基于GIN+GRPC的学生宿舍管理系统的设计与实现

计算机与信息科学学院 智能科学与技术专业 2019级 赵廷云

指导教师 朱德利

**摘 要：**随着计算机网络技术的发展，传统宿舍管理经历着互联网时代的洗礼,于是学生管理系统应运而生。学生宿舍管理系统分析了用户的功能需求，梳理了宿舍管理的相关流程，并对系统进行了结构化分析与设计；结合当下流行的前后端分离的Web开发技术（前端技术主要使用bootstrap、ajax，后端技术主要使用gin gRpc mysql等），实现了角色登录、管理员信息管理、学生信息管理、宿管信息管理、楼宇管理、宿舍管理、入住管理、数据上传导出Excel等功能，极大的提高了宿管对学生宿舍的管理效率。

**关键词：**gin gRpc mysql bootstrap ajax

**Abstract：**With the development of computer network technology, the traditional dormitory management experienced the baptism of the Internet era, so the student management system was born. The student dormitory management system analyzed the functional requirements of users, sorted out the relevant processes of dormitory management, and structured the analysis and design of the system; combined with the popular front-end and back-end separated Web development technology (front-end technology mainly uses bootstrap, ajax, back-end technology mainly uses gin gRpc mysql, etc.), to achieve the role login, administrator information management It has achieved the functions of role login, administrator information management, student information management, dormitory information management, building management, dormitory management, occupancy management, data uploading and exporting to Excel, etc., which has greatly improved the efficiency of dormitory management for students.

**Key words：**go gin gRpc etcd mysql bootstrap ajax

1 背景

##### 随着互联网技术的发展，智能化管理越来越重要。中国大学生的数量日趋上升，学校对于宿舍的管理成本也逐渐增加，宿管阿姨可能要面临着数万大学生的宿舍管理，学生数据量日渐庞大，需要一个平台来集中管理。大学生在宿舍的时间超过了1/3，因此良好的宿舍管理对学生的生活和学习极为关键。学生宿舍管理系统能够有效的帮助学校和宿舍管理员管理宿舍和公共资源，减轻管理压力，同时减少宿舍内同学之间的矛盾，促使学生享受舒适的校园生活。

2 系统分析

2.1 需求分析

2.1.1 需求背景

宿舍管理系统是面向学生与宿管人员的系统，目的是提高学校对学生宿舍的管理与方便学生的宿舍生活。本系统主要包括：学生、宿舍、宿舍管理员、楼宇等基本信息的管理。

2.1.2 需求模块

2.1.2.1 登陆模块

对登陆用户进行用户名和密码验证，用户登录后可以正常使用系统的相关功能，不同角色用户登录系统后对系统功能有相应的操作权限，用户登录前无法使用系统的功能，并会被重定向到登录页面。

2.1.2.2 超级管理员基本信息管理模块

只有角色为超级管理员的用户方可对超级管理员的基本信息进行增删改查以及批量数据的导入导出，角色为宿舍管理员以及学生的用户对此模块不可见；查询功能需要提供根据某些特定字段对数据进行精确查询、模糊查询、范围查询等。

2.1.2.3 宿舍管理员基本信息管理模块

只有角色为超级管理员的用户方可对宿舍管理员的基本信息进行增删改查以及批量数据的导入导出，角色为宿舍管理员以及学生的用户对此模块不可见；查询功能需要提供根据某些特定字段对数据进行精确查询、模糊查询、范围查询等。

2.1.2.4 学生基本信息管理模块

只有角色为超级管理员和宿舍管理员的用户方可对学生的基本信息进行增删改查以及批量数据的导入导出，角色为学生的用户只可以查看学生模块的基本信息，不可以进行新增、修改、删除以及批量数据的导入导出；查询功能需要提供根据某些特定字段对数据进行精确查询、模糊查询、范围查询等。

2.1.2.5 楼宇基本信息管理模块

只有角色为超级管理员的用户方可对楼宇的基本信息进行增删改查以及批量数据的导入导出，角色为宿舍管理员以及学生的用户对此模块不可见；查询功能需要提供根据某些特定字段对数据进行精确查询、模糊查询、范围查询等。

2.1.2.6 宿舍基本信息管理模块

只有角色为超级管理员和宿舍管理员的用户方可对宿舍基本信息进行增删改查以及批量数据的导入导出，角色为学生的用户只可以查看此模块的基本信息，不可以进行新增、修改、删除以及批量数据的导入导出；查询功能需要提供根据某些特定字段对数据进行精确查询、模糊查询、范围查询等。

2.1.2.7 入住基本信息管理模块

只有角色为超级管理员和宿舍管理员的用户方可对入住的基本信息进行增删改查以及批量数据的导入导出，角色为学生的用户对此模块不可见。查询功能需要提供根据某些特定字段对数据进行精确查询、模糊查询、范围查询等。

2.1.3 系统角色

系统分为超级管理员、宿舍管理员、学生三种不同的角色，每种角色在使用系统时都会有不同的功能权限。

2.1.4 对系统功能的一般性规定

界面格式统一，安全性高。

2.1.5 对系统性能的一般性规定

精度：数值要求高，数据量大。

时间特性要求：响应速度快。

扩展性：可扩展。

可靠性：保证50人同时在客户端登陆，系统正常运作。

安全性：有严格的权限管理功能，防止数据丢失破坏。

系统尽量满足高性能、高可用、可扩展

2.1.6 产品运行环境

基于Windows && Mac系统的各种主流浏览器上。

2.2 业务流程分析

2.2.1 业务流程分析

系统主要业务流程

（1）超级管理员管理：对超级管理员的信息进行增删改查，通过Excel导入批量数据到数据库，批量导出数据到Excel

（2）宿管管理：对宿管的信息进行增删改查，通过Excel导入批量数据到数据库，批量导出数据到Excel

（3）学生管理：对学生的信息进行增删改查，通过Excel导入批量数据到数据库，批量导出数据到Excel

（4）楼宇管理：对楼宇的信息进行增删改查，通过Excel导入批量数据到数据库，批量导出数据到Excel

（5）宿舍管理：对宿舍的信息进行增删改查，通过Excel导入批量数据到数据库，批量导出数据到Excel

（6）入住管理：对入住信息进行增删改查，通过Excel导入批量数据到数据库，批量导出数据到Excel

2.2.2 业务流程图

流程分析在系统分析中是必不可少的，超级管理员、宿管和学生作为本系统的主要用户群体，应该有完善、流畅的操作流程[1]。

（1）学生管理子系统业务流程

据调查分析：学生信息管理子系统的业务流程主要是：超级管理员或者宿舍管理员会向系统中录入学生信息，并且需要对有问题或者过期的数据进行修改和删除，同时可以批量的导入导出学生数据到Excel，超级管理员或者宿舍管理员可以通过学生编号等关键字进行精确搜索。

