1. 需求分析

题目：高校宿舍可视化系统的实际与实现

主要需求：

- 学生信息录入、修改、删除、查询
- 宿舍管理评分
- 学生早起率、懒床率
- 学生宿舍打扫频率
- 学生晚归情况
- 楼层管理

考虑到实用性，该系统需要拆分为两大子系统，一个是学生端系统，一个是后台管理端系统。学生端系统主要提供给学生使用，负责一些宿舍记录及个人信息记录的基本操作；后台管理模块则是主要负责对所有学生信息的整理，提供宿舍管理、楼层管理、数据查看等权限，提供给宿舍管理员使用的。

1.1 学生系统

学生系统拥有以下功能：

- 创建账户
- 分配宿舍
- 填写个人信息
- 修改个人信息
- 起床打卡（用于统计懒床率）
- 归宿登记（用于统计晚归情况）
- 打扫记录（用于统计宿舍打扫频率）
- 查看宿日常数据

1.2 管理系统

管理系统拥有以下功能：

- 楼层管理
- 宿舍评价
- 宿舍信息管理
- 学生信息查看
- 保洁人员管理
- 统计学生早起率
- 统计学生宿舍打扫频率
- 统计学生晚归

超级管理员在享有上述管理员同等权限的同时额外拥有如下功能：

- 创建管理员
- 创建宿舍楼
- 为宿舍楼分配管理员
- 为宿舍楼分配保洁人员

3. 技术分析

前端：

- Vue 作为基础框架
- vue-router 控制路由（hash 模式）
- vuex 状态管理
- axios 接入数据

后台（Nodejs）：

- Koa 作为基础框架
- koa-router —— 服务端路由控制
- koa-static —— 读取静态文件
- koa-jwt —— JWT 登录校验
- koa-body —— http body 数据处理
- koa-compress —— Gzip 压缩
- koa-cors —— CORS 解决跨域问题
- sequelize —— ORM

数据库：

- MySQL

4. 目录介绍

clinet 目录下为前端的项目文件， server 为后台目录下的文件

4.1 Clinet 目录

4.1.1 根目录

public :

HTML 模板和静态资源，参考 vue-cli 官方文档

src :

源码目录

.eslint* :

Eslint 代码规范规则相关

.prettierrc.js :

Prettier 代码格式化相关

vue.config.js :

webpack 规则，无需修改

4.1.2 src 源码目录（重点）

api:

抽离出的单独的请求文件，请求后台的 url

assets:

前端的静态文件资源

components：

前端的页面的组件，重点组件有：

- GroupSelector：级联选择器组件
- RecordTable：记录表格组件
- RoomSelector：房间级联选择器组件

filter：

Vue 过滤器，详见 Vue 官方文档

icons：

icon 图标文件

layout：

最外层的整体布局组件

router：

前端路由系统，详见 Vue-Router 文档

store：

前端状态管理系统，详见 Vuex 文档

utils：

页面中的某会被复用的方法，如计算日期、格式化日期的方法，会被抽离到 utils 中，再在各个页面中被引入

views（重点）：

每个页面的代码都在这里

App.vue：

最外层容器页面

main.js：

入口文件

settings.js：

配置文件（无需更改）

4.2 server 目录

server 目录下的 static 文件为静态文件目录，前端 Vue 源码编译好之后会自动存放到 static 目录下。src 为源码目录，以下介绍为 src 目录下的文件：

