

# 基于 B/S 结构的员工周报管理系统的设计与实现

## 摘要

如今企业总量和规模越来越大，员工的数量也与日俱增，无形中增加了管理成本，作为一个优秀团体的负责人和管理者，需要实时把控整个团队的状态和工作进度，以便做出相应的措施或者适时地分析，而员工周报系统十分适合此场景。传统的周报提交和汇总可能是员工手写，通过文件等方式留档，之后汇总需要人力部门依此统计相关数据，但是随着计算机的普及以及互联网的兴起，通过软件和数据库来代替原先低效易出错的模式逐渐成为企业管理的基本。

此课题设计实现的员工周报管理是基于 B/S 结构的系统，工程分为前端、后台、数据库三大部分，其中前后端分离，开发过程相互独立，最终通过 HTTP 接口进行前后端数据交互。前端采用蚂蚁金服旗下开源的基于 react 和 umi 框架的 Ant Design Pro 解决方案，后端使用 Golang 语言开发并配合 Beego 框架完成 web server 的业务逻辑，数据库使用 Mysql，部署方案使用 docker 容器维护以上三者。功能模块可以分为登录/注册、员工管理、每日工作提交/汇总、周报提交/汇总、年度计划排班表、个人信息管理六大模块，且实现员工权限区分、部门区分，可以满足企业个人考勤、周期汇总、年度安排等需求。系统界面交互性友好，功能清晰，可以分级分层使用，实现数据的有限查看范围，而且管理者可以实时查看汇总信息并做调整，并且周报、月报可以导出打印。

课题实现的管理系统从实际需求分析入手，逐步设计并实现主要切实的功能，可以实现工作的量化、进度的汇总、员工定期规划周期性输出等作用，而且根据汇总信息结合工作成果作为考核指标，一方面可以提升员工的自觉性，另一方面提升了员工评级的效率，方便企业用户合理高效地完成周报管理，加强管理效率，减小管理成本。

关键词：周报管理系统；企业管理；B/S 结构；

# DESIGN AND IMPLEMENTATION OF STAFF WEEKLY REPORT MANAGEMENT SYSTEM BASED ON B/S STRUCTURE

## Abstract

Nowadays, the total volume and scale of enterprises are increasing, and the number of staffs is increasing day by day. This increases the management cost invisibly. As the head and manager of an excellent group, it is necessary to control the status and progress of the entire team in real time in order to make the corresponding measures or timely analysis, and the staff weekly report system is very suitable for this scenario. The traditional weekly report submission and summary may be handwritten by staffs, filed by means of documents, etc., and then the summary needs the human resources department to count relevant data according to this, but with the popularity of computers and the rise of the Internet, replacing the original inefficient and error-prone model with software and database has gradually become the basis of enterprise management.

The staff weekly report management designed and implemented by this subject is based on the B/S structure. The project is divided into three parts: front end, back end and database. The front and back ends are separated, and the development process is independent of each other. Finally, the data is exchanged at the front and back through the HTTP interface. The front end uses Ant's open source Ant design Pro solution based on the react and umi framework. The back end uses the Golang language to develop and cooperate with the Beego framework to complete the business logic of the web server. The database uses Mysql, and the deployment plan uses the docker container to maintain the above three parts. The functional modules can be divided into six modules: login/registration, employee management, daily work submission/summary, weekly report submission/summary, annual plan scheduling, and personal information management. And complete staff rights division, department division. Attendance, cycle summary, annual scheduling and other needs. The system interface is friendly, clear, and can be hierarchically used to achieve limited viewing range of data, and the administrator can view the summary information and make adjustments in real time, and the weekly and monthly reports can be exported and printed.

The management system implemented by this project starts with the actual demand analysis, and gradually designs and implements the main practical functions. It can complete the quantification of work, the summary of progress, the periodic output of employees, and the work results as the assessment indicators. On the one hand, it can enhance employees' consciousness, on the other hand, it improves the efficiency of employee rating, and facilitates enterprise users to

complete weekly report management reasonably and efficiently, strengthen management efficiency, and reduce management costs.

**Key words:** weekly report management system; enterprise management; B/S structure

## 目 录

摘要.....	I
ABSTRACT.....	II
1 绪论 .....	1
1.1 课题背景 .....	1
1.2 国内外现状 .....	1
1.3 研究课题的意义 .....	2
1.4 系统完成的内容 .....	2
1.5 本论文的主要内容 .....	3
1.6 本章小结 .....	3
2 任务需求分析 .....	4
2.1 整体介绍 .....	4
2.2 各部门工作自检模板 .....	4
2.2.1 营销部 .....	4
2.2.2 技术支持部 .....	5
2.2.3 售后部 .....	5
2.2.4 研发部 .....	6
2.2.5 生产部 .....	7
2.2.6 综合办公室 .....	7
2.2.7 财务部 .....	8
2.3 周报模板 .....	8
2.3.1 周总结 .....	8
2.3.2 周计划 .....	8
2.4 年度计划排单表 .....	9
3 技术选型及相关知识简介 .....	10
3.1 整体概况 .....	10
3.2 前端 .....	10
3.3 后台 .....	11
3.4 数据库 .....	11
3.5 部署方案 .....	12
3.6 本章小结 .....	13
4 总体设计 .....	14
4.1 研究课题简介 .....	14
4.1.1 系统设计目标 .....	14
4.1.2 系统模块简述 .....	14
4.2 系统总体设计说明 .....	15
4.3 系统模块概述 .....	16
4.3.1 登录模块 .....	16
4.3.2 员工管理模块 .....	17
4.3.3 各部门日报/月报汇总模块 .....	17

4.3.4 各部门日报/周报填写提交模块 .....	17
4.3.5 周报汇总模块 .....	18
4.3.6 年度计划排班表 .....	18
4.4 数据库设计 .....	18
4.4.1 数据库设计要求 .....	18
4.4.2 数据库表和字段设计 .....	19
4.5 本章小结 .....	26
5 主要功能模块的实现 .....	27
5.1 前端功能模块描述和展示 .....	27
5.1.1 登录模块 .....	27
5.1.2 主页菜单动态渲染实现细节 .....	28
5.1.3 员工管理模块 .....	31
5.1.4 每日工作汇总模块 .....	32
5.1.5 周报汇总模块 .....	33
5.1.6 年度计划排班表 .....	33
5.1.7 个人信息管理模块 .....	34
5.1.8 每日自检提交模块 .....	34
5.1.9 每周工作汇报模块 .....	35
5.2 后端主要接口代码实现 .....	35
5.2.1 HTTP 接口 .....	35
5.2.2 接口代码实现分析 .....	37
5.3 项目目录介绍 .....	39
5.3 项目测试方法介绍 .....	40
5.4 本章小结 .....	40
结论 .....	41
参考文献 .....	42
致谢 .....	43

# 1 绪论

## 1.1 课题背景

对于互联网市场来说，以往的业务可以分为 toC 和 toB 两个方向，其中 toC 指的是面向消费者（to Customers），而 toB 则指的是面向企业、组织用户（to Businesses）。国内的互联网市场从二十几年前开始兴起，一直是以 toC 为导向的发展，一路下来，出现一众顶尖的公司，比如众所周知的 BAT，通过观察其实可以发现这些大公司初期的业务几乎都是 toC 的。其中的主要原因是因为中国在 C 端的人口和流量红利吸引了资金的流入和聚集，这二十几年来互联网消费群体从几千万发展到了现在的近十亿，这一定程度驱使互联网公司不断开发各种 C 端产品吸引用户吸引流量，而 B 端业务变得鲜有人知。

但是近几年来，toB 的概念越来越火，国内开始布局 toB 业务的公司也越来越多，比如腾讯推出的企业微信、阿里推出的钉钉、还有各种云计算云服务商等。一个很明显的趋势是，国内的互联网公司开始关心 toB 市场了，甚至网上开始流传一个信息：国内的互联网开始转入下半场了，从消费互联网转向工业互联网，从人口红利转向创新红利。出现这个转变的原因有如下几点：

1)人口和流量红利骤减。与互联网兴起之初相比，现在想吸引并留住客户和消费者的代价上升的非常明显，可能以前吸引一个新用户是几块钱的成本，现在是几十块钱的成本。而且现在互联网消费人口已经近乎饱和，不会再出现大的增长，没有了之前那种新兴群体的加入，所有公司都是瓜分当前已经入局的消费群体，也就是说不再有更多的蛋糕进来，只能大家一起瓜分当前这块大蛋糕

2)用人成本的上升。得益于教育的普及和发展，现在互联网公司招聘的人群质量越来越高，随之竞争的成本也上升，优秀应届毕业生月薪已高达上万甚至热门专业能达到几万，这无疑增加了企业的用人成本，所以购买 toB 服务来代替一部分人力资源成为了新选择，也为 toB 市场造就一部分需求；

3)企业开始重视降低管理成本。互联网快速发展的同时，企业的数量和规模也越来越大，容易出现尾大不掉的情况。所以中大型的企业都开始寻找高效低廉的管理模式，提高生产力的同时降低管理的成本。

## 1.2 国内外现状

回望 20 多年来的中国信息技术商业化大潮，一个明显的特点是：C 端异军突起、弯道超车，B 端市场却远远滞后于欧美<sup>[1]</sup>。相比于国内，国外的互联网市场比我们发展要早的多，他们的 toB 市场也具备相当大的规模，也凭此诞生了很多出名的公司，比如 SAP、IBM、甲骨文等，这些公司都有非常出色的 B 端产品，而且在国外的市场上被广泛采纳和使用。国外的 toB 市场能发展这么好，原因跟上文提到的几点不无关系，因为国外普遍人口偏少，

而且互联网发展早，相应流量红利也少，而且国外的工程师数量稀缺薪水颇丰，使得企业用人成本交大，促使了越来越多企业采取购买服务的方式代替找人开发的模式。

国内现在的互联网市场也在逐步转型当中，相信会有越来越多的 B 端产品进入我们的视野。本课题研究的员工周报系统便是一次很好的尝试，能够一定程度提高企业内部的管理协作，并较少了人力资源的调用。

### 1.3 研究课题的意义

本课题研究的是设计与实现基于 B/S 结构的员工周报管理系统，主要是通过当前互联网最常见的管理模式代替以往陈旧的手工模式。对应前面提到的 toB 业务，对于具备一定规模的企业来说，可以发挥极大的作用。

传统的周报管理主要依托于手工填写和整理，员工定期填写纸质表格并上交，之后由人力部门同意存档管理。这种方式效率非常低下，而且存在诸多问题，比如纸质文档容易损坏丢失，查找也不方便，之后统计分析时又容易出错。

而随着计算机的普及和互联网的兴起，很多工作都可以实现软件信息化、自动化。比如本课题使用 B/S 结构来实现员工周报管理系统，对于用户（也就是员工）而言，只需要在浏览器上打开网页登录即可进入系统界面，而且针对不同部门不同员工不同职级的用户，系统会匹配相应的操作页面，用户可以在网页上完成周报等内容的填写和提交，而经理和管理员则可以查看和管理所有员工的提交信息。系统内部的数据操作过程对用户来说是无感知的，网页上不同组件的交互调用不同的后端接口，然后完成数据集的增删改查。开发过程前后端分离，各自完成 ui 和数据接口的任务后，最后对接联调完成系统。信息技术的引入是企业搭建信息化管理平台的基础。在信息技术不断创新发展的过程中,企业应立足自身管理需要,对信息技术进行分析与研究，并促使其逐渐融入到企业管理运行中来<sup>[2]</sup>。