其中主要业务流程图，如下图所示：

🤮图2-1 学生信息管理业务流程图

（2）超级管理员管理子系统业务流程

据调查分析：超级管理员信息管理子系统的业务流程主要是：超级管理员会向系统中录入超级管理员信息，并且需要对有问题或者过期的数据进行修改和删除，同时可以批量的导入导出数据到Excel，超级管理员可以通过编号等关键字进行精确搜索。

其中主要业务流程图，如下图所示：

图2-2 超管信息管理业务流程图

（3）宿管管理子系统业务流程

据调查分析：宿管信息管理子系统的业务流程主要是：超级管理员会向系统中录入信息，并且需要对有问题或者过期的数据进行修改和删除，同时可以批量的导入导出数据到Excel，超级管理员可以通过编号等关键字进行精确搜索。

其中主要业务流程图，如下图所示：

图2-3 宿管信息管理业务流程图

（4）楼宇管理子系统业务流程

据调查分析：楼宇信息管理子系统的业务流程主要是：超级管理员会向系统中录入学生信息，并且需要对有问题或者过期的数据进行修改和删除，同时可以批量的导入导出数据到Excel，超级管理员可以通过编号等关键字进行精确搜索。

其中主要业务流程图，如下图所示：

图2-4 楼宇信息管理业务流程图

（5）宿舍管理子系统业务流程

据调查分析：宿舍信息管理子系统的业务流程主要是：超级管理员或者宿舍管理员会向系统中录入信息，并且需要对有问题或者过期的数据进行修改和删除，同时可以批量的导入导出数据到Excel，超级管理员或者宿舍管理员可以通过编号等关键字进行精确搜索。

其中主要业务流程图，如下图所示：

图2-5 宿舍信息管理业务流程图

（6）入住管理子系统业务流程

据调查分析：入住信息管理子系统的业务流程主要是：超级管理员或者宿舍管理员会向统中录入信息，并且需要对有问题或者过期的数据进行修改和删除，同时可以批量的导入导出数据到Excel，超级管理员或者宿舍管理员可以通过编号等关键字进行精确搜索。

其中主要业务流程图，如下图所示：

图2-6 入住信息管理业务流程图

3 系统设计与实现

3.1 技术选型

系统采用前后端分离的架构模式来开发，前后端分离已成为互联网项目开发的业界标准使用方式，可以有效的对前后端代码进行解耦，并且前后端分离会为以后的大型分布式架构、弹性计算架构、微服务架构、多端化服务（多种客户端，例如：浏览器，车载终端，安卓，IOS等等）打下坚实的基础。这个步骤是系统架构从猿进化成人的必经之路。

核心思想是前端html页面通过ajax调用后端的restuful api接口并使用json数据进行交互。

3.1.1 后端技术选型

对与系统的后端服务采用微服务架构的模式进行设计与落地开发，微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间相互协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务和服务之间采用轻量级的通信机制相互沟通（通常是基于Thrift、Protobuf等高性能的序列化方式)。每个服务都围绕着具体的业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。

3.1.1.1 编程语言

系统后端代码采用GO语言进行开发，GO语言是一种静态类型，可编译，可并行化，并具有垃圾回收的编程语言，通常被称为golang。它不像Java/C++拥有完整的面向对象编程的语法机制，但他的结构体和接口类型的功能足以满足面向对象的编程范式（可以通过结构体嵌套来代替继承）。GO语言不仅在内存占用以及编译执行性能上优于java、python、php等语言，而且它天然支持高并发，GO语言的协程goroutine具有很多的优点，比如占用内存小每个2kb内存，可以轻松创建大量的goroutine；比线程轻量很多启动时间快于线程；原生支持通过channel进行通信，go推荐使用通信来并发而不是内存共享，不用操心锁和同步等。GO语言最初由google的Robot•Griesemer、Rob Pike和Ken Thompson设计，并于2009年11月由谷歌公开发布为开源免费项目。

3.1.1.2 API框架

当前市面上流行的GO语言开源框架有很多，比如Gin、Beego、Echo、Fasthttp等，其中Gin框架支持功能比较完善并且在GitHub上面获得的Star也是比较多的，所以这个项目我选择Gin来作为API框架来提供Restful风格的API。

**一个完整的API框架需要支持许多功能比如：**

Server：作为server，监听端口，接受请求；

Router：路由和分组路由，可以把请求路由到对应的处理函数；

Middleware：支持中间件，对外部发过来的http请求经过中间件处理，再给到对应的处理函数。例如http请求的日志记录、请求鉴权(比如校验token)、CORS支持、CSRF校验等；

template engine：模板引擎，支持后端代码对html模板里的内容做渲染(render)，返回给前端渲染好的html。

**Gin框架除了支持以上功能外还支持很多方便开发者的特色功能：**

Crash-free：崩溃恢复，Gin可以捕捉运行期处理http请求过程中的panic并且做recover操作，让服务一直可用。

JSON validation：JSON验证。Gin可以解析和验证request里的JSON内容，比如字段必填等。

Error management：错误管理。Gin提供了一种简单的方式可以收集http request处理过程中的错误，最终中间件可以选择把这些错误写入到log文件、数据库或者发送到其它系统。

Middleware Extendtable：可以自定义中间件。Gin除了自带的官方中间件之外，还支持用户自定义中间件，甚至可以把自己开发的中间件提交到[官方代码仓库](https://link.zhihu.com/?target=https://github.com/gin-gonic/contrib)里。

