软件设计文档

目录

第一部分 架构设计

一、 架构描述

本系统采用了 MVC（Model-View-Controller）设计模式，包括表现层，业务逻辑层以及数据访问层 , 分析过程使用面向对象方法。

1. 表示层

表示层负责接收用户的命令，交个控制层处理，并显示处理结果。根据使用对象的不同，可以分为三个部分：

1) 学生功能页面
2) 课室预订申请审批部门功能页面
3) 管理员功能页面

2. 业务逻辑层

业务逻辑层可分为以下四个部分

1) 用户登录的交互控制逻辑。使用该系统的用户包括课学生、课室预订审批工作人员、管理员，不同的用户有着不同的权限和交互功能需求。当用户输入登录信息后，控制器从表示层取得用户登录信息，与数据库信息进行匹配，从而配置权限及工作页面。
2) 学生进行查询课室使用情况、预订课室、修改预订、取消预订等操作的控制逻辑。学生在相应的视图页面进行以上操作，控制器从视图层获取用户

的指令或输入的信息等数据，利用这些数据对数据库进行相应的操作，并将结果反馈给用户。

3) 课室预订审批工作人员审批课室申请的控制逻辑。课室预订审批工作人员在相应的视图上输入指令或信息，控制器从视图层获取这些数据，根据这些数据对课室申请相关的数据库进行操作，并将结果反馈给工作人员及学生。

4) 管理员维护学生信息、课室信息、课室申请审批部门信息的控制逻辑。管理员在相应的页面对以上信息数据进行操作，控制器从视图层获取这些操作的相关信息数据，将管理员的操作同步到对应的数据库，并将结果反馈给管理员。

3. 数据访问层

本系统数据模型包括三部分：

1) 系统用户数据模型。其中包括了学生、审批部门工作人员、管理员等用户的登录信息及权限信息。

2) 课室基本信息及预订申请信息数据模型。其中包括了课室的编号、人数、占用情况等基本

3) 信息和课室的预订、审批等信息。

二、 架构图

本系统的架构设计图如下（图 2-1）：

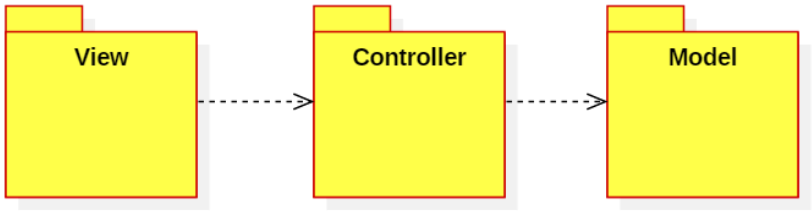
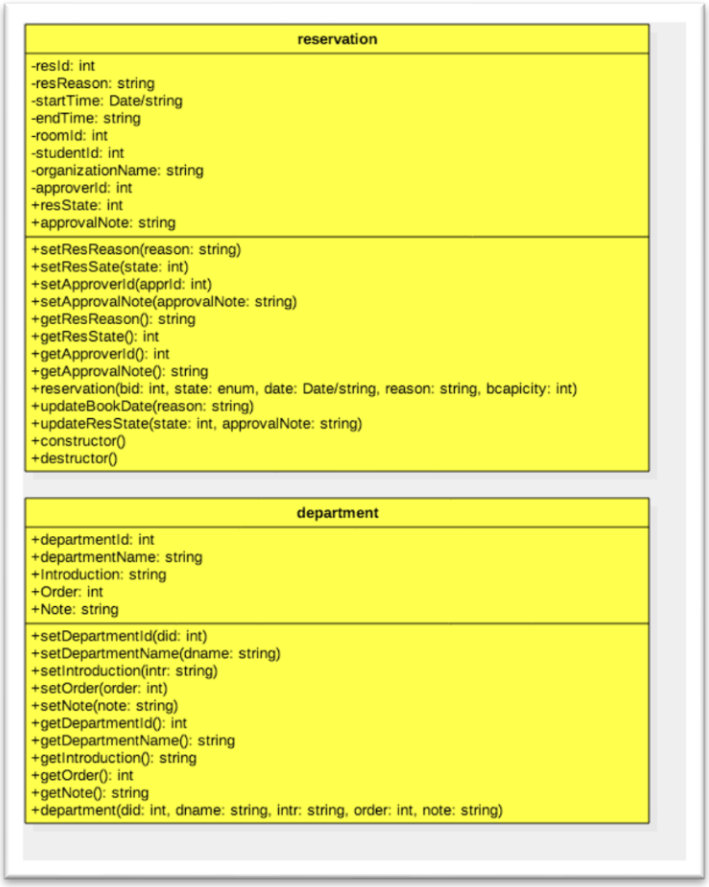