一个完善便携实用的周报系统，可以实现工作的量化、进度的汇总、员工定期规划周期性输出等作用。而且根据汇总信息结合工作成果作为考核指标，一方面可以提升员工的自觉性，另一方面提升了员工评级的效率。本课题。不同部门不同职级的员工完成各自的任务并上交系统，之后的整理归档都不需要操心，极大的解放了生产力和工作效率。

而且这是对 toB 业务的一次很好的尝试，可以探索 toB 市场的前景，提升企业用户对 toB 服务的依赖和信任。

### 1.4 系统完成的内容

- 1)了解所研究内容的知识背景，并选择合适开发方案；
- 2)完成管理系统的整体设计需求，包括前端页面的设计、后端接口的设计和数据库的设计；
- 3)完成员工周报管理系统的前端交互页面；
- 4)完成员工周报管理系统的后端接口与数据库增删改查；
- 5)完成员工周报管理系统的前后端对接联调和测试工作；

## 1.5 本论文的主要内容

本论文主要是对员工周报管理系统的设计分析和实现过程进行详细的阐述，一开始需求分析，接着技术选型，然后系统模块设计，最后是代码实现和测试，其中代码实现又分为前端和后台两大块，按功能划分又可以分为登录/注册、权限、每日工作、周报（总结和计划）、月度汇总、年度安排等模块。本文主要分为六章，其结构如下：

### 第一章

绪论。绪论部分主要是对研究课题的相关背景知识调查，然后评定课题的实际研究意义，同时对比了一下国内外相关领域的发展趋势和异同，最后列出本论文的大体结构和内容。

### 第二章

任务需求分析。这部分着重分析需求，理清软件交互流程，站在用户的角度分析拆解各个页面和功能，同时思考背后的处理逻辑和涉及到的数据变化。

### 第三章

技术选型。本章主要是对现有需求和功能上进行技术选型的工作，结合自己所擅长的领域选择合适的技术方案来实现后续要完成的模块，同时对比总结不同方案的优势。

### 第四章

工程设计细节。包含系统的整体架构、前端页面的交互设计、后端 http 接口设计、数据库表设计等。

### 第五章

对系统的功能和实现进行了详细的介绍。本章主要讨论工程的详细实现过程以及部分功能的细节处理。

### 第六章

总结。本章包括总结本文的主要工作和实验结果，及对本系统开发研究时碰到的问题和本课题今后努力方向的展望。

## 1.6 本章小结

本章从课题背景、国内外发展概况谈起，探究本课题的研究意义。然后简单介绍了本课题中基于 B/S 结构的员工周报系统的设计和实现过程，包括需求分析、采用的相关技术调研和选型、工程设计概览、功能实现和总结等部分。通过本章内容可以大体了解本论文的内容组织结构和课题研究意义。



## 2 任务需求分析

### 2.1 整体介绍

本课题研究的是基于 B/S 结构的员工周报管理系统，针对的用户是中小型企业。研究的对象需要满足以下功能需求：

1)每日工作项目检查。不同部门日报的格式不同，需要根据员工的部门信息自动使用相应的填写模板；日报当天提交的可以修改，但不能修改之前已提交的旧日志；

2)每周工作内容汇总和计划。每周需要填写下一周的周计划，比如工作内容和预计完成目标等；每周也需要填写这一周的工作总结，比如工作完成情况和个人总结等；周计划和周总结的填写模板所有员工都是一样的，生成报表（可打印）时把周计划附在周总结后面。

3)可以根据日期生成月度报表，记录一个月的每日项目检查结果，不同角色的员工可查看的人员范围不同；

4)部门主管和总经理可以查看和修改年度计划排班表。这张表主要记录一年的公司项目计划和目标；

5)权限管理。企业分为总经理/总工、部门主管、普通职员，此外还有一个管理员账户，不同角色的使用权限不同，使用的交互界面也有差异，比如总经理和总工可以查看所有员工的日报和周报信息，可以查看年度计划排班表，部门经理可以查看本部门的日报和周报信息，也可以查看年度计划排班表，职员可以查看自己的日报和周报信息，管理员可以对所有信息进行管理维护；

### 2.2 各部门工作自检模板

#### 2.2.1 营销部

每日自检项目：卫生、晨会、工作

使用表格样式：

2019年营销部_____月自检表																																
	1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16	
姓名	卫生	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作	卫生	晨会	工作
王秋艳																																
肖立芳																																
张轩																																
田娜																																
张素娜																																
张淼																																

图 2-1 营销部自检表图

备注：一.1.合格“√”；2.不合格“×”；3.出差“差”；4.请假“O”

1. 卫生：工位、窗台摆放有序，干净整洁；地面清洁；
2. 晨会：是否定时召开，准时参加；
3. 工作：是否满足每天 3 个有质量沟通，客户档案是否及时归档

每日自检项目：办公室物品摆放整齐；办公环境卫生整洁；例行晨会；技术方案资料及时备案；按照既定流程制定方案；今日事今日毕；下班前断电、熄灯

使用表格样式：

每日自检项目：物品摆放整齐；环境卫生整洁；例行晨会；售后工作是否按时汇报工作情况；施工工艺符合规范（反馈照片）；工作设备、工具处于正常状态；文件、记录完整；记录工作记录。



备注：结果栏√说明合格，×说明不合格

## 2.2.5 生产部

每日自检项目：生产现场是否清洁；物料摆放是否整齐；是否按规定工艺流程生产；是否记录每日的工作内容；整改描述。

使用表格样式：

	自检记录表																															日期:	
序号	项目及质量要求	自检记录																														状况描述	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30		31
1	生产现场是否清洁																																
2	物料摆放是否整齐																																
3	是否按规定工艺流程生产																																
4	是否记录每日工作内容																																
5	整改描述																																
6	备注：“√”表示合格，“×”表示不合格，“○”表示未上班，必要时在状况栏上加以说明																																

图 2-5 生产部自检表图

备注：“√”表示合格，“×”表示不合格，“○”表示未上班，必要时在状况栏上加以说明

## 2.2.6 综合办公室

每日自检项目：办公室环境卫生情况；对各部门协助情况；文件是否及时整理归档；日计划、日总结；周计划、周总结；月计划、月总结。

使用表格样式：

自检记录表																																	
序号	项目、质量要求	每日检查记录																															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
1	办公室环境卫生情况																																
2	对各部门协助情况																																
3	文件是否及时整理归档																																
4	日计划、日总结																																
5	周计划、周总结																																
6	月计划、月总结																																
备注：“√”表示合格，“×”表示不合格，“○”表示未上班，必要时在状况栏上加以说明																																	
状况描述																																	

图 2-6 综合办公室自检表图

备注：“√”表示合格，“×”表示不合格，“○”表示未上班

## 2.2.7 财务部

每日自检项目：工作计划；临时工作；办公室卫生情况；办公用品摆放情况；明日计划。

使用表格样式：

日期：		部门：	财务部	完成情况	领导审核
工作计划					
临时工作					
办公室卫生情况	办公室卫生清洁				
办公用品摆放情况	办公物品分类摆放，整齐划一				
明日计划					

图 2-7 财务部自检表图

备注：“√”表示合格，“×”表示不合格，“○”表示未上班，必要时在状况栏上加以说明

## 2.3 周报模板

### 2.3.1 周总结

表 2-1 周总结表

姓名：

部门：

日期：

已完成任务内容		
序号	工作内容描述	结 果
1		
2		
3		
4		
5		
6		
本周目标完成综		
任务目标完成阶		

### 2.3.2 周计划

表 2-2 周计划表

姓名：

部门：

日期：

任务 内容		
本周工 作目标		
序号	工作计划描述	完成目标
1		
2		
3		
4		
5		
6		
备注		

## 2.4 年度计划排班表

表 2-3 年度计划排班表

总体目标						
年度工作计划						
序号	项目名称	紧急程 度	计划完成时间	负责人	项目进度	验收结 果
1						
2						
3						
4						

## 3 技术选型及相关知识简介

### 3.1 整体概况

技术选型分为三大块，分别是前端、后台、数据库。

其中前端页面是运行在浏览器上的，开发语言首选当然是 `JavaScript`，然后 UI 开发框架选择的是 `React`，因为当前前端开发领域 `React` 可以说是红极一时，互联网上已经有很多使用案例；前端工程框架采用的是 `Ant Design Pro`，这个是蚂蚁金服推出的开箱即用的中台前端/设计解决方案，UI 组件库使用的是与其配套的 `Ant Design`，非常适合管理后台的页面开发；上述框架中还包含 `umi` 应用框架，这个属于可插拔的企业级 `react` 应用框架，可以实现页面路由、编译打包、处理全局状态数据流、网络请求等。

后台服务端是运行在服务器上的 `http services`，开发语言选择的是 `Golang`，因为它兼具开发效率和性能，语言特性简练高效；服务端开发框架使用的是 `beego`，一款由国人开发的应用框架，有详细的使用文档和案例，足以满足开发需要。

数据库选择的是 `Mysql`，因为目前 `Mysql` 是最常用的服务端数据库，有很多使用案例，配合 `docker` 部署十分便捷。

项目工程完成后还有一个部署问题，本课题选择 `docker` 来实现容器化部署，最终前端、后台、数据库分为三个容器运行，只需要一个 `docker compose` 文件即可一键部署，三者相互之间通过暴露的端口进行通信和交互。

### 3.2 前端

当前的前端领域，提到 web 开发框架离不开 `React`、`Vue`、`Angular` 三个框架，这三个框架都有不少用户群体的拥趸，都有非常不错的开发体验和性能保证，各有异同各有特色，选择 `React` 的原因是其开源的时间较早，开发流程已经非常成熟，网络上也有很多使用案例，而且 `React` 是 Facebook 推出的开源 js 库，维护可以得到保证。

`React` 的核心理念是万物皆组件，通过声明式的编写，提供一个可以交互的 UI，而且每个组件都有各自的属性（`props`）和状态（`state`），还有一套完整的生命周期，凭此可以实现很多交互组件。而且不同组件可以相互复用嵌套，提升开发效率，节省开发时间。每个组件提供一个 `render()` 函数来接收输入的数据并返回需要展示的内容，`React` 的理念归结为一个公式，就像下面这样： $UI = render(data)$ <sup>[3]</sup>。底层则是用过 `virtualDom` 的算法来实现组件高效的更新，为前端页面的性能提供保证。

`React` 知识提供 UI 的渲染，一般不会单独地使用，工程上一般会配合一些第三方应用框架来实现更高级的功能实现和状态管理，比如使用 `react-router` 实现路由定位和跳转、`webpack` 来实现模块的打包、`axios` 来实现网络请求等。而本课题使用的是开源的 `umi` 框架，内置 `react`、`react-router`、`jest`、`webpack`、`rollup` 等工具，满足开箱即用，集成了 `dva` 框架的特性，可以方便管理全局的状态。

Ant Design Pro 则是进一步整合以上提到的框架，属于一套开箱即用的中台前端/设计解决方案，可以在其基础上快速搭建一套自用的管理后台模板，开发者只需要关心实际功能页的开的状态管理、请求接口的问题，而打包、部署、组件库、请求方法等工具不用自己再一一去添加整合。而且其内部根据常用场景实现了一些可用的组件来帮助实现一些复杂页面和功能，还集成了 UI 测试和数据 Mock，可以说是最佳实践的一套解决方案。