3.1.1.3 RPC框架以及序列化协议

RPC是Remote Procedure Call的简称，中文叫远程过程调用。可以这么理解：现在有两台服务器A和B。部署在A服务器上的应用，想调用部署在B服务器上的另一个应用提供的方法，由于不在一个内存空间，不能直接调用，需要通过网络来达到调用的效果。

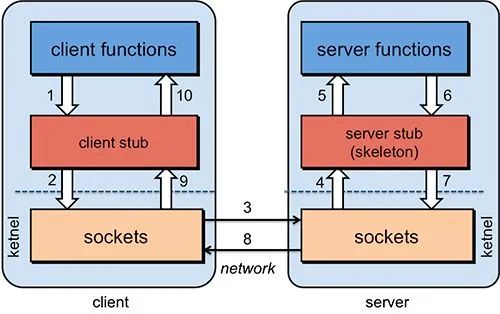


图3-1 rpc调用原理

现在市面上有很多高性能的RPC框架如Spring Cloud、Alibaba开源框架Dubbo、Google的gRPC、Thrift等；本项目选择gRPC框架来做后端微服务开发，gRPC 是一个高性能、通用的开源 RPC 框架，它具有如下特点：

（1）跨语言使用，支持 C++、Java、Go、Python、Ruby、C#、Node.js、Android Java、Objective-C、PHP等编程语言；

（2）基于 IDL 文件定义服务，通过 proto3 工具生成指定语言的数据结构、服务端接口以及客户端Stub；

（3）通信协议基于标准的HTTP/2设计，支持双向流、消息头压缩、单 TCP 的多路复用、服务端推送等特性，这些特性使得 gRPC 在移动端设备上更加省电和节省网络流量；

（4）序列化支持 PB（Protocol Buffer）和JSON，PB是一种语言无关的高性能序列化框架，基于HTTP/2 + PB, 保障了RPC调用的高性能；

（5）安装简单，扩展方便（用该框架每秒可达到百万个RPC）

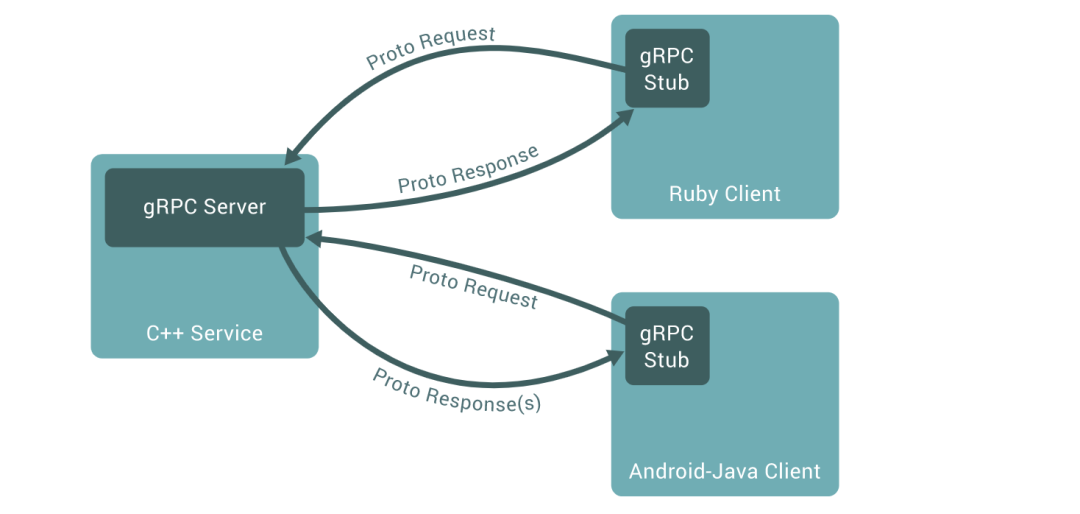


图3-2 grpc实现远程调用原理

**Protocol Buffers**

ProtoBuf 是一种更加灵活、高效的数据格式，与 XML、JSON 类似，在一些高性能且对响应速度有要求的数据传输场景非常适用；ProtoBuf 在 gRPC 的框架中主要有三个作用：定义数据结构、定义服务接口，通过序列化和反序列化方式提升传输效率；我们知道使用 XML、JSON 进行数据编译时，数据文本格式更容易阅读，但进行数据交换时，设备就需要耗费大量的 CPU 在 I/O 动作上，自然会影响整个传输速率。Protocol Buffers不像前者，它会将字符串进行序列化后再进行传输，即二进制传输。

Protobuf和Json两种序列化方式的优劣：

Protobuf

优势：

* 类型安全
* 易用性好
* 序列化/反序列化性能好
* 兼容性好
* 不仅可以定义结构体，还可以定义rpc接口

劣势：

* 可读性较差：没有schema的情况下难以阅读和编辑
* 灵活性较差：无法动态修改schema

json

优势：

* 可读性好：方便理解和编辑
* 易用性好：使用简单
* 灵活性好：支持动态修改schema

劣势：

* 序列化/反序列化性能差
* 编码问题导致解析失败类的

3.1.1.4 注册中心组件

由于gRPC是比较原生的没集成服务发现等功能，需要我们自己去稍微封装一下；这里本项目选择使用etcd作为注册中心组件，也可以选择zookeeper、consul这些高性能的组件。

etcd 是一个分布式键值对存储系统，由coreos 开发，内部采用raft 协议作为一致性算法，用于可靠、快速地保存关键数据，并提供访问。通过分布式锁、leader选举和写屏障(write barriers)，来实现可靠的[分布式](https://so.csdn.net/so/search?q=%E5%88%86%E5%B8%83%E5%BC%8F&spm=1001.2101.3001.7020" \t "_blank)协作。etcd集群是为高可用、持久化数据存储和检索而准备并以一致和容错的方式存储元数据。分布式系统使用 etcd 作为一致性键值存储系统，用于配置管理、服务发现和协调分布式工作。使用 etcd 的通用分布式模式包括领导选举、分布式锁和监控机器活动。