图 2-1 MVC 架构图

三、 关键抽象

经过我们分析，本系统中有 5 个实体类，分别是学生表、管理员表、审批人员表、课室表以及预订表，学生表、管理员表和审批人员表存储了与其相关的一切信息，包含用户名、密码、姓名以及相应的权限，课室表存储了课室的地址和容量，预订表存储了预订信息，包含日期、预订原因等等。如图 2-2 所示。


reservation
-resId: int
-resReason: string
-startTime: Date/string
-endTime: string
-roomId: int
-studentId: int
-organizationName: string
-approverId: int
+resState: int
+approvalNote: string
+setResReason(reason: string)
+setResSate(state: int)
+setApproverId(apprId: int)
+setApprovalNote(approvalNote: string)
+getResReason(): string
+getResState(): int
+getApproverId(): int
+getApprovalNote(): string
+reservation(bid: int, state: enum, date: Date/string, reason: string, bcapicity: int)
+updateBookDate(reason: string)
+updateResState(state: int, approvalNote: string)
+constructor()
+destructor()
department
+departmentId: int
+departmentName: string
+Introduction: string
+Order: int
+Note: string
+setDepartmentId(did: int)
+setDepartmentName(dname: string)
+setIntroduction(intr: string)
+setOrder(order: int)
+setNote(note: string)
+getDepartmentId(): int
+getDepartmentName(): string
+getIntroduction(): string
+getOrder(): int
+getNote(): string
+department(did: int, dname: string, intr: string, order: int, note: string)

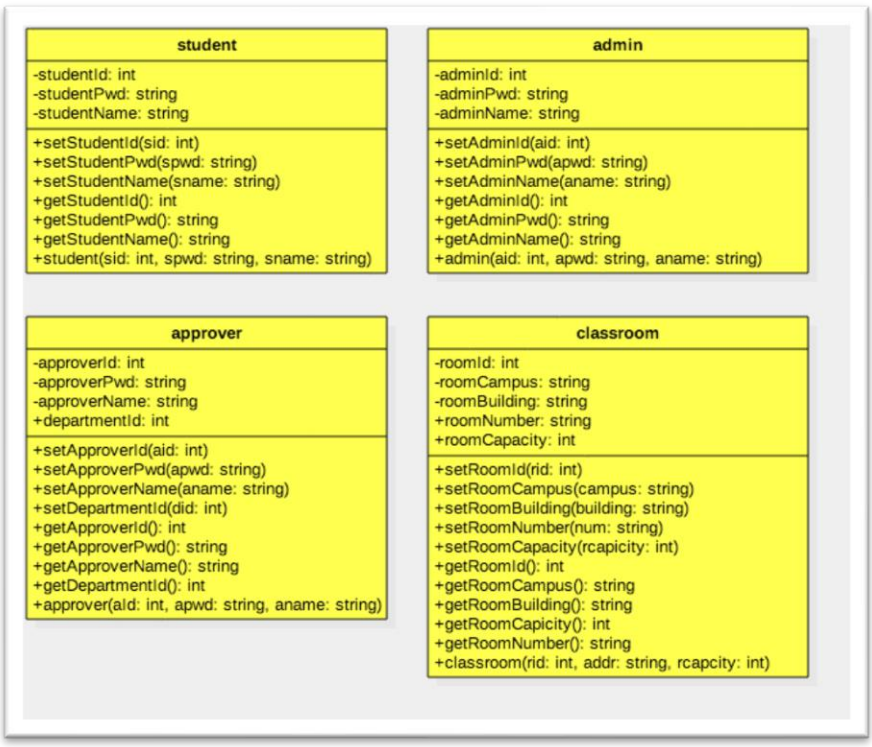

student
-studentId: int
-studentPwd: string
-studentName: string
+setStudentId(sid: int)
+setStudentPwd(spwd: string)
+setStudentName(sname: string)
+getStudentId(): int
+getStudentPwd(): string
+getStudentName(): string
+student(sid: int, spwd: string, sname: string)
admin
-adminId: int
-adminPwd: string
-adminName: string
+setAdminId(aid: int)
+setAdminPwd(apwd: string)
+setAdminName(aname: string)
+getAdminId(): int
+getAdminPwd(): string
+getAdminName(): string
+admin(aid: int, apwd: string, aname: string)
approver
-approverId: int
-approverPwd: string
-approverName: string
+departmentId: int
+setApproverId(aid: int)
+setApproverPwd(apwd: string)
+setApproverName(aname: string)
+setDepartmentId(did: int)
+getApproverId(): int
+getApproverPwd(): string
+getApproverName(): string
+getDepartmentId(): int
+approver(aid: int, apwd: string, aname: string)
classroom
-roomId: int
-roomCampus: string
-roomBuilding: string
+roomNumber: string
+roomCapacity: int
+setRoomId(rid: int)
+setRoomCampus(campus: string)
+setRoomBuilding(building: string)
+setRoomNumber(num: string)
+setRoomCapacity(rcapicity: int)
+getRoomId(): int
+getRoomCampus(): string
+getRoomBuilding(): string
+getRoomCapicity(): int
+getRoomNumber(): string
+classroom(rid: int, addr: string, rcapcity: int)