### 3.3 后台

后端开发语言目前有众多选择，比如 Java、PHP、NodeJS、Golang、Python、C 等等，每种语言都有其特点。Java 是目前服务端领域采用最多的语言，但是开发效率并不算出色，而且编译部署略微麻烦；PHP 和 NodeJS 以及 Python 属于动态语言，开发效率确实很高，但是单线程性能表现没那么出色，而且后期维护也不轻松；C 和 Golang 属于强静态语言，在类型验证上比较安全，但是 C 很少作为服务端开发的选择，因为相关的工具较少而且开发效率低下；Golang 是谷歌团队 2009 年推出的新时代编程语言，Go 语言专门针对多处理器系统应用程序的编程进行了优化，使用 Go 编译的程序可以媲美 C 或 C++代码的速度，而且更加安全、支持并行进程，而且语言特性简练高效，兼具了开发效率和性能，目前已经有越来越多的公司开始采用 Golang 作为开发首选。

Golang 编译速度极快，而且编译后可以直接生成二进制可执行文件，除了少数代码需要 libc，运行几乎没有任何环境依赖，这对于部署特别是容器化部署来说是非常棒的特性。此外 Golang 还有极致的并发编程体验，因为其底层实现专门为现代多处理器设计，可以快速安全地写出并程序，代码中只需要在函数前声明一个 go 关键字即可将其放入协程中并行处理，这当中的协程（goroutine）是语言本身实现的运行时（runtime）创建的用户层线程，相比于操作系统中提到的进程和线程，协程占用的资源非常少，一个 goroutine 初始只需 2kb 内存，而一个线程则需要几 MB，因为协程的轻量一台机器可以运行百万级别的协程。而且运行时有自己的一套调度机制，没有操作系统切换线程带来的上下文变更引起的消耗，协程因网络请求而阻塞也不会引起线程阻塞，大大提升了程序的运行效率。此外 Golang 还有非常高效的垃圾回收机制，其 GC 的性能当属后端语言的前列。Golang 还是开源的语言，由一众顶尖的程序员操刀设计并实现，其理念非常适合当代互联网，Golang 也被比作 21 世纪的 C 语言，因此 Golang 的使用前景可以得到保证。

而本课题使用的服务端框架 Beego 就是一款国人基于 Golang 开发 MVC 架构服务框架。Beego 内置了很多服务端开发需要用到的工具，比如路由模块、缓存模块、日志模块、数据库 orm 模块等，凭借这些内置的工具可以快速开发一个 RESTful 服务，框架支持 MVC 的方式，用户只需要关注逻辑，实现对应 method 的方法即可。

### 3.4 数据库

目前数据库领域主要可以划分为两种，一种是传统的关系型数据库管理系统(RDBMS)，另一种是近年来兴起的 NoSQL 数据库（泛指非关系型的数据库）。后者数据模型不再遵



从以往的强关系，一般也不使用基于 SQL 语句的查询，其内部实现大部分是基于 K/V 的数据模型实现（也有基于列存储、文档型存储、图形结构的数据库），其特点是查询的效率很高，对于很多高并发的极大数据使用场景来说非常合适，而且经过不断发展，NoSQL 的表现和性能越来越受到关注，很多领域比如 CDN、缓存、大数据、云计算出现的很多，同时也逐渐渗透进传统数据库的市场，比较火的非关系型数据库的有 redis、mongoDB、Cassandra、Hbase 等。

非关系型数据库的使用场景一般数据模型比较简单，变动少，但对性能要求极高，不需要高度的强一致性。并不适用于本课题的工程研究，对于本课题的数据库选型更偏向于关系型数据库。

关系型数据库市场也有众多选择，比如大型软件常用 Oracle、Sql Server 等大型数据库，而小型需求的也有 sqlite 等，而 Mysql 介于两者之间，是目前服务端最常用的数据库选择，Linux 服务器常用的网站结构“LAMP”中 M 就代表 Mysql。

Mysql 虽然相比 Oracle 这些大型传统数据库存在一些差距，但是其本身的性能和易用性足以受到大部分企业的青睐。Mysql 跨平台兼容性很好，几乎支持主流的所有平台。对 SQL 语句的执行也做了一定优化，有很强的性能表现。Mysql 属于开源产品，没有费用，而且企业和用户可以在其基础上添加一些中间件实现一些定制化功能。Mysql 不仅可以单机使用，还可以以集群的方式部署，给中大型企业提供服务，其内部提供了分布式环境下主从机制、数据同步、数据备份等功能。Mysql 的特性很多，而且兼容性和易用性极佳，使用案例也众多，作为本课题的数据库选型非常合适。

### 3.5 部署方案

最终完成整个工程的开发和测试后，如何部署也是一个很重要的问题。前端页面借助 umi 自带的编译打包工具可以一键生成静态文件目录（dist），配合 nginx 镜像可以实现网页访问。后端 beego 框架因为是 golang 编写的，可以编译成二进制可执行文件，只要有进程守护程序即可，这里选择用容器托管。数据库有现成 docker 镜像可供使用。最后配合 docker-compose 可以将上述三者配置成三个独立的容器运行，相互之间暴露端口以供服务。

容器化可以实现高效的隔离，保证进程、数据、状态的密封性，每个容器只向外暴露指定的端口，内部逻辑并不暴露，其中一个出错也不会影响其他，提高了安全性、可用性和稳定性。而且容器配置了重启机制，出错了也会自动尝试重启，提供进程守护的功能。配合 docker-compose 可以实现一键部署，简化了整个项目的部署复杂度，解放双手，提升生产力。容器化也可以解决环境依赖问题，如果直接在裸机上部署项目，相关的环境搭建可能需要很多步骤，而 docker 镜像已经可以提供我们需要的环境，只需要拉取相应的镜像然后加入工程编译好的代码包，即可运行，解决了环境搭建的麻烦。

在机器上安装了 `docker` 的任何人都是可以运行该服务，不必担心依赖项的安装，不必考虑编译器或任何其他需要支持的基础设施，开发机器也不会因为安装了这个服务的配置和依赖项而受影响。所有内容都包含在 `Docker` 镜像中<sup>[15]</sup>。

### 3.6 本章小结

本章在上一张需求分析的基础上做了相关功能实现的技术选型，在不同领域对相关产品或者技术方案做了一些对比，选出适合本课题研究和使用的技术方案，并对选择的技术做简单的优势分析和介绍，之后功能实现模块可能还会反复提到并使用上述提到的框架和技术。

## 4 总体设计

### 4.1 研究课题简介

#### 4.1.1 系统设计目标

员工周报管理系统，给企业用户提供一个便捷高效的工作成效记录的作用，不局限于周报的记录，还可以提供每日的工作检查记录然后按月返回报表、周总结和周计划、年度计划排班表，管理员用户还有员工信息和权限管理，满足不同部门不同职级的员工能完成本职内的工作，同时系统还能够划分不同角色权限的查看范围，避免数据的泄漏。

以下为本次系统设计的主要目标：

- 1) 系统可以稳定运行，安全可靠；
- 2) 人机交互界面友好；
- 3) 系统可以实现动态管理；
- 4) 系统中各类信息的查询都灵活、方便、快捷、准确；
- 5) 实现所有模块功能，工作人员操作方便；
- 6) 数据库存储安全；
- 7) 数据保密性非常好，为每个员工设置相应的权限；

#### 4.1.2 系统模块简述

员工周报管理系统按工程可分为前端、后台、数据库三个模块，按功能可分为登录模块、注册/创建用户模块、员工管理模块、权限管理模块、各部门日报/月报汇总模块（每个部门对应不同模板）、周报汇总模块、年度排版计划模块、员工日报/周报填写提交模块、用户信息修改模块、用户密码登录模块。

不同部门的员工每日自检项目不同，所以日报/月报的交互页面也不同；不同角色的员工对应不同权限，不仅交互界面不同，看查看的数据范围也不同。

系统角色权限关系如图 4-1 所示：

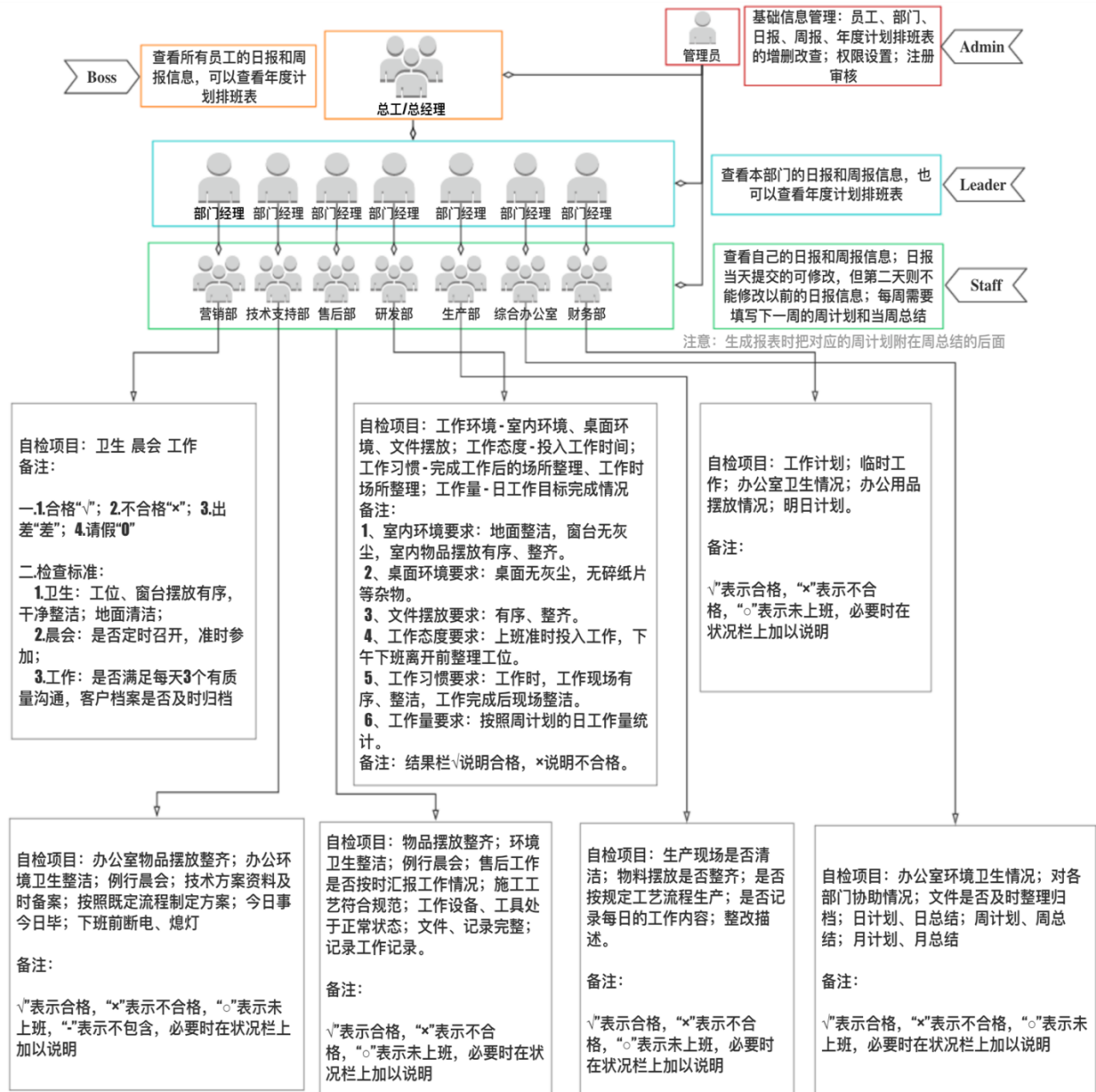


图 4-1 角色权限关系图

## 4.2 系统总体设计说明

员工周报管理系统整体主要分为登录模块、员工管理 模块、各部门日报/月报汇总模块、周报汇总模块、年度排版计划模块、员工日报/周报填写提交模块、用户信息修改模块。角色分为游客、普通职员、部门主管、总工、管理员五种权限。