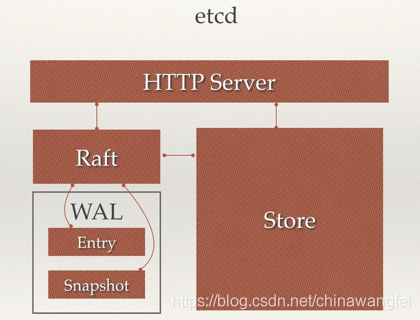


图3-3 etcd架构图

3.1.1.5 数据库以及ORM框架

本项目选择MySQL作为主要业务数据存储引擎，此外可以选择使用Redis作为缓存中间件，ElasticSearch作为服务日志的存储引擎，由于它的高效全文检索性能可以快速检索到日志。

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 最流行的关系型数据库管理系统，在 WEB 应用方面MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。 MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

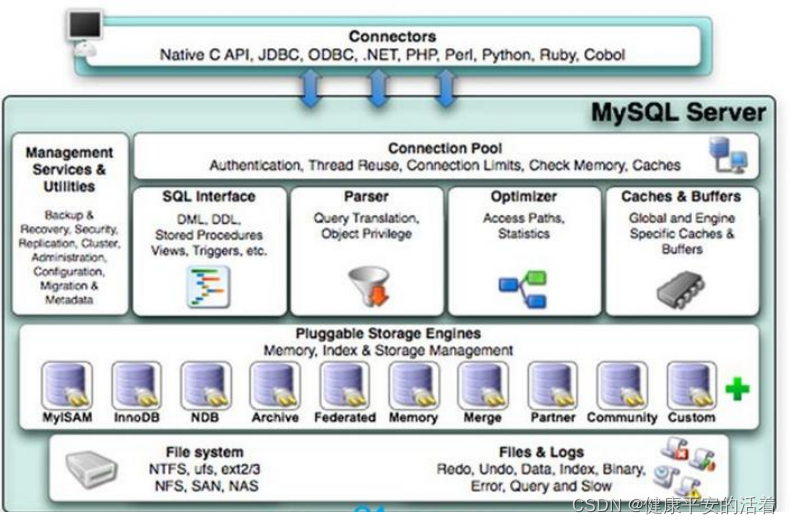


图3-4 mysql架构图

项目采用GORM来和持久层数据库进行交互，gorm是面向golang语言的一种[ORM](https://so.csdn.net/so/search?q=ORM&spm=1001.2101.3001.7020" \t "_blank)(持久层)框架，支持多种数据库的接入，例如MySQL，PostgreSQL，SQLite，SQL Server，Clickhouse。此框架的特点，弱化了开发者对于sql语言的掌握程度，使用提供的API进行底层数据库的访问。

3.1.1.6 用户身份验证（后端接口鉴权）

后端API层采用JWT来做用户身份验证和端接口鉴权，全程为Json Web Token。**JWT的本质就是一个字符串**，它是将用户信息保存到一个Json字符串中，然后进行编码后得到一个JWT token，**并且这个**JWT token**带有签名信息，接收后可以校验是否被篡改**，所以可以用于在各方之间安全地将信息作为Json对象传输。JWT的认证流程如下：

·首先，前端通过Web表单将自己的用户名和密码发送到后端的接口，这个过程一般是一个POST请求。建议的方式是通过SSL加密的传输(HTTPS)，从而避免敏感信息被嗅探

·后端核对用户名和密码成功后，将包含用户信息的数据作为JWT的Payload，将其与JWT Header分别进行Base64编码拼接后签名，形成一个JWT Token，形成的JWT Token就是一个如同lll.zzz.xxx的字符串

·后端将JWT Token字符串作为登录成功的结果返回给前端。前端可以将返回的结果保存在浏览器中，退出登录时删除保存的JWT Token即可

·前端在每次请求时将JWT Token放入HTTP请求头中的Authorization属性中(解决XSS和XSRF问题)

·后端检查前端传过来的JWT Token，验证其有效性，比如检查签名是否正确、是否过期、token的接收方是否是自己等等

·验证通过后，后端解析出JWT Token中包含的用户信息，进行其他逻辑操作(一般是根据用户信息得到权限等)，返回结果

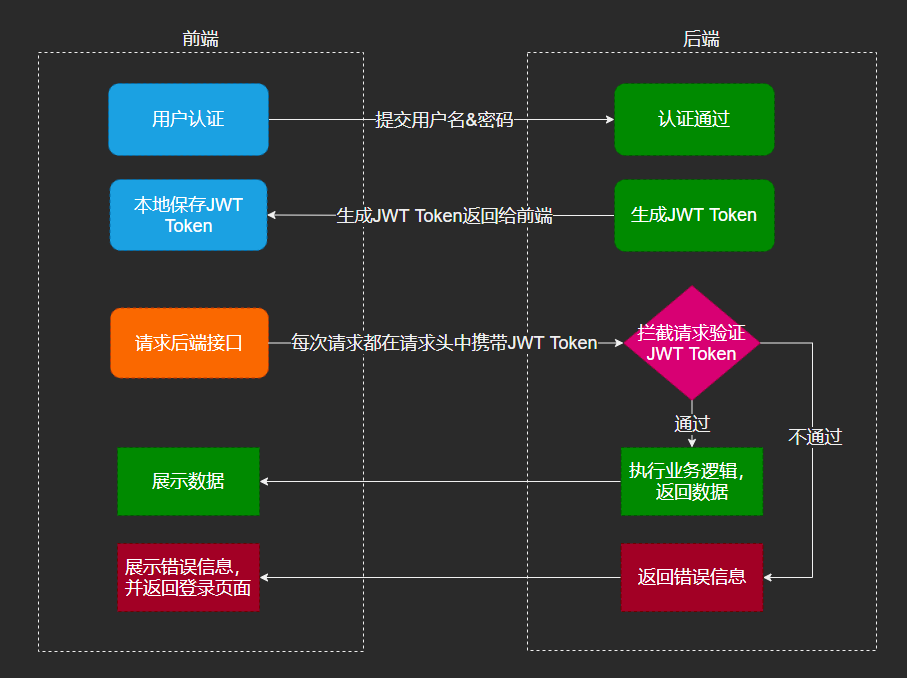


图3-4 jwt认证流程图

3.1.1.7 设计模式

MVC设计模式：

MVC是Model（数据层）、View（视图层）、Controller（控制层）的缩写。在MVC设计模式下，所有的程序无论简单或是复杂都可以将其分为M、V、C三个部分来实现：