图 2-2 数据类图

第二部分 子系统及其接口设计

一、 确定设计类

以下是系统的总体类设计图。

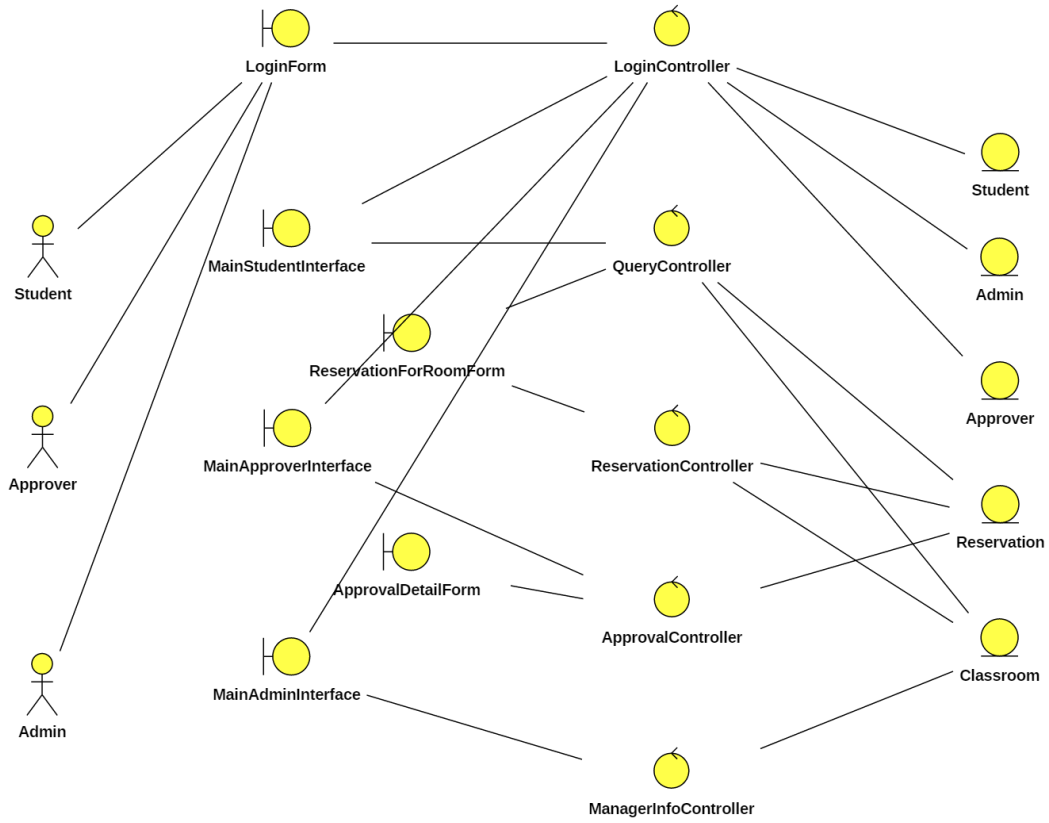


图 2-1 课室预订系统总体类分析

二、 对系统进行子系统设计

经过分析，本课室预订系统的子系统设计如图 2-2 所示，本系统中共有四个子系统，其中 Login 子系统提供用户登录功能，ClassroomReservation 子系统提供课室预订功能，Approval 子系统提供审批预订功能，ManagerInformation 子系统提供系统管理功能。