员工周报管理系统的系统体系结构图如图 4-2 所示：

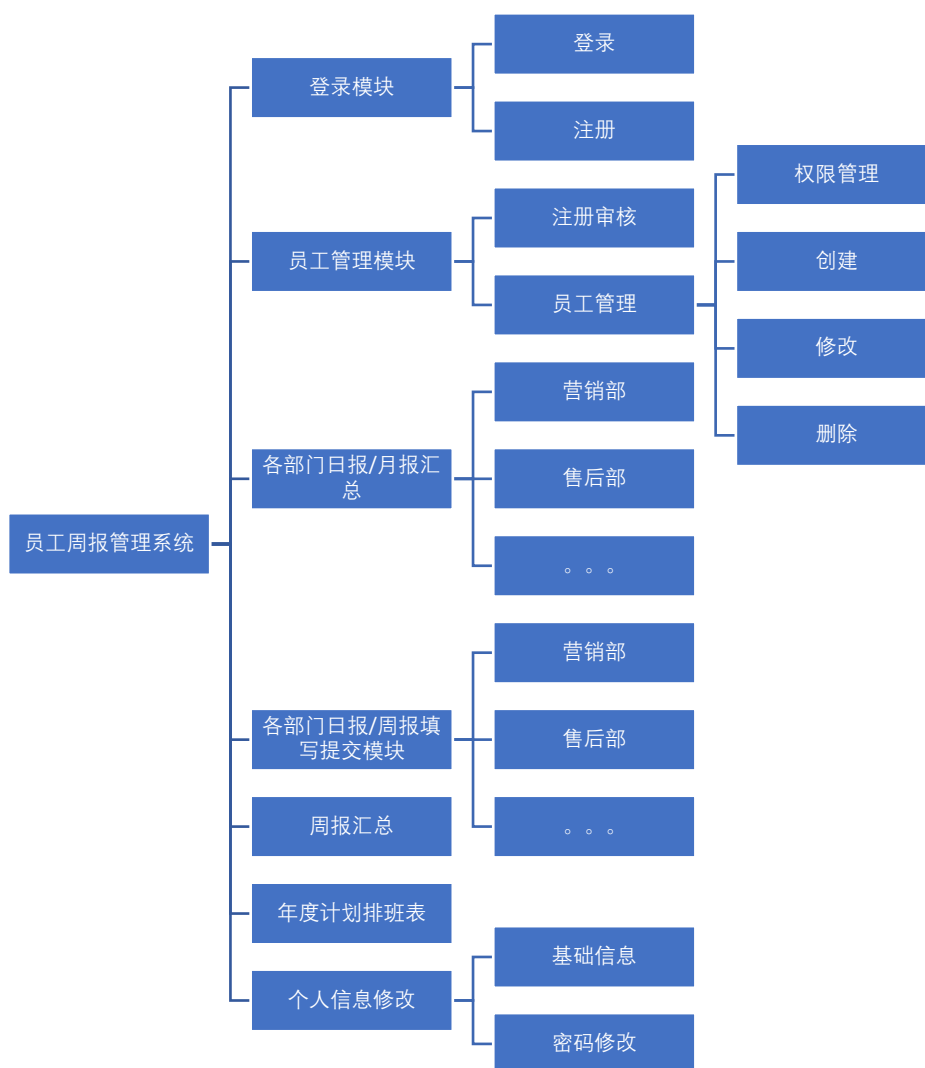


图 4-2 系统体系结构图

## 4.3 系统模块概述

### 4.3.1 登录模块

登录模块分为登录页和注册页。

- 1) 登录页使用账户名和密码登录，密码验证通过后才可进入主页面；
- 2) 注册页则用于注册新用户，填写账户名、密码、姓名、入职日期、手机、邮件、部门等相关信息并提交后会返回添加成功与否的响应，创建成功后可以使用新账号登录。新账号属于游客身份，虽然也能登录进入主页面，但是功能菜单只有个人信息修改模块，只有等管理员在注册审核模块中通过验证并设置角色权限，新账户才能正常展示功能页。

登录流程图如图 4-3 所示：

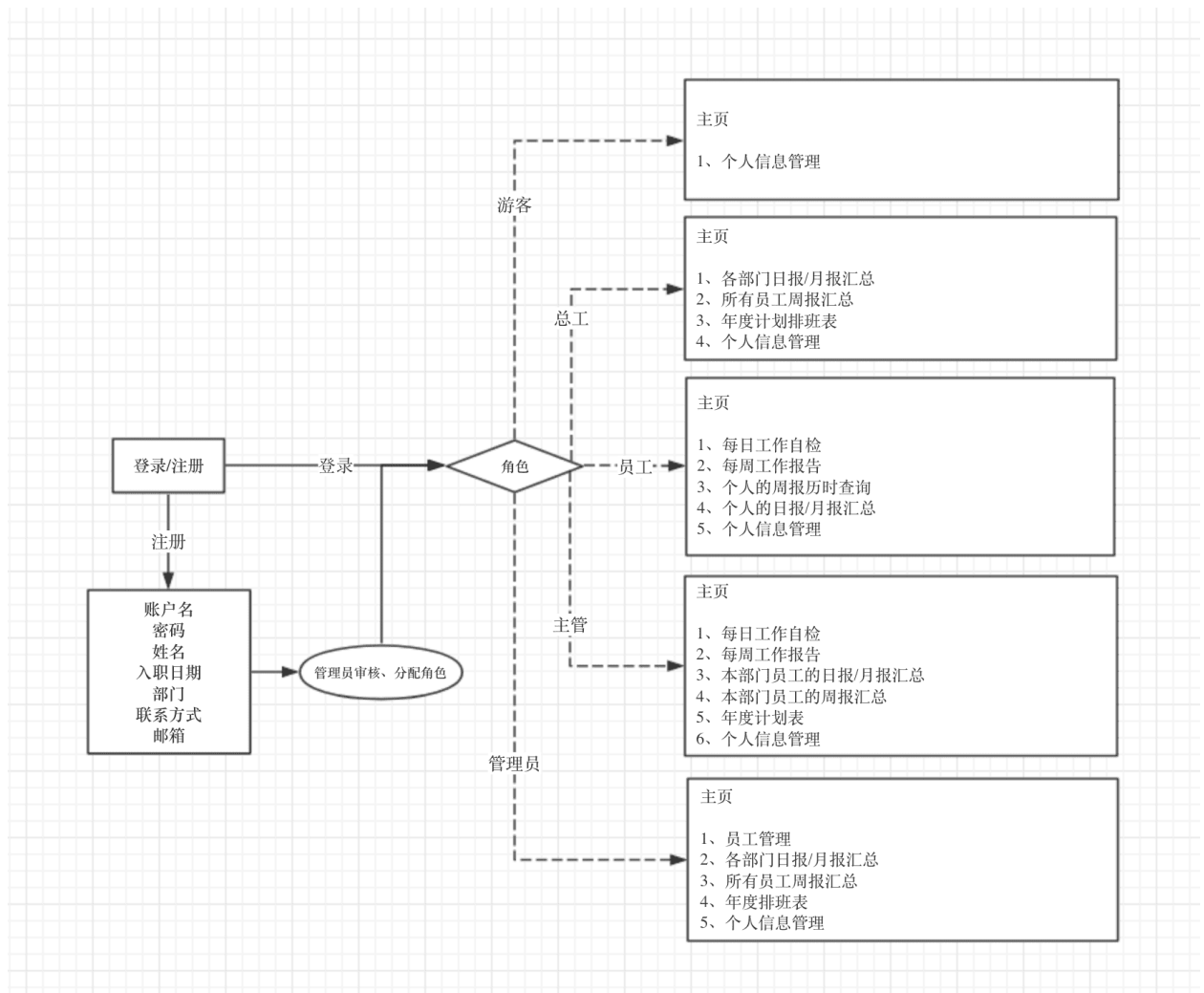


图 4-3 登录流程图

### 4.3.2 员工管理模块

员工管理模块只有管理员账户才能使用，且划分为注册审核和员工管理两个功能页。

- 1) 注册审核页面对新注册的账户进行权限设置、删除等操作；
- 2) 员工管理页面会列出所有在职员工，管理员可以创建员工、修改员工信息、删除员工（状态转为离职，信息仍保存在数据库中）、修改员工权限。

### 4.3.3 各部门日报/月报汇总模块

各部门日报/月报汇总模块，管理员、总工可以查看所有员工的记录，部门主管只能查看同一部门的汇总信息，普通职员只能查看自己的历史记录。

因为各部门的每日自检项目不同，此模块需要根据部门所属展示不同汇总表格，且功能页需要提供日期范围选择，以便动态查询不同时间段内的数据，还要提供导出按钮方便导出表格数据的图片以供打印。

### 4.3.4 各部门日报/周报填写提交模块

各部门日报/周报填写提交模块，只有普通职员和主管可以使用。

- 1) 日报的填写每个部门都有各自的填写项目，需要前端交互页面判断当前用户的部门所属信息然后返回不同的功能页；
- 2) 周报的填写格式统一，其中包含周总结和周计划两部份。

#### 4.3.5 周报汇总模块

周报汇总模块，管理员、总工可以查看所有员工的记录，主管可以查看本部门的所有记录，普通员工只可以查看自己的。

周报汇总页面也提供日期范围选择的工具以便查询指定周的数据，同时也提供导出打印功能。

#### 4.3.6 年度计划排班表

年度计划排班表模块，只有管理员和总工、主管可见。年度计划排班表记录每年的整体目标和计划，同时还记录一些具体的项目工程和完成进度信息，提供增删改查的功能。

### 4.4 数据库设计

#### 4.4.1 数据库设计要求

对于管理系统一类的应用来说，前端界面的交互最终操作的都是后台的数据，而数据如何高效存储则会影响整个系统的运行情况。如何把需求分析中涉及的信息抽象出合适的数据库模型也是本课题设计的重点。抽象出的数据库模型不仅要能代表信息状态，同时要满足关系约束和字段类型约束，避免出现数据错乱的或者查找不方便的情形，另外常做筛选条件的字段要适当的添加索引，以便提升查找的效率。