（1）最上面的一层，是直接面向最终用户的"视图层"（View）。它是提供给用户的操作界面，是程序的外壳。

（2）最底下的一层，是核心的"数据层"（Model），也就是程序需要操作的数据或信息。

（3）中间的一层，就是"控制层"（Controller），它负责根据用户从"视图层"输入的指令，选取"数据层"中的数据，然后对其进行相应的操作，产生最终结果。

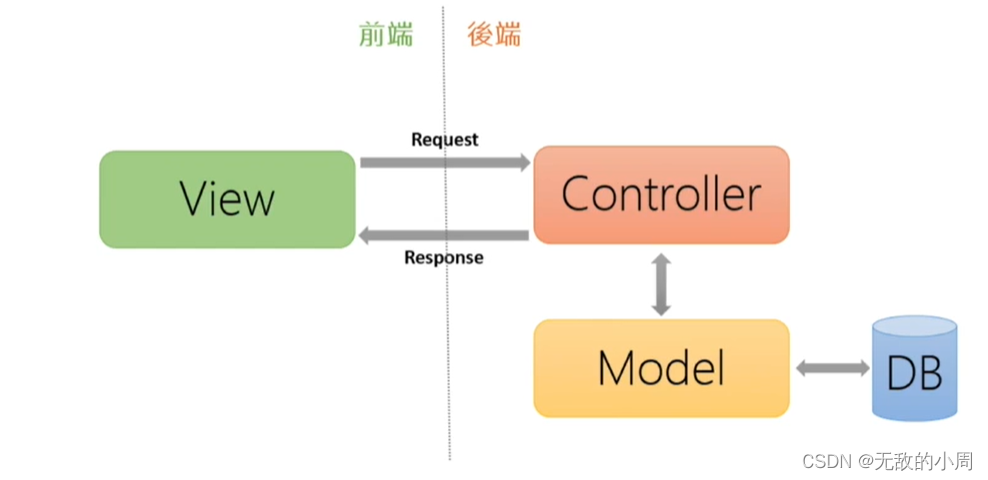


图3-5 MVC模式架构图

DDD领域驱动设计模式：

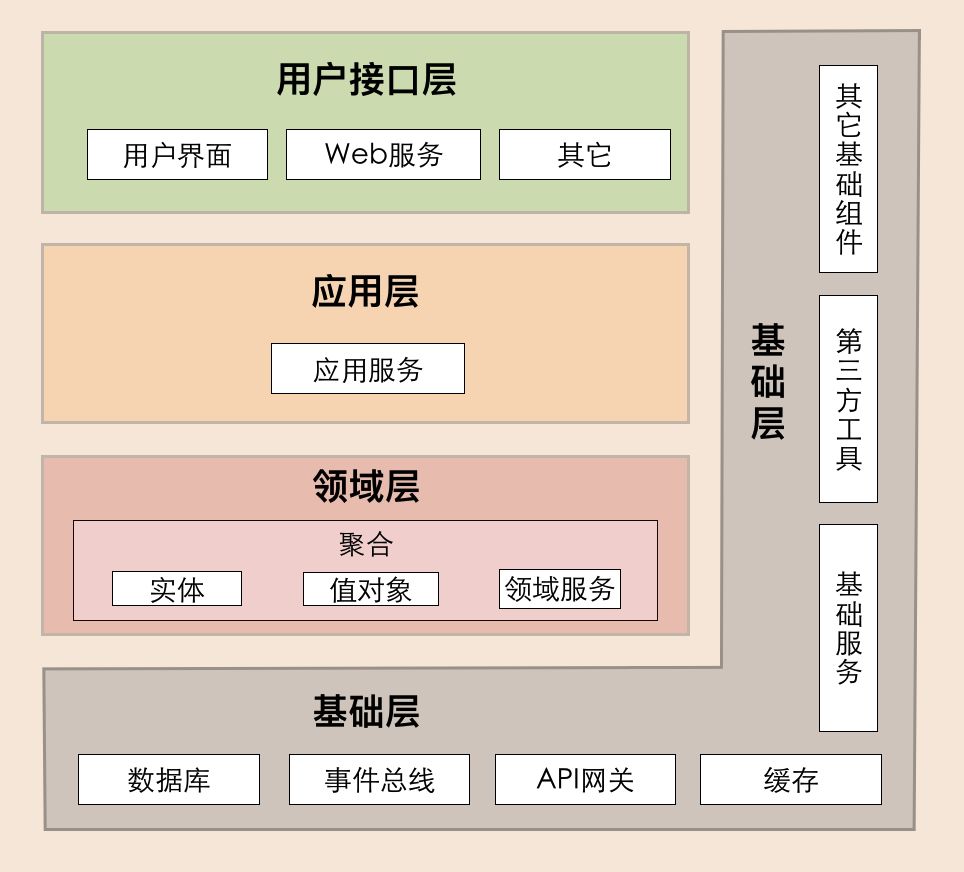


图3-6 DDD模式架构图

3.1.1.8 其他

在开发过程中除了以上技术问题需要解决外还有一些其他的问题和三方包的引入，如开发过程中常见的跨域问题。

跨域问题解决：

跨域问题在前后端分离的项目中大多数都会遇到，项目采用CORS这种规范化的跨域请求解决方案，他的优点在于可以在服务端进行控制是否允许跨域，可以自定义规则并且支持各种请求方式。

3.1.2 前端技术选型

系统的前端模块采用html+css+JavaScript+jquery以及bootstrap框架进行开发；Bootstrap是Twitter推出的一个用于前端开发的开源工具包，是一个做网页的框架，用它提供的样式和组件可以快速开发网站；它简洁灵活，使得Web开发更加快捷，可以用来快速创建一个响应式网站。

前端采用Ajax来和后端API接口进行数据传输， Ajax是一种用来改善用户体验的技术，其本质是利用浏览器提供的一个特殊的对象（XMLHttpRequest,也可称之为ajax对象）向服务器发送异步请求。服务器利用这些数据对当前页面做部分更新。整个过程，页面无刷新，不打断用户的操作。