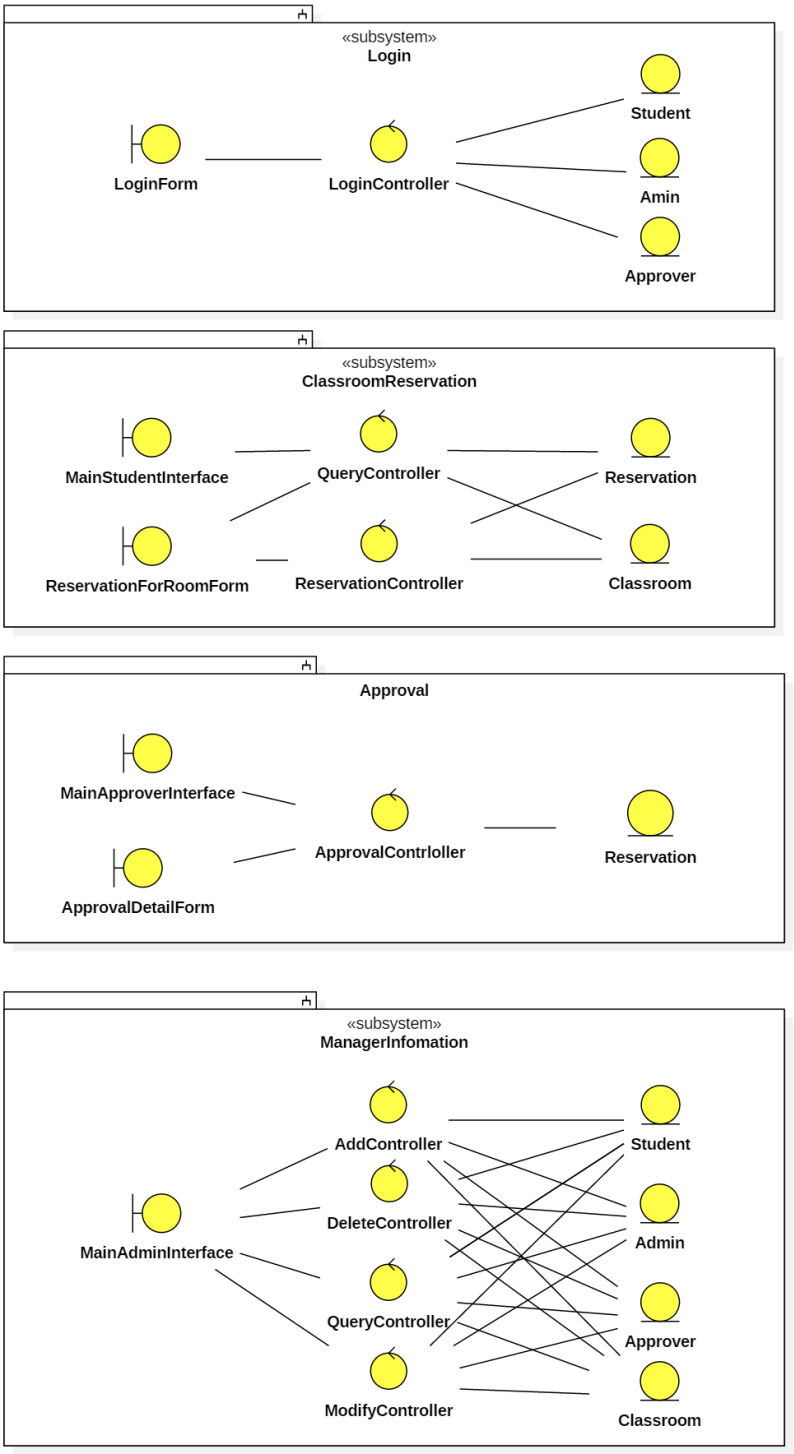


图 2-2 课室预订系统总体类分析

三、 子系统及其接口设计

经过分析，本课室预订系统的子系统及其接口设计图如下图所示：

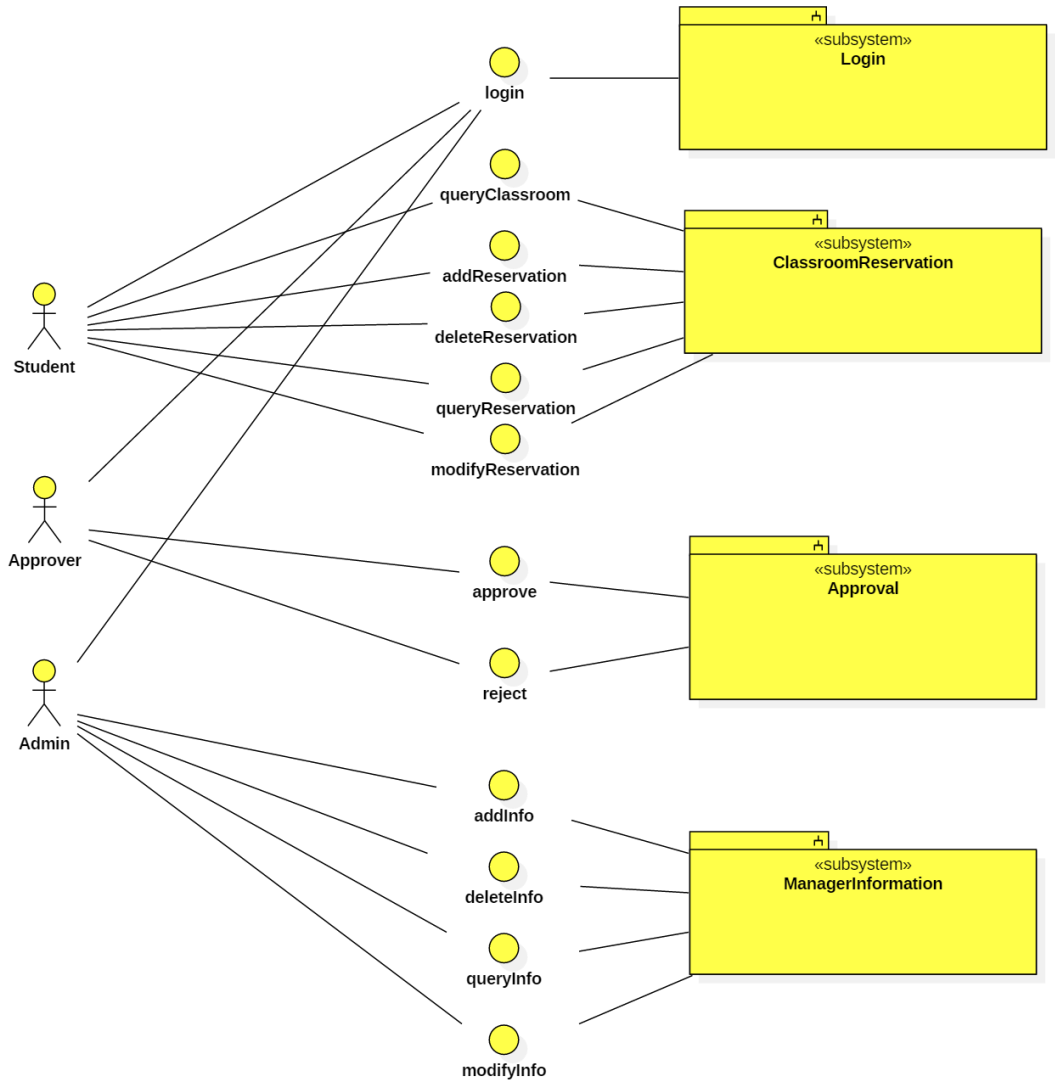


图 2-3 子系统及其接口设计

第三部分 技术选型

一、 后台

后端采用 go 语言作为编程语言，使用面向对象编程(Object-oriented Programming)的软件设计技术完成后台工作。由于系统较为简单，所以并没有按照技术设计文档，将系统划分为子系统而分别实现。同时，go 语言并非面向对象的编程语言，但是它的一些特性却使得它可以使用面向对象的思想进行编程。面向对象编程的思想是：包装(Encapsulation)、信息隐藏(Information Hiding)、数据抽象(Data Abstraction)、继承(Inheritance)、多态(Polymorphism)。系统的开发，使用了这些思想来实现。

1. 具体技术介绍

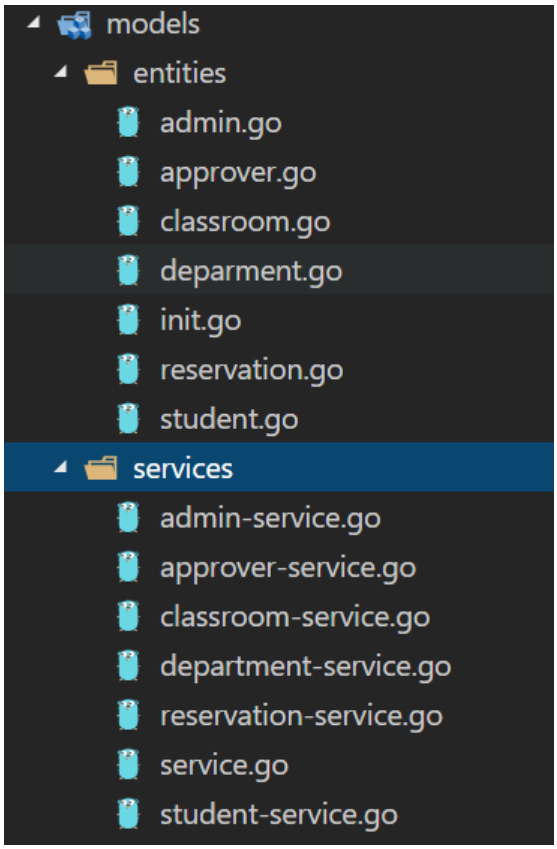
go 语言作为一种新型的语言，具有并发性能高、执行性能好、工具链完善、语言规范但组够简单灵活、易于上手等优势。go 语言作为后端语言已经有了很多成功的案例，并且已经形成了一些成熟 web 的框架，如 beego、gin 等。本系统没有使用这些成熟的框架，而是使用了一些成熟的组件来完成开发。

1) 使用 gorilla/mux 组件完成路由设计
2) 使用 bitly/go-simplejson 组件解析 http 请求数据
3) 使用 natefinch/lumberjack 组件完成日志模块功能
4) 使用 gorilla/sessions 组件完成 session 功能
5) 使用 go-xorm 组件完成数据实体的持久化，操作数据库

2. 模块划分

2.1 数据访问模块

数据访问层定义了 student、approver、admin、reservation、classroom、department 五个数据实体，并定义一些对数据库的操作。这些通过面向对象中"类"的概念实现。go 语言中，接口实现抽象类，struct 实现具体的类，包实现数据隐藏(包中的小写字母开头的变量、函数不能被外部访问)。



1) student 类

```go
package entities

/*
* StudentInfo store student information
 */
type StudentInfo struct {
  StudentId   int `xorm:"pk 'id'"`
  StudentPwd  string
  StudentName string
}
```