整体的数据库模型关系如图 4-4 所示：

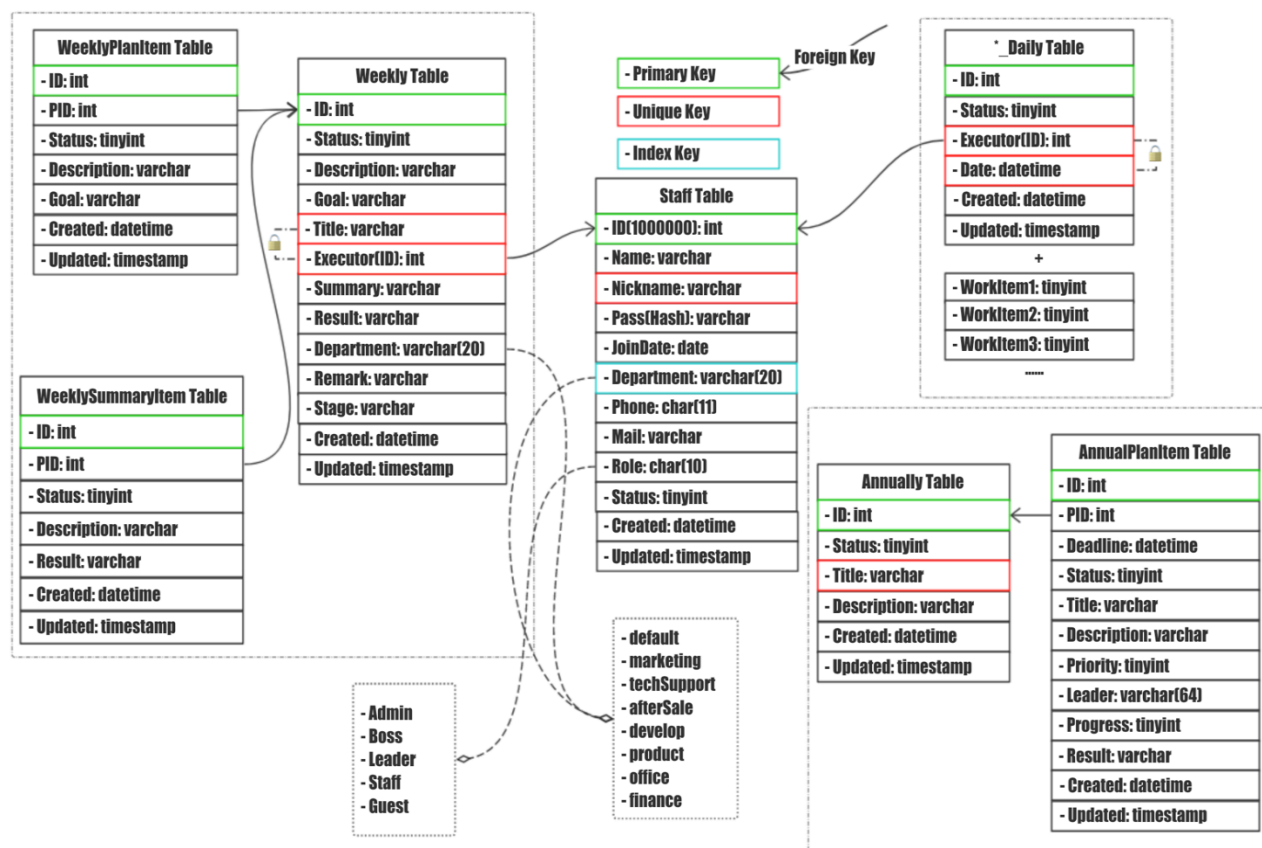


图 4-4 数据模型关系图

#### 4.4.2 数据库表和字段设计

表 4-1 员工信息表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
nickname	varchar(64)	no	Unique	null	
name	varchar(64)	no		null	
password	varchar(128)	no		null	
join_date	date			null	
department	varchar(20)	no	Index	default	
mail	varchar(64)	no		null	
mobile	varchar(12)	no		null	
role	char(10)	no		null	
status	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

员工信息表 4-1 中 id 为自增主键，初始值为 10000，项目中 mysql 初始化时会导入建表语句，在员工信息表中插入管理员账户，默认 id 10000 就是管理员账户（初始账户和密码是 admin/123456789）；nickname 是账户名，为唯一索引，不允许重复；department 代表部门（值为 marketing、techSupport、afterSale、develop、product、office、finance 之一），



因为此键常做查询条件，所以添加索引；name 代表实际名字，可以重复；password 记录密码，为了避免明文泄漏，此处存的是哈希后的字符串；join\_date 代表入职日期；mail 代表邮箱地址；mobile 代表手机；role 代表用户角色，值为 guest、staff、leader、boss、admin 之一，分别对应游客、普通员工、部门主管、总工、管理员，用以区分权限；status 代表状态，用于标记数据，方便后续查询筛选，值为 -1 时代表员工离职，0 代表初始游客状态，1 代表普通员工和部门主管状态，3 代表总工状态，5 代表管理员状态；created 代表数据添加的时间；updated 代表数据变更的时间戳。

表 4-2 营销部自检表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
executor	uint(16)	no	Foreign	null	
date	date	no		null	
status	tinyint(4)	yes		0	
hygiene	tinyint(4)	yes		0	
meeting	tinyint(4)	yes		0	
work	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

营销部自检表 4-2 中 id 为自增主键；executor 为指向员工 id 的外键；date 代表此记录的日期，格式为‘2019-05-29’，（executor，date）键值对为唯一索引，因为同一员工一天只会记录一条自检数据，不允许重复；status 保留字段，可用于标记删除等；hygiene 代表卫生检查情况；meeting 代表晨会参加情况；work 代表每日工作情况；created 和 updated 同上。

表 4-3 技术支持部自检表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
executor	uint(16)	no	Foreign	null	
date	date	no		null	
status	tinyint(4)	yes		0	
work_item1	tinyint(4)	yes		0	
work_item2	tinyint(4)	yes		0	
work_item3	tinyint(4)	yes		0	
work_item4	tinyint(4)	yes		0	
work_item5	tinyint(4)	yes		0	
work_item6	tinyint(4)	yes		0	
work_item7	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

技术支持部自检表 4-3 中 id 为自增主键；executor、date、status、created、updated 同营销部自检表 4-2；work\_item1 代表办公室物品摆放情况；work\_item2 代表办公室卫生情况；work\_item3 代表例行晨会情况；work\_item4 代表技术方案备案情况；work\_item5 代表流程制定方案情况；work\_item6 代表今日事今日毕完成情况；work\_item7 代表下班断电、熄灯情况。

表 4-4 售后部自检表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
executor	uint(16)	no	Foreign	null	
date	date	no		null	
status	tinyint(4)	yes		0	
work_item1	tinyint(4)	yes		0	
work_item2	tinyint(4)	yes		0	
work_item3	tinyint(4)	yes		0	
work_item4	tinyint(4)	yes		0	
work_item5	tinyint(4)	yes		0	
work_item6	tinyint(4)	yes		0	
work_item7	tinyint(4)	yes		0	
work_item8	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

售后部自检表 4-4 中 id 为自增主键；executor、date、status、created、updated 同营销部自检表 4-2；work\_item1 代表物品摆放情况；work\_item2 代表环境卫生情况；work\_item3

代表例行晨会情况；work\_item4 代表售后工作汇报情况；work\_item5 代表施工工艺情况；work\_item6 代表工作设备正常情况；work\_item7 代表文件记录情况；work\_item8 代表记录工作内容情况。

表 4-5 综合办公室自检表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
executor	uint(16)	no	Foreign	null	
date	date	no		null	
status	tinyint(4)	yes		0	
work_item1	tinyint(4)	yes		0	
work_item2	tinyint(4)	yes		0	
work_item3	tinyint(4)	yes		0	
work_item4	tinyint(4)	yes		0	
work_item5	tinyint(4)	yes		0	
work_item6	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

综合办公室自检表 4-5 中 id 为自增主键；executor、date、status、created、updated 同营销部自检表 4-2；work\_item1 代表环境卫生情况；work\_item2 代表对各部门协助情况；work\_item3 代表文件整理归档情况；work\_item4 代表日计划、日总结情况；work\_item5 代表周计划、周总结情况；work\_item6 代表月计划、月总结情况。

表 4-6 研发部自检表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
executor	uint(16)	no	Foreign	null	
date	date	no		null	
status	tinyint(4)	yes		0	
work_item1	tinyint(4)	yes		0	
work_item2	tinyint(4)	yes		0	
work_item3	tinyint(4)	yes		0	
work_item4	tinyint(4)	yes		0	
work_item5	tinyint(4)	yes		0	
work_item6	tinyint(4)	yes		0	
work_item7	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

研发部自检表 4-6 中 id 为自增主键；executor、date、status、created、updated 同营销部自检表 4-2；work\_item1 代表室内环境情况；work\_item2 代表桌面环境清洁；work\_item3

代表文件摆放情况；work\_item4 代表投入工作时间情况；work\_item5 代表完工后整理场所情况；work\_item6 代表工作室场所整理情况；work\_item7 代表日工作目标完成情况。

表 4-7 生产部自检表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
executor	uint(16)	no	Foreign	null	
date	date	no		null	
status	tinyint(4)	yes		0	
work_item1	tinyint(4)	yes		0	
work_item2	tinyint(4)	yes		0	
work_item3	tinyint(4)	yes		0	
work_item4	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

生产部自检表 4-7 中 id 为自增主键；executor、date、status、created、updated 同营销部自检表 4-2；work\_item1 代表生产现场清洁情况；work\_item2 代表物料摆放情况；work\_item3 代表工艺流程生产规范情况；work\_item4 代表每日工作内容记录情况。

表 4-8 财务部自检表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
executor	uint(16)	no	Foreign	null	
date	date	no		null	
status	tinyint(4)	yes		0	
work_item1	tinyint(4)	yes		0	
work_item2	tinyint(4)	yes		0	
work_item3	tinyint(4)	yes		0	
work_item4	tinyint(4)	yes		0	
work_item5	tinyint(4)	yes		0	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

财务部自检表 4-8 中 id 为自增主键；executor、date、status、created、updated 同营销部自检表 4-2；work\_item1 代表工作计划情况；work\_item2 代表临时工作情况；work\_item3 代表办公室清洁情况；work\_item4 代表办公室物品分类情况；work\_item5 代表明日计划情况。

表 4-9 周报记录表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
title	varchar(12)	no		null	
executor	uint(16)	no	Foreign	null	
status	tinyint(4)	yes		0	
description	varchar(512)	yes		null	
goal	varchar(512)	yes		null	
remark	varchar(512)	yes		null	
department	varchar(20)	no		default	
complete	varchar(512)	yes		null	
summary	varchar(512)	yes		null	
stage	varchar(512)	yes		null	
result	varchar(512)	yes		null	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

周报记录表 4-9 中，id 为自增主键；executor 为指向员工 id 的外键；title 代表此记录的周信息，格式为‘2019-21 周’，（executor，title）键值对为唯一索引，因为同一员工一周只会记录一条周报数据，不允许重复；status 为保留字段，用作标记，比如标记删除等；description 代表周计划中计划内容描述；goal 代表周计划中周工作目标；remark 代表周计划中备注；department 代表部门标记，方便部门筛选；complete 代表周总结已完成的任务内容；summary 代表周总结中目标完成综述；stage 代表周总结中目标完成阶段；result 代表部门审核结果，保留作后续使用；created 和 updated 同上。

表 4-10 周总结工作项记录表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
pid	uint(16)	no	Foreign	null	
status	tinyint(4)	yes		0	
description	varchar(512)	yes		null	
result	varchar(512)	yes		null	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

周总结工作项记录表 4-10 中，id 为自增主键；pid 为指向周报记录表的 id，每份周报中周总结有多项具体工作内容的细分总结；description 代表工作项内容描述；result 代表工作项结果；status、created 和 updated 同上。

