
intr4 := Interval{}  intr4.end = 0  intr4.start = 3 // (D)
```

2024.1.10

- **做了**
  1. go知识点:
     a. 协程的细节
     b. **结构体变量初始化**和**结构体内部**的内存布局:
        i. 结构体定义
        ii. go-结构体内存布局 - 浪客禅心 - 博客园 (cnblogs.com)
     c. 仓库byte.Buffer的使用：GO Buffer.Read用法及代码示例
  2. go技术点:
     a. 打印变量地址（注意Printf和Println的使用）：如何在Go中打印结构变量的地址
- **计划**
  - 想要看重构项目QAQ，主要是暂时不知道干什么
- **遇到的问题**
  - buffer报错：panic: runtime error: invalid memory address or nil pointer dereference
    - 因为没给指针初始化，只声明，使用报错nil
  - ⭐**byte.buffer.ReadByte()底层多协程读问题**
    - buffer.ReadByte()它不是原子操作，所以两个协程同时进行这个操作的时候，就容易互相影响，可能出现同时读到同一个byte，或是读到不同的byte，后者会覆盖前者的b.off值(啊啊啊啊啊啊啊啊啊阿巴阿巴阿巴阿巴阿巴啊啊啊啊啊阿)
  - buffer报错：fatal error: sync: unlock of unlocked mutex gorout
    - go Mutex和RWMutex_fatal error: sync: unlock of unlocked mutex gorout
- **笔记**
  - 协程的细节
    - GPM调度模型
      - GPM介绍：go 怎么等待所有的协程完成_谈谈Go语言的协程
      - GPM实例&细节：关于Go协程的一些理解_go若干协程如果一个协程oom
    - runtime.GOMAXPROCS():
      - 你不知道的runtime.GOMAXPROCS(1)

        - Go语言GOMAXPROCS（调整并发的运行性能）
    - 协程的读锁：
        - 可以选择性的添加写锁或读锁，如果是读锁，那意味着可以同时读取，**可能会相互影响**:比如`byte.buffer.ReadByte()`
        - 不控制cpu数量的话，默认物理线程数量（最多有多少个Goroutine可以**并行**执行）为CPU数量(Go 1.5 版本开始默认执行`runtime.GOMAXPROCS(runtime.NumCPU())`)
        - 注意：读写锁的写锁一般没有wait，一般是`通知`、`有个全局的变量来标志第一个协程数据是否接受完毕，剩下的协程，反复检查该变量的值，直到满足要求`、`或者创建多个channel，每个协程阻塞在一个channel上，由接收数据的协程在数据接收完毕后，逐个通知`

2024.1.11

- **做了**
    1. 把重构的前后项目跑起来
    2. 技术—调试：
        a. **用dlv调试**代码：
            i. Golang程序调试工具介绍(gdb vs dlv) - sunsky303 - 博客园 (cnblogs.com)
            ii. go调试工具，协程调试 - 方东信 - 博客园 (cnblogs.com)
    3. 实践—**远程调试**代码（利用dlv）：
        a. Goland结合dlv进行调试 dlv --listen=-CSDN博客，注意：运行的时候，需要选择可执行文件，第一个命令可以不用，可以找到main就好了
- **计划**
    - 熟悉重构项目业务，细看代码
    - 思考自己能否实现
    - 思考有没有更好的方式实现
    - 思考项目整体是怎么设计的/该怎么设计
    - 多余时间看其他技术(比如：go路线中的技术、提升项目的性能技术等等)
- **遇到的问题**
    - CHAR, CHARACTER, CHARSET or COLLATE expected, got 'ENCRYPTION'
        - 不用管，直接运行
- **笔记**
    - goland插件：goland必须要装的插件
    - 项目架构图怎么设计：手把手教你画产品架构图

2024.1.12

- **做了**
    - 学习了怎么制作mock sql数据
- **计划**
    - 熟悉重构项目业务，细看代码
- **笔记**
    - 多环境:
        - Go 多环境下配置管理方案
        - spring boot 切换（dev 、test 、prod）环境 springboot没有指定dev\prod\test

- mock数据自动生成:
  - 基于mockaroo快速生成随机测试数据地址:Mockaroo - 随机数据生成器和 API 模拟工具 | JSON / CSV / SQL / Excel
  - Java开源工具库使用之虚假数据生成库datafaker
  - 五花八门的前端 Mock 数据方案，我是如何选择的？ - 知乎 (zhihu.com)
  - 前端MOCK数据
  - 将json数据转为sql的insert语句_go sqlmock怎么写数据库插入客
  - gomonkey使用
- 正则(批量修改标题等级):
  - 使用正则表达式对Typora中的全部标题升降级
  - 正则表达式-菜鸟
  - 常用正则表达式大全—包括校验数字、字符、一些特殊的需求等等
- 安卓脚本：（真香）
  - 资料：解放双手，手机自动化神器-AutoJS的使用
  - 官网：autoxjs.com
  - autojs悬浮窗模拟toast气泡, 放到屏幕底部居中 - 知乎 (zhihu.com)
  - 做了:
    - 帮我刷pdd视频的脚本
    - 帮我自动跳过部分软件的广告
  - 计划:
    - 做帮我破译B站VIP的番剧
    - 做帮我打音游的脚本
    - 做图书馆抢座位的脚本

周小结

关键词：**go协程、gorm(gen)、远程调试、结构体、byte.buffer、重构项目学习**

做了:

1. go知识点和技术点的进一步理解
2. 实践goland远程调试go程序(设置远程ip为127.0.0.1)
3. 重构项目前后端的代码的启动
4. 学习了怎么自动生成mock数据库数据
5. 次要的:

    a. 学习并实践安卓脚本的制作(让我对js语法更加熟悉，也看到了更多go和其他语法的相同点)

计划:

- 熟悉重构项目业务，细看代码
- 学习缓存，日志等技术点
- 继续同时补缺补漏