```go
package services

import (
  "github.com/kangbb/ccrsystem/models/entities"
)

type StudentInfoService struct{}

var StudentService = StudentInfoService{}

/*
* Create a new student information.
 */
func (*StudentInfoService) NewStudent(id int, pwd string, name string) *entities.StudentInfo {
  student := &entities.StudentInfo{
    StudentId:   id,
    StudentPwd:  pwd,
    StudentName: name,
  }
  return student
}

/*
* Insert student information to the db
 */
func (*StudentInfoService) SaveAInfo(std *entities.StudentInfo) error {
  _, err := entities.MasterEngine.InsertOne(std)

  return err
}
```

2) approver 类

```go
package entities

/*
* ApproverInfo store approver information
 */
type ApproverInfo struct {
  ApproverId   int `xorm:"pk 'id'"`
  ApproverPwd  string
  ApproverName string
  DepartmentId int
}
package services

import (
  "github.com/kangbb/ccrsystem/models/entities"
)

type ApproverInfoService struct{}

var ApproverService = ApproverInfoService{}

/*
* Create a new Approver
 */
func (*ApproverInfoService) NewApprover(id int, pwd string, name string, departmentId int) *entiti
  approver := &entities.ApproverInfo{
    ApproverId:   id,
    ApproverPwd:  pwd,
    ApproverName: name,
```