表 4-11 周计划工作项记录表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
pid	uint(16)	no	Foreign	null	
status	tinyint(4)	yes		0	
description	varchar(512)	yes		null	
goal	varchar(512)	yes		null	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

周计划工作项记录表 4-11 中，id 为自增主键；pid 为指向周报记录表的 id，每份周报中周计划有多项具体工作内容的细分；description 代表工作项计划描述；goal 代表该工作项预期目标；status、created 和 updated 同上。

表 4-12 年度计划排班表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
status	tinyint(4)	yes		0	
title	varchar(8)	no	Unique	null	
goal	varchar(512)	yes		null	
plan	varchar(512)	yes		null	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

年度排班表 4-12 中，id 为自增主键；title 代表年份，格式为‘2019’，建立了唯一索引，不允许重复；goal 代表年度总体目标；plan 代表年度工作计划；status、created 和 updated 同上。

表 4-13 年度计划工作项记录表

列名	数据类型	是否允许空	键类型	默认值	其他
id	uint(16)	no	Primary	null	auto_increment
title	varchar(64)	no		null	
pid	uint(16)	no	Foreign	null	
status	tinyint(4)	yes		0	
description	varchar(512)	yes		null	
priority	tinyint(4)	yes		0	
deadline	datetime	yes		null	
leader	varchar(64)	no		default	
progress	varchar(128)	yes		null	
result	varchar(128)	yes		null	
created	datetime	no		CURRENT_TIMESTAMP	
updated	timestamp	no		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

年度计划工作项记录表 4-13 中，id 为自增主键；title 代表项目名称；pid 为指向年度计划排班表 id 的外键，年度计划排班表中有多项具体项目的细分；description 代表项目描述，保留字段；priority 代表项目优先级，用以区分紧急程度；deadline 代表项目预估截止时间；leader 代表项目负责人；progress 代表项目完成进度；result 代表项目验收结果；status、created 和 updated 同上。

## 4.5 本章小结

本章对整个工程项目的设计做了详尽的描述，主要讲解本课题如何处理第二章整理分析的需求。在需求任务和技术选型的基础上，将系统拆分成多个细的模块，然后梳理清楚每个模块的任务，方便后续实现和整体设计的思路对接，因此每个模块的实现涉及到的交互 UI、数据接口、数据库表可以清晰地拆分出来。本章还详细介绍了数据库表和字段的设计，方便后续实现对整个系统的理解。

## 5 主要功能模块的实现

### 5.1 前端功能模块描述和展示

#### 5.1.1 登录模块

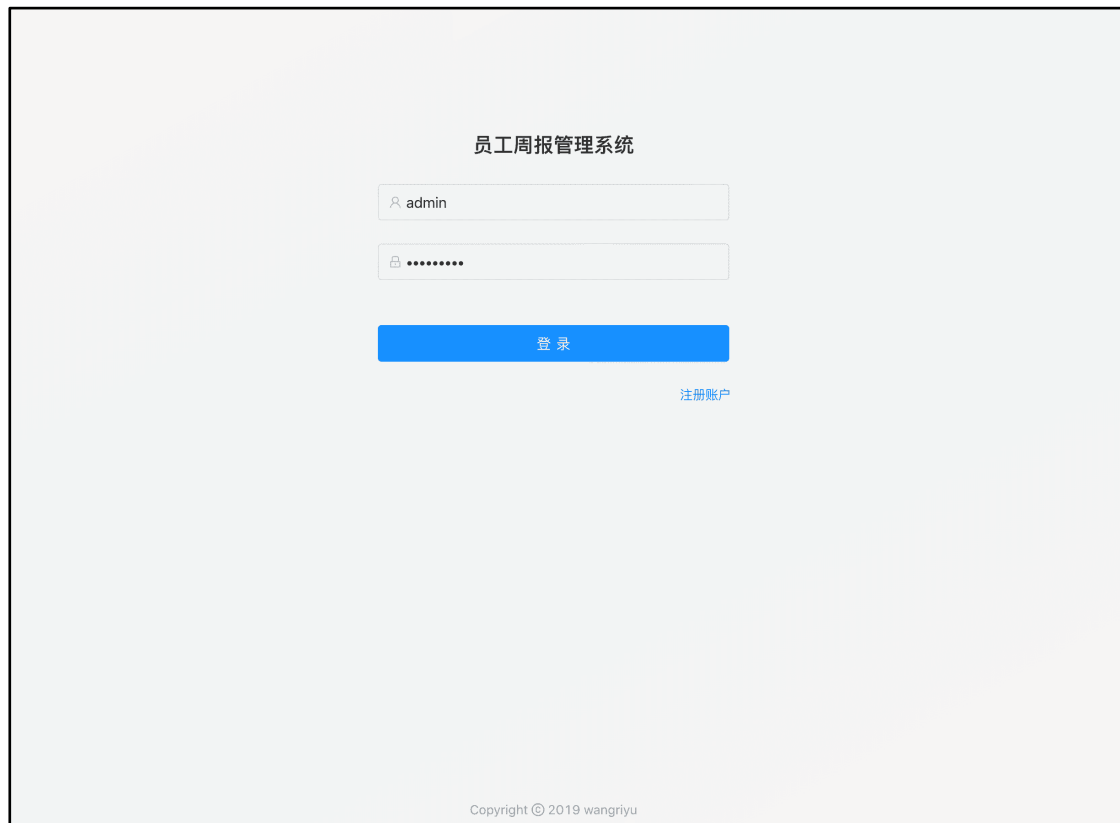


图 5-1 登录页展示图

员工打开网页初始状态会重定向到登录功能页，使用账号和密码登录成功即可进入主页，不同部门不同职级的员工使用的主页不同。如果密码错误交互界面会做出提示。

一开始只有一个 **admin** 账户可用，可以通过右下角注册进入注册功能页。注册页需要填入登录用的账户名（不可与他人重复）、密码（强度提示），确认密码（验证与前者相同）、真实姓名、入职日期（点击打开选择面板）、手机号、邮箱、部门（下拉框选择，如果是总工这种不属于某个部门的选择 **default** 选项），以上参数需要填入指定格式的数据且不能为空，否则表单会提示错误。填写完点击注册如果成功会进入显示注册成功的页面否则返回错误信息，然后返回注册页可以使用新注册的账户进入主页，因为是新账户，权限目前是游客状态，功能菜单只有个人信息页，等待管理员审核分配相应权限后才能正常显示完整的功能页列表。新账户也可以使用管理员账号进入员工管理功能页进行创建。



员工周报管理系统

注册

游客 101

.....

.....

两次输入的密码不匹配!

姓名

2019-06-20

请输入真实姓名

手机号

请输入 11 位有效手机号!

邮箱

请输入邮箱地址!

研发部

注册

使用已有账户登录

强度: 中

请至少输入 6 个字符。请不要使用容易被猜到的密码。

Copyright © 2019 wangriyu

图 5-2 注册页展示图

### 5.1.2 主页菜单动态渲染实现细节

不同部门不同职级的用户登陆后的功能页面是不同的，这一块需要前端实现动态选择性渲染，所有用户都有欢迎页和个人信息页，下面不再重复描述。

- 1) 假如当前登入用户是游客，侧边栏只有欢迎页和个人信息页，可操作的数据只有个人信息修改或者密码修改；
- 2) 假如当前登入的用户是在职普通员工，侧边栏包含每日工作自检、每周工作报告、周报历史清单、月度工作汇总的功能页。每日工作自检根据用户的部门信息提供相应工作项的填写；每周工作报告的功能页使用的是通用模板，包括当周总结和下周计划；周报历史清单可以查看自己的历史周报但不能修改；月度工作汇总可以查看指定范围内的每日自检结果。
- 3) 假如当前登入的用户是部门主管，侧边栏包含每日工作自检、每周工作报告、部门周报汇总、月度工作汇总、年度计划排班表的功能页。每日工作自检、每周工作报告同上；部门周报汇总可以查看本部门的员工的历史周报记录；月度工作汇总同理

可以查看本部门的员工每日自检结果；年度计划排班表是全年计划清单，主管及以上权限的人可看。

- 4) 假如当前登入的用户是总工/总经理，侧边栏包含每日工作汇总、周报汇总、年度排班表的功能页。总工不需要写每日自检和周报，但是可以查看所有部门的日报、周报记录，还有年度计划安排。
- 5) 假如当前登入的用户是管理员，侧边栏包含员工管理、每日工作汇总、周报汇总、年度计划排班表。管理员相当于在总工的基础上多了员工管理，还有月报、周报导出的功能。

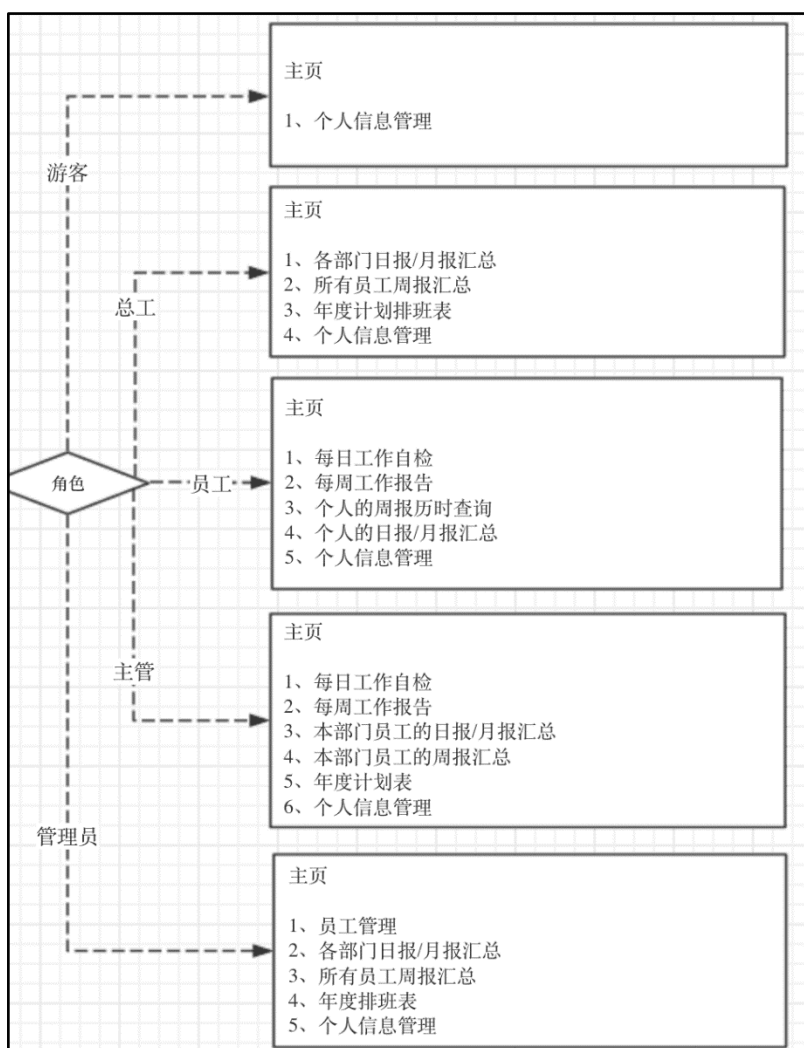


图 5-3 动态菜单展示图

前端实现以上功能的逻辑是：

- 1) 在页面路由文件中给相应功能页的组件添加一个 **authority** 标识，标识准入权限；
- 2) 在用户登陆后将用户基础信息（比如 **id** 和部门信息）保存在本地，其中用户角色会作为之后的权限；

