# Entry\_Task 项目提交报告

## 

## 报告人：周玮奇

### 项目需求

### 1.1 项目背景

完成Entry\_Task作为go语言编程技巧的学习成果检验。

### 1.1 功能需求

实现一个用户管理系统，用户可以登录、拉取和编辑他们的profiles。

用户可以通过在Web页面输入username和password登录，backend系统负责校验用户身份。成功登录后，页面需要展示用户的相关信息；否则页面展示相关错误。

成功登录后，用户可以编辑以下内容：

上传profile picture

修改nickname（需要支持Unicode字符集，utf-8编码）

用户信息包括：

username（不可更改）

Nickname

profile picture

### 1.2 设计需求

分别实现HTTP server和TCP server，所有功能逻辑放在TCP server实现

用户账号信息必须存储在MySQL数据库。

使用基于Auth/Session Token的鉴权机制

日志必须规范，需要分级，并且单条请求可追溯

Web server不允许直连MySQL、Redis。所有HTTP请求只处理API和用户输入，具体

的功能逻辑和数据库操作，需要通过gRPC请求TCP server完成

鲁棒性

性能

### 1.3 设计需求

数据库必须有10,000,000条用户账号信息

必须确保返回结果是正确的

每个请求都要包含RPC调用以及Mysql或Redis访问

200并发（固定用户）情况下，HTTP API QPS大于3000

200个client，每个client模拟一个用户（因此需要200个不同的固定用户账号）

200并发（随机用户）情况下，HTTP API QPS大于1000

200个client，每个client每次随机从10,000,000条记录中选取一个用户，发起请求（如

果涉及到鉴权，可以使用一个测试用的token）

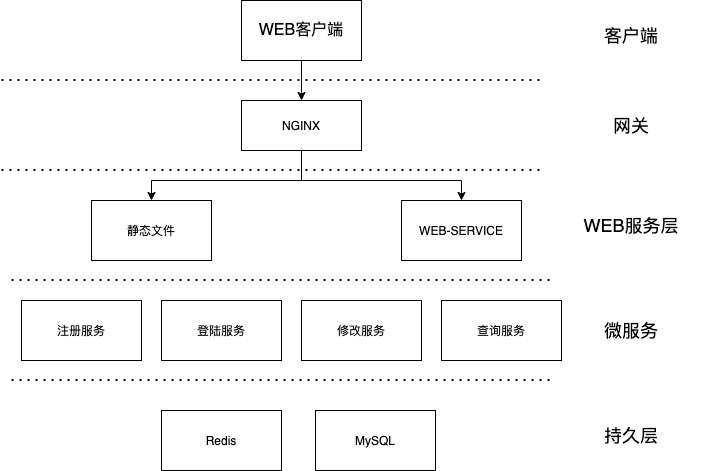
2000并发（固定用户）情况下，HTTP API QPS大于1500

2000并发（随机用户）情况下，HTTP API QPS大于800

### 项目概要设计

### 2.1 总体设计

项目总体架构图如下：



构架总体分为五层：

1. 客户端：用户使用的前端界面，负责与用户交互
2. 网关：使用Nginx作为网站的统一入口。Nginx主要负责1、将请求分发至WEB服务器。2、处理静态文件的请求
3. web服务器：负责提供接口，处理业务流程。
4. 微服务：将应用服务器提供业务所需要的服务。
5. 持久层：负责存取数据

### 2.2 客户端设计

设计需求：1.提供用户的交互界面

1. 对用户的操作保证安全。
2. 提供给用户登陆界面，注册界面、编辑界面、信息展示功能
3. 提供给用户反馈信息

功能实现

1、交互界面

提供给用户HTML页面，用户能够在界面上完成登陆，编辑，查看信息功能。

前端使用HTML静态化，减少请求次数。动态部分请求web服务器接口，前端页面的返回由nginx处理，将前后端分离。

1. 对用户的操作保证安全。

登陆成功后将用户签名作为TOKEN保存在COOKIE，用户进行操作时以TOKEN作为凭证进行权限检查。

1. 完成功能调用接口

完成登陆，编辑，注册功能的接口

/user/info 查询用户信息接口

/user/login 登陆接口，返回用户签名

/user/register 注册用户

/user/edit 编辑用户接口

1. 用户反馈信息

返回给用户的操作结果

200 操作成功

400 操作失败

500 由于程序问题产生错误

### 2.3 Nginx

设计需求：1.反向代理将请求分发给web服务器

2.处理对静态资源的请求。

功能实现：

通过配置nginx配置文件即可

### 2.3 WEB服务器

设计需求：1.作为接口服务器，提供给客户端调用接口

2.对请求进行权限检查，保证安全性

3.能够通过日志追踪每一条请求的完成状态，对错误能够追踪。

功能实现

1.使用gin框架完成WEB服务

2.使用权限检查中间件，作为权限检查

3.使用日志中间件

### 2.4 微服务

设计需求：1.提供给完成业务需要的服务

2.对性能有一定需求，能够处理并发

3.提供的服务必须可靠，不会产生错误

功能实现：

1.使用RPC协议链接WEB服务器，WEB服务器通过RPC协议调用服务。

2.由于测试数据到达千万级别，如果只使用MySQL作为存储，难以达到并发的性能需求。

使用Redis作为缓存服务器，将经常查询的数据存在Redis缓存中。

3.由于使用了Redis缓存与MySQL持久化必须保证两个数据库查询结果的一

致性。

### 2.5 持久层

设计需求：1.存储和读取数据

2.减少查询业务所需要的时间

功能实现：

使用Docker生成Redis与mysql的镜像即可。

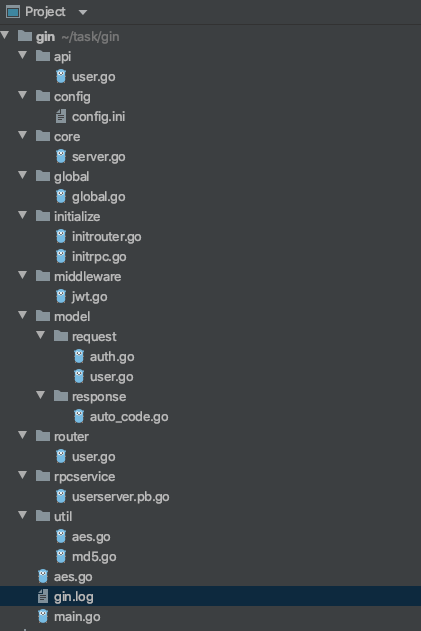
### 项目详细设计

### 3.1 WEB服务器

### 3.1.1 WEB服务器详细设计

WEB服务器使用gin框架作为WEB容器，WEB服务器仅提供面向客户端的业务请求接口。

### 3.1.2 WEB服务器目录结构



WEB服务器目录结构如上图所示下面解释目录作用

main.go作为程序启动入口

gin.log 日志存储文件

api 存储业务逻辑的包

Config 存储配置文件的包

Global 业务工具类，封装了指向rpc服务，日志服务的指针，当业务需要时可以引用global中的工具类

global.go

Model 存储模型的包，分为请求模型和返回模型

Router 存储了分级目录的router加载流程

Initialize 存储了组件的初始化流程

MiddleWare 存储了中间件

Util 存储了业务实现所需要的工具类

### 3.1.3 WEB服务器处理流程

gin框架对于注册的Router都设定了一个HandlersChain。处理请求将按照次序调用链中的Handler函数。

对于

flow

### 3.2 RPC服务器

### 3.2.1 WEB服务器详细设计

### 3.2.2 WEB服务器详细设计