3) admin 类

```go
package entities

/*
* AdminInfo store admin information
*/
type AdminInfo struct {
  AdminId   int `xorm:"pk 'id'"`
  AdminPwd  string
  AdminName string
}
```

```go
package services

import (
  "github.com/kangbb/ccrsystem/models/entities"
)

type AdminInfoService struct{}

var AdminService = AdminInfoService{}

/*
* Create a new Admin
*/
func (*AdminInfoService) NewAdmin(id int, pwd string, name string) *entities.AdminInfo {
  admin := &entities.AdminInfo{
    AdminId:   id,
    AdminPwd:  pwd,
    AdminName: name,
  }
  return admin
}
```

4) reservation 类

```go
package entities

import (
  "time"
)

/*
* ReservationInfo store reservation information
*/
type ReservationInfo struct {
  ResId            int `xorm:"autoincr pk 'id'"`
  ResReason        string
  StartTime        time.Time `xorm:"unique(reservation)"`
  EndTime          time.Time `xorm:"unique(reservation)"`
  ClassroomId      int       `xorm:"unique(reservation)"`
  StudentId        int
  OrganizationName string
  ApproverId       int
  ResState         int // 0, 未审批; 1, 审批中; 2, 审批通过; 3, 审批未通过
  ApprovalNote     string
}
```

```go
package services

import (
  "time"

  "github.com/kangbb/ccrsystem/models/entities"
)

type ReservationInfoService struct{}

var ReservationService = ReservationInfoService{}

func (*ReservationInfoService) NewReservation(reason string, start time.Time, end time.Time, class
  studentId int, organizationName string, approverId int, note string, state int) *entities.Reserv
  reservation := &entities.ReservationInfo{
    ResReason:        reason,
    StartTime:        start,
    EndTime:          end,
    ClassroomId:      classroomId,
    StudentId:        studentId,
    OrganizationName: organizationName,
    ApproverId:       approverId,
    ApprovalNote:     note,
    ResState:         state,
  }
  return reservation
}
```

5) classroom 类

```go
package entities

/*
* ClassroomInfo store approver information
 */
type ClassroomInfo struct {
  ClassroomId       int    `xorm:"pk autoincr 'id'"`
  ClassroomCampus   string `xorm:"unique(classroom)"`
  ClassroomBuilding string `xorm:"unique(classroom)"`
  ClassroomNum      string `xorm:"unique(classroom)"`
  Capacity          int
}
package services

import (
  "github.com/kangbb/ccrsystem/models/entities"
)

type ClassroomInfoService struct{}

var ClassroomService = ClassroomInfoService{}

//Create a new Classroom
func (*ClassroomInfoService) NewClassroom(campuse string, building string, num string, cap int) *e
  classroom := &entities.ClassroomInfo{
    ClassroomCampus:   campuse,
    ClassroomBuilding: building,
    ClassroomNum:      num,
    Capacity:          cap,
  }
  return classroom
}
```

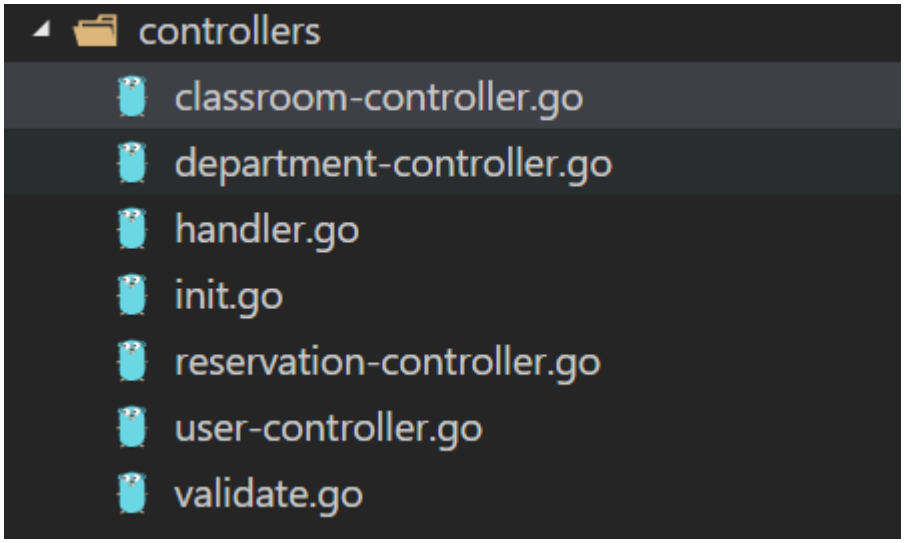
6) department 类

```go
package entities

/*
* DepartmentInfo store approval department information
 */
type DepartmentInfo struct {
  DepartmentId    int    `xorm:"'id' autoincr pk"`
  DepartmentName  string `xorm:"unique"`
  Introduction    string
  DepartmentOrder int //不能命名为order,它是mysql的关键字
  Note            string
  //Note, note whether it is Initial approval department or final approval department
  //value: initial, middle, final
}
```

```go
package services

import (
  "github.com/kangbb/ccrsystem/models/entities"
)

type DepartmentInfoService struct{}

var DepartmentService = DepartmentInfoService{}

/*
* Create a new deparment
 */
func (*DepartmentInfoService) NewDeparment(name string, introduction string, order int, note strin
  department := &entities.DepartmentInfo{
    DepartmentName:  name,
    Introduction:    introduction,
    DepartmentOrder: order,
    Note:            note,
  }
  return department
}
```