- 3) 然后实现一层权限组件，通过比对现有权限与准入权限，决定相关元素的展示。核心代码如下；
- 4) 主功能页通过路由配置作为 Authorized 组件的子组件，只有权限标识符合的子组件才会被渲染，其中 getAuthority 函数就是从本地保存的用户数据中获取用户角色权限信息，AuthComponent 的作用就是比对用户权限和准入权限。

```
import Authorized from '@/utils/Authorized';
import { getAuthority } from '@/utils/authority';
function AuthComponent({ children, location, routerData }) {
  const auth = getAuthority();
  const isLogin = auth && auth[0] !== 'guest';
  const getRouteAuthority = (path, routeData) => {
    let authorities;
    routeData.forEach(route => {
      // match prefix
      if (pathToRegexp(`${route.path}(.*)`).test(path)) {
        authorities = route.authority || authorities;
        // get children authority recursively
        if (route.routes) {
          authorities = getRouteAuthority(path, route.routes) || authorities;
        }
      }
    });
    return authorities;
  };
  return (
    <Authorized
      authority={getRouteAuthority(location.pathname, routerData)}
      noMatch={isLogin ? <Exception403 /> : <Redirect to="/user/login" />}
    >
      {children}
    </Authorized>
  );
}
```

5.1.3 员工管理模块

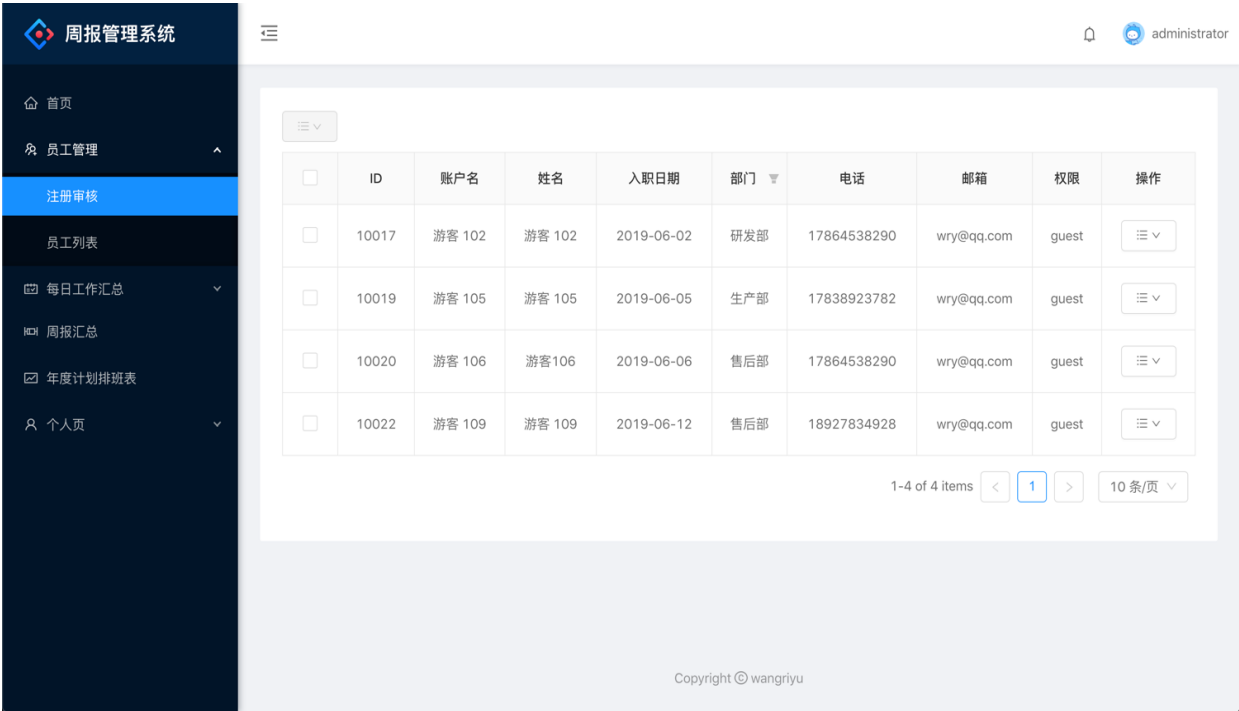


图 5-4 注册审核图

员工管理模块分为注册审核和员工列表两个功能页。注册审核页面列出所有新注册的游客账户，可以点击单条数据的最后一栏选择“设为员工”、“设为主管”、“设为总工”或者删除记录四种操作，其中删除是硬删除，直接删除数据库记录；也可以勾选多条数据，点击左上方按钮进行多项操作。



图 5-5 员工列表图

员工列表会列出所有在职员工；点击右上角新建可以添加新员工；点击单条数据最后一栏可以选择修改员工信息，此处可以修改员工的部门和角色信息但不能修改密码，个人信息管理功能页则可以修改自己的密码和基础信息但不能修改自己所属的部门和角色；删除记录相当于将员工状态设为离职，数据库中会把 `status` 字段设为 `-1`，但不会硬删除。

### 5.1.4 每日工作汇总模块

此模块可以查看所有部门的每日自检记录，功能页顶部提供日期选择工具可以动态选择时间范围，还提供了一个可以快速选择月份的方便生成月报，右上角导出按钮可以将表格导出图片，方便打印。其中营销部的报表跟其他部门略有不同，因为营销部自检项目只有三个而且字段短，所有员工的记录可以都放在一张表格里，而其他部门因为自检项目相对复杂一些，只能按个人记录返回。

图 5-6 营销部月报展示图

图 5-7 个人月报展示图

### 5.1.5 周报汇总模块

周报汇总功能页会列出员工列表，然后点击操作栏中查看周报按钮可以打开周报展示面板。对于不同用户，返回的员工数据集不太一样，主管可以看到自己部门的员工，而总工和管理员可以看到所有普通职员和部门主管。周报面板也提供了快速日期选择工具，左边可以选择周总结对应的周，右边可选择周计划对应的周，右上角也提供了导出按钮。

营销部刘逸东 2019- 23 周总结		
已完成任务内容 按实际达拉斯的		
目标完成综述 去外地了昆明气温的		
目标完成阶段 起来看我们的期望的		
序号	工作内容描述	结果
1	请问考虑到目前我的	辣么撒大声地
2	完全离开的暖气温度	拉伸膜；奥斯陆大厦上次

营销部刘逸东 2019- 24 周计划		
计划任务内容 阿萨德叫你起来味道		
周工作目标 奥斯卡流程那上次		
备注 ；爱死了马上吃三次		
序号	工作计划描述	完成目标
1	爱死；里充满期望；承	爱死；开幕词；请我吃

图 5-8 个人周报展示图

### 5.1.6 年度计划排班表

主管、总工、管理员均可查看和编辑此界面。可以新建年度计划，也可以添加具体工作项内容，也可以进行修改删除等操作。

🏠 首页

👤 员工管理

📅 每日工作汇报

📊 周报汇总

📅 年度计划排班表

👤 个人页

新建

	ID	标题	总体目标		年度计划			操作
-	4	2020	看我而温柔		我看见恩认为了人口为老人			<a href="#">编辑</a>   <a href="#">添加任务</a>
	ID	项目名称	紧急程度	计划完成时间	负责人	项目进度	验收结果	操作
	8	奥会计师	● 高	2019-06-21 15:18:07				<a href="#">编辑</a>   <a href="#">删除</a>
	6	爱神的箭你欠我的	● 低	2019-06-30 22:54:57	小泽			<a href="#">编辑</a>   <a href="#">删除</a>
+	1	2019	拉手可能长期来看我才能请问大神；撒即可从前完成了洒出		1、圣诞树老师看； 2、萨科技你是谁大V			<a href="#">编辑</a>   <a href="#">添加任务</a>
+	2	2018	全文看全文		看见你的身份水电费			<a href="#">编辑</a>   <a href="#">添加任务</a>

图 5-9 年度排班表展示图

### 5.1.7 个人信息管理模块

所有用户均有此功能页，可以点击侧边栏的个人设置进入，也可以点击系统右上角登录用户的下拉菜单进入。基础设置可以修改基础信息，但不能修改部门和角色；密码设置可以修改密码，需要填入旧密码和新密码一起提交，验证通过后才能修改。

图 5-10 个人信息管理展示图

### 5.1.8 每日自检提交模块

普通员工和部门主管需要每日填写自检项目结果，采用下拉菜单选择，自建项目根据用户部门信息返回相应模板。

图 5-11 每日自检提交展示图

### 5.1.9 每周工作汇报模块

图 5-12 每周工作汇报展示图

## 5.2 后端主要接口代码实现

### 5.2.1 HTTP 接口

```
// login
beego.NSRouter("/register", &controllers.StaffController{}, "post:CreateStaff"),
beego.NSRouter("/login", &controllers.StaffController{}, "post:Login"),
beego.NSRouter("/currentUser", &controllers.StaffController{}, "get:CurrentUser"),
// common
```



```
beego.NSRouter("/staff/changePass", &controllers.StaffController{}, "post:ChangePass"),
beego.NSRouter("/staff/changeinfo", &controllers.StaffController{}, "post:ChangeInfo"),
beego.NSRouter("/staff/harddelete", &controllers.StaffController{}, "post:HardDelete"),
beego.NSRouter("/staff/softdelete", &controllers.StaffController{}, "post:SoftDelete"),
// admin
beego.NSRouter("/admin/auditlist", &controllers.AdminController{}, "get:AuditList"),
beego.NSRouter("/admin/stafflist", &controllers.AdminController{}, "get:StaffList"),
beego.NSRouter("/admin/setrole", &controllers.AdminController{}, "post:SetRole"),
beego.NSRouter("/admin/marketingMonthly", &controllers.AdminController{},
"get:MarketingMonthly"),
beego.NSRouter("/admin/marketingMonthlyByID", &controllers.AdminController{},
"get:MarketingMonthlyByID"),
beego.NSRouter("/admin/techSupportMonthly", &controllers.AdminController{},
"get:TechSupportMonthly"),
beego.NSRouter("/admin/afterSaleMonthly", &controllers.AdminController{},
"get:AfterSaleMonthly"),
beego.NSRouter("/admin/developMonthly", &controllers.AdminController{},
"get:DevelopMonthly"),
beego.NSRouter("/admin/productMonthly", &controllers.AdminController{},
"get:ProductMonthly"),
beego.NSRouter("/admin/officeMonthly", &controllers.AdminController{},
"get:OfficeMonthly"),
beego.NSRouter("/admin/financeMonthly", &controllers.AdminController{},
"get:FinanceMonthly"),
beego.NSRouter("/admin/weeklyPlan", &controllers.AdminController{}, "get:WeeklyPlan"),
beego.NSRouter("/admin/weeklySummary", &controllers.AdminController{},
"get:WeeklySummary"),
beego.NSRouter("/admin/annuallyList", &controllers.AdminController{},
"get:AnnuallyList"),
beego.NSRouter("/admin/annuallyItemList", &controllers.AdminController{},
"get:AnnuallyItemList"),
beego.NSRouter("/admin/addAnnuallyPlan", &controllers.AdminController{},
"post:AddAnnuallyPlan"),
beego.NSRouter("/admin/updateAnnuallyPlan", &controllers.AdminController{},
"post:UpdateAnnuallyPlan"),
```

```

beego.NSRouter("/admin/addAnnuallyItem", &controllers.AdminController{},
"post:AddAnnuallyItem"),
beego.NSRouter("/admin/updateAnnuallyItem", &controllers.AdminController{},
"post:UpdateAnnuallyItem"),
beego.NSRouter("/admin/deleteAnnuallyItem", &controllers.AdminController{},
"get:DeleteAnnuallyItem"),
// staff
beego.NSRouter("/staff/marketingDaily", &controllers.StaffController{},
"post:MarketingDaily"),
beego.NSRouter("/staff/techSupportDaily", &controllers.StaffController{},
"post:TechSupportDaily"),
beego.NSRouter("/staff/afterSaleDaily", &controllers.StaffController{},
"post:AfterSaleDaily"),
beego.NSRouter("/staff/developDaily", &controllers.StaffController{}, "post:DevelopDaily"),
beego.NSRouter("/staff/productDaily", &controllers.StaffController{}, "post:ProductDaily"),
beego.NSRouter("/staff/officeDaily", &controllers.StaffController{}, "post:OfficeDaily"),
beego.NSRouter("/staff/financeDaily", &controllers.StaffController{}, "post:FinanceDaily"),
beego.NSRouter("/staff/weeklySummary", &controllers.StaffController{},
"post:AddWeeklySummary"),
beego.NSRouter("/staff/weeklyPlan", &controllers.StaffController{},
"post:AddWeeklyPlan"),