app.js：

入口文件，所有的框架在此引入

config.js：

配置文件，除了数据库相关的文件，其余配置项不要更改

utils：

后台某些会被复用的方法，会被抽离到 utils 中，再在各个页面中被引入

struct：

全局中新建的结构体（非重要，涉及 Javascript 面向对象设计）

routes：

后台路由

model：

后台 Model 层（定义数据库表结构）

db：

数据库连接与生成的方法

controller：

后台的 Controller 层

middle：

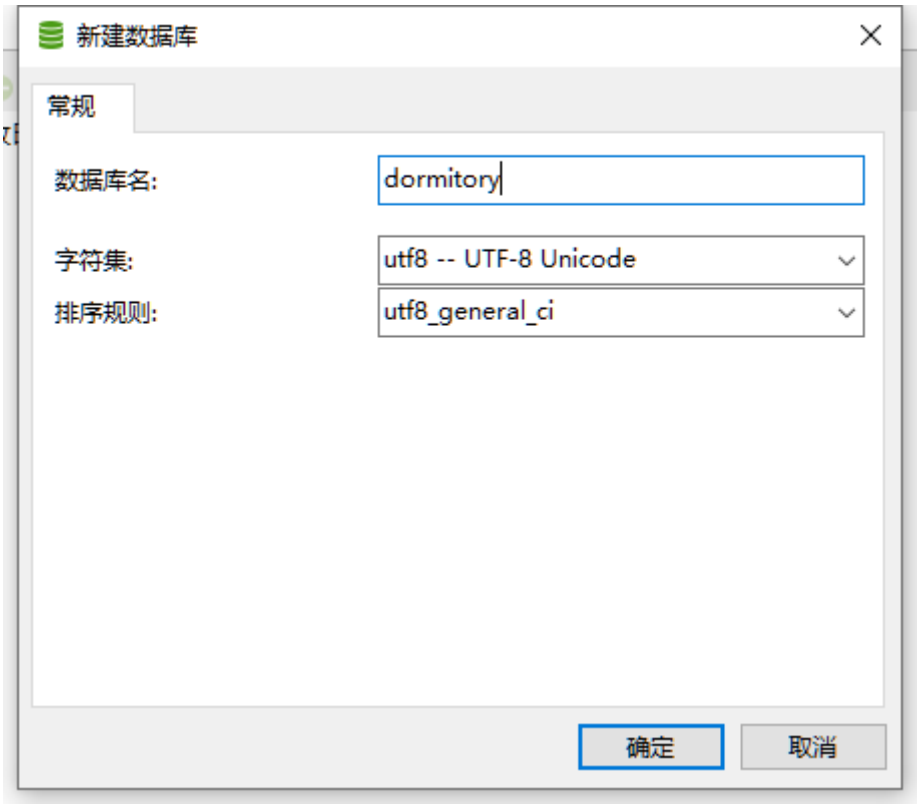
Koa 中间件，详见 Koa 中间件机制。

5. 项目启用方式

首先安装 nodejs 与 npm 环境。

5.1 运行后台

数据库中创建表 "dormitory"（如下为 Navicat 数据库管理工具）：



将 `dormitory.sql` 文件写入该表，完成数据初始化。

修改 `/server/src/config.js` 文件下的数据库地址、用户名、密码。

使用 cmd 或其他命令行工具移动至 server 目录下，运行：

```
## 安装依赖
$ npm install

## 运行项目
$ npm run serve
```

看到以下输出，即为成功：

```
esunr@EsunR_Shinelon MINGW64 /d/Code/Business/Dormitory-management/server (master)
$ npm run start

> dormitory-management-server@1.0.0 start D:\Code\Business\Dormitory-management\server
> node ./src/app.js

DataBase Syncing ... ...
serve running on: http://localhost:8080
DataBase Sync done
```

浏览器输入 `http://localhost:8080`

5.2 前端文件

按照上述步骤已经可以正常运行项目，如果需要修改与调试前端页面请按照以下步骤：

> 首先要确保已经在 `/clinet` 目录下运行过 `npm install` 安装好了前端依赖

1. 在 `/server` 目录下运行 `npm run dev` 开启后台调试模式
2. 在 `/client` 目录下运行 `npm run dev` 开启前端调试模式
3. 修改 `/clinet` 目录下的任意文件
4. 修改完成后，在 `/client` 目录下运行 `npm run build:prod`

6. API 文档

共计 32 个接口，接口文档：https://documenter.getpostman.com/view/6817046/SzKWuHJh