2.1 业务逻辑模块

这部分主要处理前端的业务请求。由于用户信息业务请求类似，所以将用户信息相关的业务并入一个类；classroom、reservation、department 相关的业务请求分别作为一个类。handler 则负责验证用户权限并分发前端的业务到具体的函数。其位置如下：



1) Handler 类

```go
package controllers

import (
  "encoding/json"
  "net/http"
  "regexp"

  "github.com/kangbb/ccrsystem/logs"
)

//Render file
func GetIndex(w http.ResponseWriter, r *http.Request) {
  renderFile("index", w, r)
}
func GetStudentIndex(w http.ResponseWriter, r *http.Request) {
  if !validatePerm([]string{"Student"}, false, w, r) {
    return
  }
  renderFile("studentIndex", w, r)
}
```

2) User 类

```go
package controllers

import (
  "encoding/json"
  "fmt"
  "html/template"
  "math/rand"
  "net/http"
  "strconv"

  simplejson "github.com/bitly/go-simplejson"
  "github.com/kangbb/ccrsystem/logs"
  "github.com/kangbb/ccrsystem/middlewares"
  "github.com/kangbb/ccrsystem/models/services"
)

/***************************** Signin and Signout ***********************/
/*
* User Signin
* For student, approver, admin. If sucess, redirect to the index which belong to themselves
* If failed, redirect to the home page.
 */
func signin(userType string, w http.ResponseWriter, r *http.Request) {
  var (
    id   int
    name string
    pwd  string
  )
```

3) reservation 类

```go
package controllers

import (
  "crypto/md5"
  "encoding/json"
  "fmt"
  "io"
  "log"
  "math/rand"
  "net/http"
 "regexp"
 "strconv"
 "time"

 simplejson "github.com/bitly/go-simplejson"
 "github.com/gorilla/mux"
 "github.com/kangbb/ccrsystem/logs"
 "github.com/kangbb/ccrsystem/models/services"
)

type Reservation struct {
 ResId            int
 ResReason        string
 StartTime        string
 EndTime          string
 StudentId        int
 ClassroomId      int
 ClassroomNum     string
 Capacity         int
 OrganizationName string
 ApproverId       int
 ResState         int
 ApprovalNote     string
}

/************************ Data Interface For Reservation ***********************/
/*
* Get reservation by id
 */
func getResById(w http.ResponseWriter, r *http.Request) {
 //default use the parameter from r.body
 id := getIdFromPath(r)
 res, err := services.ReservationService.FindInfoById(id)
 if logs.SqlError(err, w, res.ApproverId != 0) {
  return
 }
```

4) classroom 类

```go
package controllers

import (
  "encoding/json"
  "net/http"
  "strconv"

  simplejson "github.com/bitly/go-simplejson"
  "github.com/kangbb/ccrsystem/logs"
  "github.com/kangbb/ccrsystem/models/services"
)

/************************ Data Interface For Classroom ***********************/
```

```go
/*
* Get classroomlist
* Accessed for all user, which are student, approver, and admin
 */
func getClassroomList(w http.ResponseWriter, r *http.Request) {

  classrooms, err := services.ClassroomService.FindAllInfo()
  if logs.SqlError(err, w, len(classrooms) != 0) {
    return
  }

  data, _ := json.Marshal(classrooms)
  w.WriteHeader(200)
  w.Write(data)
}
```

5) department 类

```go
package controllers

import (
  "encoding/json"
  "math/rand"
  "net/http"

  simplejson "github.com/bitly/go-simplejson"
  "github.com/kangbb/ccrsystem/logs"
  "github.com/kangbb/ccrsystem/models/services"
)

/************************ Data Interface For Department **********************/
/*
* Get department list
 */
func getDepartmentList(w http.ResponseWriter, r *http.Request) {

  departments, err := services.DepartmentService.FindAllInfo()
  if logs.SqlError(err, w, len(departments) != 0) {
    return
  }

  data, _ := json.Marshal(departments)

  w.WriteHeader(200)
  w.Write(data)
}
```

2.3 日志模块

该部分主要负责处理程序错误，并将错误时间、错误信息内容写入文件。

主要函数如下：

```go
/*
* A simple logs management.
* It will process all of the errors about the system and write them to file.
 */
package logs

import (
  "encoding/json"
```

```
  "errors"
  "fmt"
  "log"
  "net/http"
  "regexp"
  "runtime"
  "strings"
  "time"

  "github.com/natefinch/lumberjack"
)

var Log = &log.Logger{}
var SWITCH_BRANCH_ERROR = errors.New("The branch shouldn't appear.")
var PERMISSION_DENY = errors.New("Permission deny.")

type ErrorMsg struct {
  Msg string
}

/*
* Excute when the system initial
* It will create a log file to store the error logs.
 */
func init() {
  // Create a new file to store the log.
  f := &lumberjack.Logger{
    Filename:   "./data/logs/ccrsystemlog/" + time.Now().Format("2006-01-02") + "-ccrsystem.log",
    MaxSize:    500, // megabytes
    MaxBackups: 3,
    MaxAge:     28,   //days
    Compress:   true, // disabled by default
  }

  // New a logger object.
  Log = log.New(f, "[Error]", log.Ldate|log.Ltime|log.Lmicroseconds)
}
```