```

### 5.2.2 接口代码实现分析

后端主要是对数据库数据进行增删改查，这里不再对每个接口一一介绍，这里只介绍一下创建用户的流程，其他接口可以依次类推。

Beego 采用的处理模型是 MVC，一个 http 接口对应一个 controller 方法，controller 方法中负责业务逻辑，需要查询操作数据库时调用 models 里的方法，models 里的方法只是对一些数据库增删改查的一层封装，不负责业务。

前端注册或者创建用户时请求的 http 接口是 `http://ip:port/register`，调用的方法是 `CreateStaff`，代码如下：

```

func (c *StaffController) CreateStaff() {
    body := c.Ctx.Input.RequestBody // body 代表前端 http 请求的消息体字节切片
    staff := models.StaffRequest{} // 声明一个 model 对象
    var (
        id int64
        err error
    )
}

```

```

)
if err := json.Unmarshal(body, &staff); err != nil { //用 json 解析数据反射到 staff 上
    logs.Error("[CreateStaff]: %+v", err)
    c.Data["json"] = NewErrorMsg("error", codes.JsonErr, err, nil)
    goto finish
}
if (staff.Department != "" && !models.VerifyDepartment(staff.Department)) ||
(staff.Role != "" && !models.VerifyRole(staff.Role)) {
    c.Data["json"] = NewMsg("unValid", codes.ParameterUnValid, "UnValid
department or role", nil)
    goto finish
}
id, err = staff.Create() // 调用 models 中的方法将数据保存到 mysql 中
if err != nil {
    logs.Error("[CreateStaff]: %+v", err)
    c.Data["json"] = NewErrorMsg("failed", codes.DBExecErr, err, nil)
} else {
    c.Data["json"] = NewDataMsg("ok", codes.Success, id) // http 响应的消息体内容
}
finish: c.ServeJSON() // 处理结束返回 json 格式的响应
}

```

staff.Create 的代码如下：

```

func (staff *StaffRequest) Create() (int64, error) {
    o := orm.NewOrm() // 申请数据库操作对象
    encryptPass := GeneratePassword(staff.Password) // 将密码加密
    if encryptPass == "" {
        return 0, errors.New("internal error with encrypt")
    }
    status := GuestStatus
    if staff.Role != "" {
        switch staff.Role {
        case StaffRole, LeaderRole:
            status = NormalStatus
        case BossRole:
            status = BossStatus

```

```

    }
}
// 执行插入语句，将数据保存到 staff 表中
res, err := o.Raw("INSERT INTO
staff(nickname,password,name,join_date,mobile,mail,department,role,status)
VALUES(?,?,?,?,?,?,?,?)",staff.Nickname,encryptPass,staff.Name,staff.JoinDate.ToTime(),st
aff.Mobile,staff.Mail,staff.Department,staff.Role,status).Exec()
if err != nil {
    return 0, err
}
id, _ := res.LastInsertId()
return id, nil
}

```

### 5.3 项目目录介绍

antd-ui 是前端文件，其中 config/router.config.js 是路由页面信息，src/pages 包含主要页面渲染和交互的实现；beego-server 是后端工程代码，其中 routers 是 HTTP 接口路由，controllers 代表接口处理方法，models 代表底层数据处理的封装；deploy 是项目工程部署的文件，其中 mysql/tables.sql 是初始化数据库时用到的建表语句，nginx.conf 是前端部署运行时 nginx 的 server 配置，docker-compose.yml 是部署配置文件。

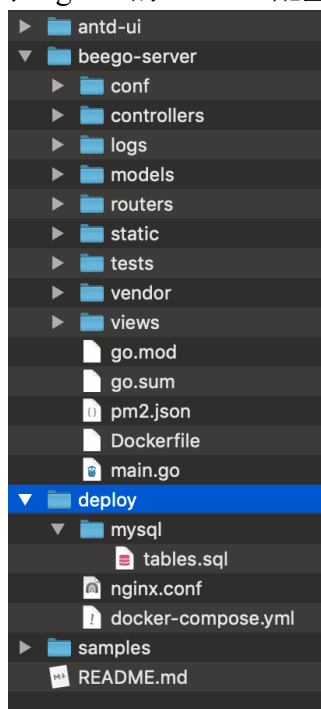


图 5-13 工程目录展示图

### 5.3 项目测试方法介绍

本工程属于前后端分离独立的工程，一般前端开发过程中因为没有后端接口的提供，无法完整的测试页面功能或者展示和操作数据，本工程借助 umi 的 mock 功能，前端可以模拟网络请求然后自己生成一些测试接口。后端因为 golang 语言本身的特性支持，可以方便地对实现的函数进行单元测试、基准测试。

### 5.4 本章小结

本章节主要是对员工周报管理系统的具体实现进行了讲解，因为涉及较多，只讲述重要部分。整个工程的代码实现分为前端、后台两部分，其他工作还包含数据库的搭建和部署方案的集成，前端着重考虑了功能和交互，保证用户的使用体验同时满足功能需求；后端着重考虑接口的简洁性，保证前后端数据请求的正确性，同时保证后端数据库增删改查到位；工程的环境搭建和部署追求简单快速，所以数据库的简历和表的初始化工作均交给容器实现，前端展示交给 nginx 镜像实现，后端也依托容器编译运行，最终只需要 docker-compose 即可一键部署运行。

## 结论

本文主要介绍员工周报管理系统的设计与实现过程。从一开始探讨课题背景，研究发张现状，弄清课题的实际参考价值和意义；然后开始分析工程需求，列出实现目标，方便后续设计与实现的合理分工；接着讲述技术选型和工程的设计细节，包括页面如何展示、如何交互、涉及到哪些数据请求接口、后端数据库如何存储等等；最后介绍实现细节，提供系统的运行逻辑和组织架构。

系统基本达到预期要求，预期实现的功能也都一一实现，系统能够安全稳定地运行。在系统的设计中，主要工作如下：

- 1) 对员工周报管理系统的使用背景做了详尽的需求分析，拆分实现目标。
- 2) 给系统用户划分角色和使用权限，使得数据做到相互隔离，保证安全性和保密性。
- 3) 按照需求分析的内容，对系统进行了详细的功能设计，确定了系统的部署架构和功能架构，完成了对每个子模块的功能设计。
- 4) 根据数据模型的相互关系，设计建立底层数据库表和字段，保证后端运行时处理业务的逻辑尽量简单。
- 5) 不仅完成整个系统的功能开发，还要保证每一模块的测试和连接，最终系统能够稳定安全地运行。

在对系统做出了完整的设计后，发现了许多自己在设计时的问题，而这些问题为我以后的工作起到了警示作用，提醒了我在以后的工作中需要更认真、更仔细。以下是在完成系统开发后发现的问题：

- 1) 在开发系统前，应该对自己要做的工作做一个简单的规划，不论是系统的规划还是时间的规划，以免在设计过程中遇到瓶颈和因时间规划不合理导致的时间浪费。
- 2) 系统的开发不仅要做到功能全面，还要做到页面合理，要合理的设计页面布局，设计出人机交互友好的系统。

## 参考文献

- [1] 从消费互联网到产业互联网 行动!代号“TO B”[J]. 商学院,2019(04):47.
- [2] 张萌. 信息化在企业管理中的作用[J]. 现代信息科技,2019(10):1-2.
- [3] 程墨. 深入浅出 React 和 Redux[M].机械工业出版社,2017:33-34.
- [4] 李书杰,李志刚. B/S 三层体系结构模式[J]. 河北理工学院学报,2002(S1):25-28+34.
- [5] 张楠. 互联网时代下的人力资源管理新思维探讨[J]. 中国市场,2019(18):186-187.
- [6] 连九研. 企业知识管理系统的设计和实现[D].北京交通大学,2011.
- [7] 施炜. 管理架构师:如何构建企业管理体系[M]. 中国人民大学出版社,2019:35-196.
- [8] 尹晓峰. 人力资源管理必备制度与表格规范[M]. 北京联合出版公司, 2015:53-96.
- [9] 胡宸. IT 项目管理系统的设计与实现[J].电子科技大学 学报, 2013(6):1-16.
- [10] 曾春先. 基于 Web2.0 的知识社区应用框架研究[J]. 重庆电子工程职业学院学报,2011(03).
- [11] 郭威,王建永,李颖,林俊. 基于 B/S 技术的企业员工档案信息化管理系统设计[J]. 电子设计工程,2019,27(06):156-159+164.
- [12] 兰旭辉,熊家军,邓刚. 基于 MySQL 的应用程序设计[J]. 计算机工程与设计, 2004, 25(3):442-443.
- [13] 高礼,高昕. Docker 技术在软件开发过程中的应用研究[J].软件, 2016, 37(3):110-113.
- [14] 钟良侃. Docker 技术在 Web 服务系统中的应用研究[J].电脑知识与技术, 2016, 12(26):123-126.
- [15] Kevin Hoffman. Dan Nemeth.Cloud Native Go[M]. 宋净超,吴迎松,徐蓓,马超,译.中国工信出版集团,2017:36.
- [16] Robin Wieruch. The Road to learn React[M]. Independently published,2018(9).
- [17] Alan A. A. Donovan,Brian W. Kernighan. The Go Programming Language[M]. Addison-Wesley Professional,1 edition,2010.
- [18] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, Premkumar Devanbu. A large scale study of programming languages and code quality in github [J]. 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2014:155-165.
- [19] Alex Banks, Eve Porcello. Learning React: Functional Web Development with React and Redux[M]. O'Reilly Media,2017.
- [20] R. S. Pressman Software Engineering. New York:McGraw-Hill, 2010.
- [21] Bernstein D. Containers and Cloud: From LXC to Docker to Kubernetes[J]. IEEE Cloud Computing, 2015, 1(3):81-84.
- [22] R. Pike The cover story 2013 [online] Available: <https://blog.golang.org/cover>.