3.2 系统架构设计

项目整体架构图如下：

图3-7 宿舍管理系统架构图

3.3 数据模型（数据库）设计

3.3.1 概念结构设计

“概念结构设计是根据用户需求分析的结果，设计数据库的概念数据模型（简称概念模型），目前用于建立概念模型的方法有许多种，其中最常用、最著名的方法是实体-联系方法，即E-R图法。

学生宿舍管理系统整体数据模型E-R图：



图3-8 学生宿舍管理系统E-R图

3.3.2 逻辑结构设计

逻辑结构设计是将概念结构设计阶段完成的概念模型转换成能被选定的数据库管理系统支持的数据结构模型（简称数据模型），数据模型可由E-R模型转换而来。下文是将E-R图通过转换得到该系统的逻辑结构。

学生（ID、编号、姓名、性别、密码、是否删除、版本、创建时间、修改时间）

超级管理员（ID、姓名、密码、是否删除、版本、创建时间、修改时间）

宿舍管理员（ID、编号、姓名、性别、密码、是否删除、版本、创建时间、修改时间）

楼宇（ID、宿舍管理员、名字、地址、是否删除、版本、创建时间、修改时间）

宿舍（ID、编号、所属楼宇、层数、已住人数、最多可住人数、是否删除、版本、创建时间、修改时间）

入住信息（ID、学生、宿舍、入住日期、是否删除、版本、创建时间、修改时间）

以下为数据库中的数据表设计。

表3.1 学生表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| 学生ID | Int | 64 | Not Null | 主关键字 |
| 编号 | Character | 100 | Not Null |  |
| 名字 | Character | 20 | Not Null |  |
| 性别 | Character | 10 | Not Null |  |
| 密码 | Character | 40 | Not Null |  |
| 是否删除 | Int | 32 | Not Null |  |
| 版本号 | Int | 32 | Not Null |  |
| 创建时间 | Timestamp | 32 | Not Null |  |
| 修改时间 | Timestamp | 32 | Not Null |  |

表3.2 超级管理员表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| 管理员ID | Int | 64 | Not Null | 主关键字 |
| 名字 | Character | 20 | Not Null |  |
| 密码 | Character | 40 | Not Null |  |
| 是否删除 | Int | 32 | Not Null |  |
| 版本号 | Int | 32 | Not Null |  |
| 创建时间 | Timestamp | 32 | Not Null |  |
| 修改时间 | Timestamp | 32 | Not Null |  |

表3.3 宿舍管理员表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| 宿管ID | Int | 64 | Not Null | 主关键字 |
| 编号 | Character | 100 | Not Null |  |
| 名字 | Character | 20 | Not Null |  |
| 性别 | Character | 10 | Not Null |  |
| 密码 | Character | 40 | Not Null |  |
| 是否删除 | Int | 32 | Not Null |  |
| 版本号 | Int | 32 | Not Null |  |
| 创建时间 | Timestamp | 32 | Not Null |  |
| 修改时间 | Timestamp | 32 | Not Null |  |

表3.4 楼宇表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| 楼宇ID | Int | 64 | Not Null | 主关键字 |
| 宿舍管理员ID | Int | 64 | Not Null | 外键 |
| 名字 | Character | 20 | Not Null |  |
| 地址 | Character | 40 | Not Null |  |
| 是否删除 | Int | 32 | Not Null |  |
| 版本号 | Int | 32 | Not Null |  |
| 创建时间 | Timestamp | 32 | Not Null |  |
| 修改时间 | Timestamp | 32 | Not Null |  |

表3.5 宿舍表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| 宿舍ID | Int | 64 | Not Null | 主关键字 |
| 编号 | Character | 100 | Not Null |  |
| 楼宇ID | Int | 64 | Not Null | 外键 |
| 层数 | Int | 32 | Not Null |  |
| 已住人数 | Int | 32 | Not Null |  |
| 最多可住人数 | Int | 32 | Not Null |  |
| 是否删除 | Int | 32 | Not Null |  |
| 版本号 | Int | 32 | Not Null |  |
| 创建时间 | Timestamp | 32 | Not Null |  |
| 修改时间 | Timestamp | 32 | Not Null |  |

表3.6入住表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| 入住ID | Int | 64 | Not Null | 主关键字 |
| 学生ID | Int | 64 | Not Null | 外键 |
| 宿舍ID | Int | 64 | Not Null | 外键 |
| 入住时间 | Character | 40 | Not Null |  |
| 是否删除 | Int | 32 | Not Null |  |
| 版本号 | Int | 32 | Not Null |  |
| 创建时间 | Timestamp | 32 | Not Null |  |
| 修改时间 | Timestamp | 32 | Not Null |  |

4 项目部署运行

4.1 系统开发工具与开发环境

学生宿舍管理系统采用B/S模式，以Web网页的形式呈现给用户；本平台在window10操作系统上开发，借助Visio工具进行系统业务流程图、架构功能模块图、数据模型ER图的设计，分别使用vscode和Goland集成开发环境来完成整个系统前后端代码的编写。

4.2 项目部署

4.2.1 前端部署

前端服务通过tomcat服务器进行部署（tomcat服务器的端口为8848），如果后续需要通过集群方式部署高可用服务到云端服务器，可以使用nginx来做负载均衡流量分发。

部署后启动tomcat服务器即可：

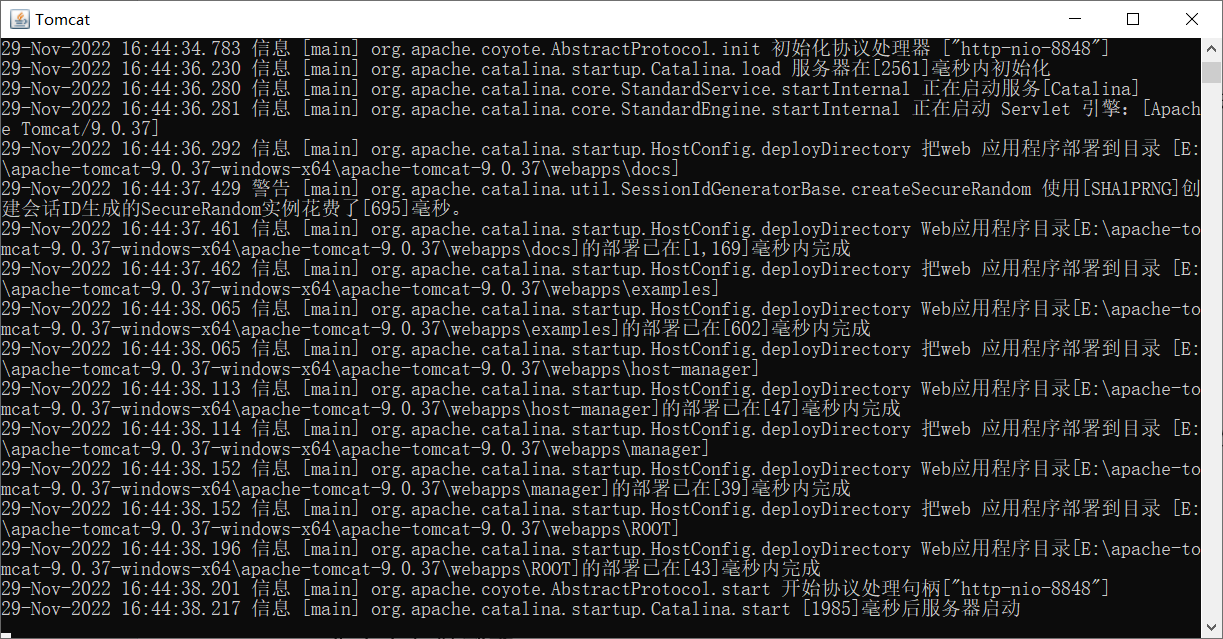


图4-1 前端服务tomcat启动图

4.2.2 后端部署

后端服务编译为可执行文件后直接运行在本地相应的服务端口，需要先启动组件服务（Mysql、Redis、ES、ETCD等，windows系统Mysql、Redis服务开机后会自动启动，linux系统需要手动启动）

启动ETCD：

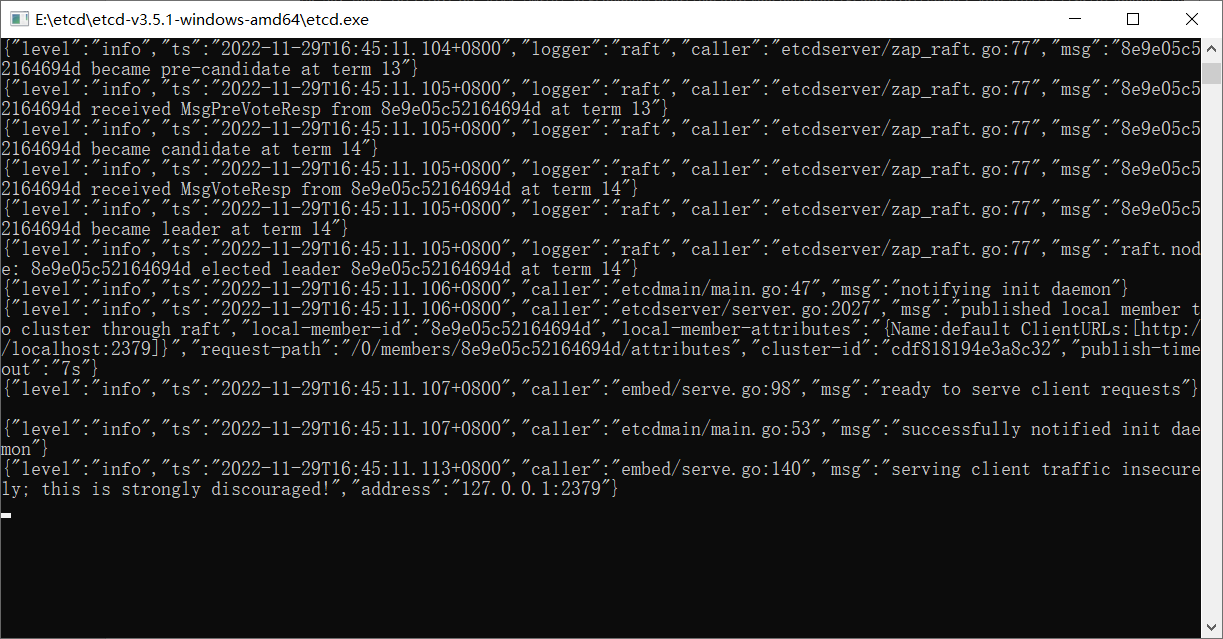


图4-2 注册中心etcd启动图

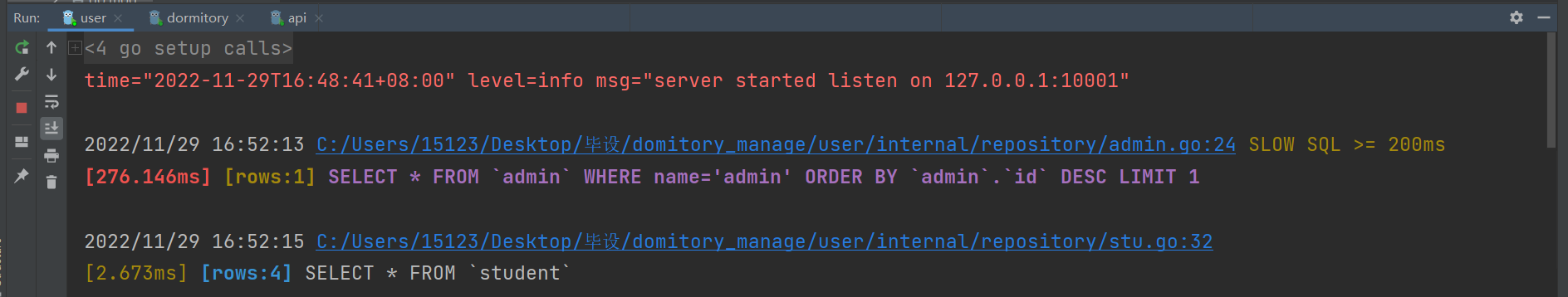


图4-3 user服务启动图

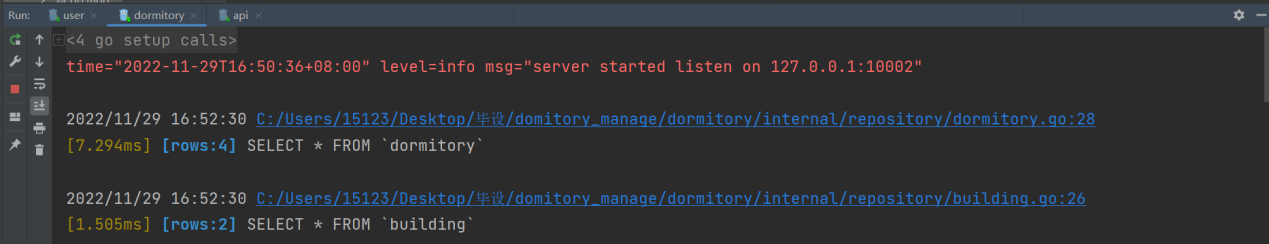


图4-4 dormitory服务启动图

通过etcd keeper可以看到RPC服务已经成功注册到etcd



图4-5 etcd keeper

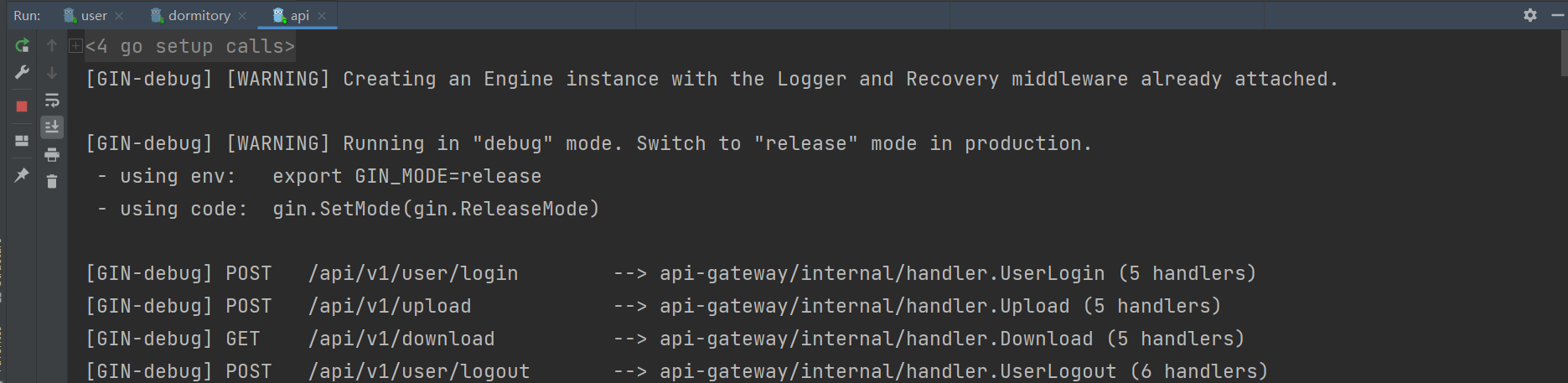


图4-6 api服务启动图

4.3 项目运行

4.3.1 用户登录



图4-7 用户登陆界面

4.3.2 学生管理



图4-8 学生管理界面

4.3.3 管理员管理

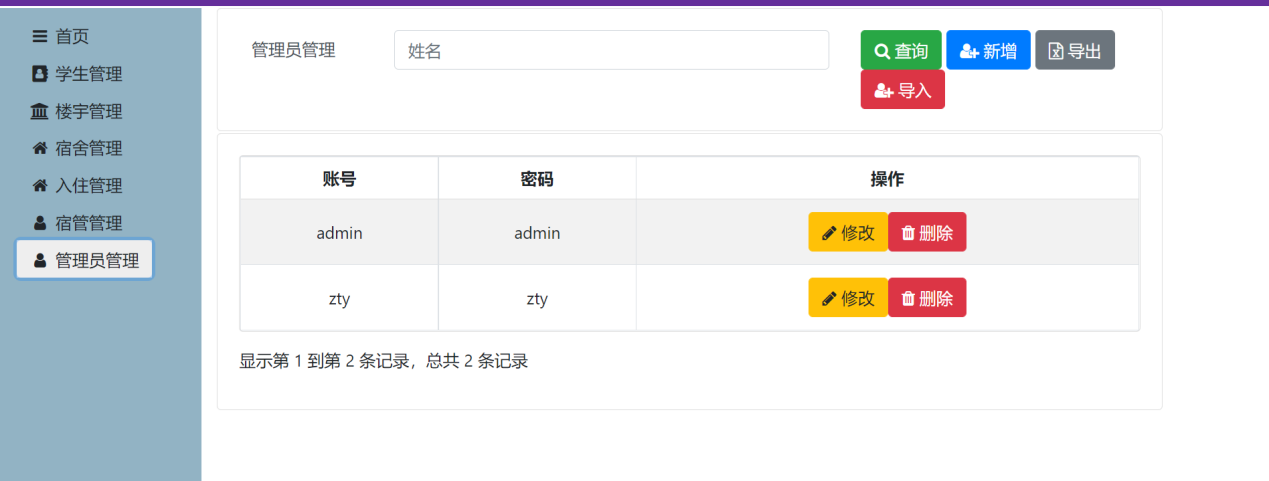


图4-9 管理员管理界面

4.3.4 宿管管理



图4-10 宿管管理界面

4.3.5 楼宇管理



图4-11 楼宇管理界面

4.3.6 宿舍管理



图4-12 宿舍管理界面

4.3.7 入住管理



图4-13 入住管理界面

参考文献：

[1]向阳.信息系统分析与设计[M].机械工业出版社，2014.

[2][Protocol Buffers Language Guide](https://developers.google.com/protocol-buffers/docs/proto). Google Developers. [2016-04-21].

[3]马博文.gRPC官方文档中文版，2018

[4]Gin contributors.gin官方文档 https://github.com/gin-gonic/gin

.