2.4 路由模块

该模块主要负责分发前端的请求到相应的业务模块，从而进行处理。其位置如下：

其部分函数如下：

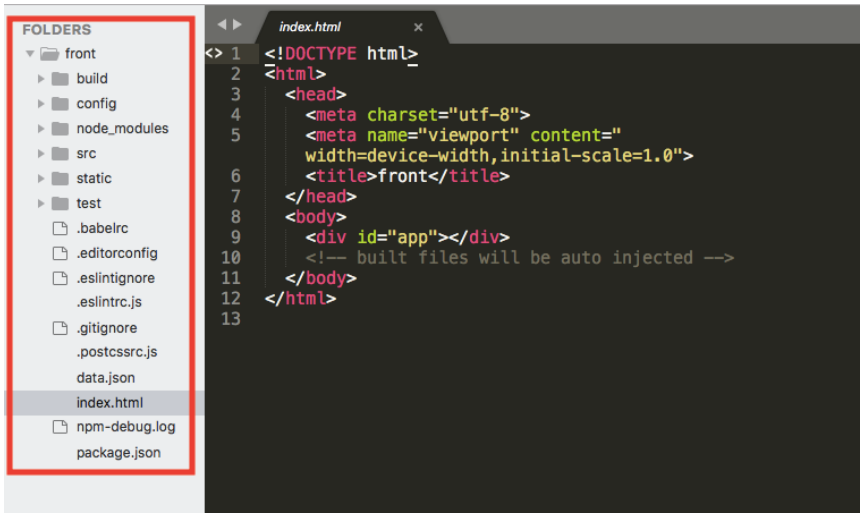
```
package routers

import (
  "net/http"
  "os"

  "github.com/gorilla/mux"
  "github.com/kangbb/ccrsystem/controllers"
  "github.com/kangbb/ccrsystem/middlewares"
)

/*
* Get routers. The function return a router list to the main.go
* It registes some routers which used to process business request.
 */
func GetRouters() *mux.Router {
```

二、 前端

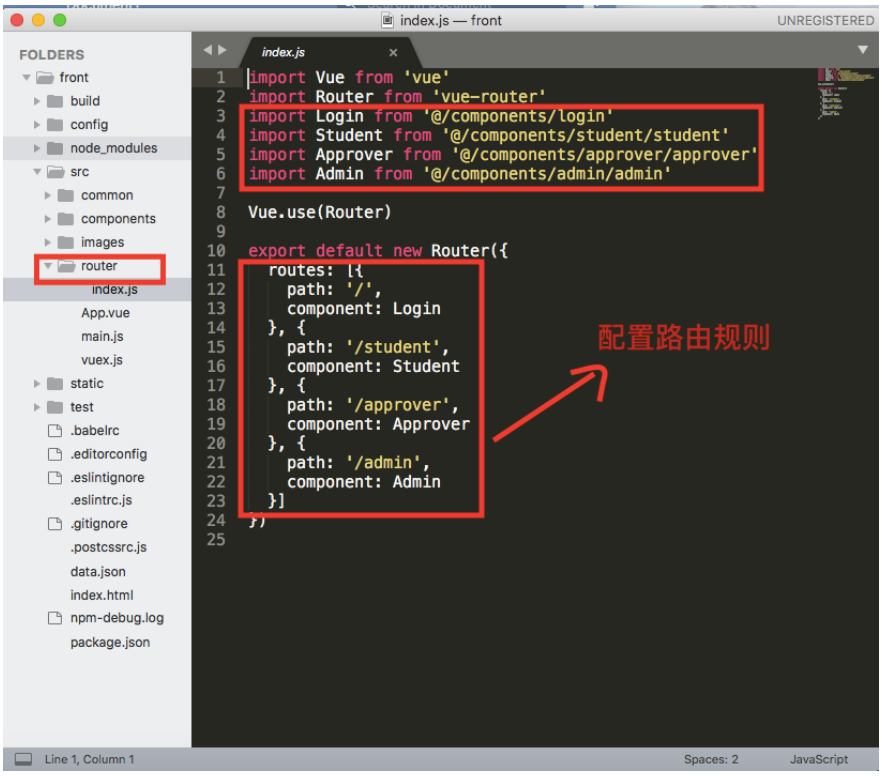
前端采用 Vue 框架开发, 它是一个构建数据驱动的 web 界面的渐进式框架。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。核心是一个响应的数据绑定系统。

1. 具体技术介绍

1) 使用 vue-cli 搭建项目, 同时也方便对项目的大包和维护.搭建好后生成如下目录:



2) 使用 vue-router 配置路由规则、实现路由跳转页面展示



```
this.$http.post(apiStr, obj).then(res => {
  if (res.status === 200) {
    setCookie(pId, this.id, 1000 * 60)
    this.$router.push('/' + param)
  }
}).catch(err => {
  console.log(err)
  this.error = '密码错误'
})
```

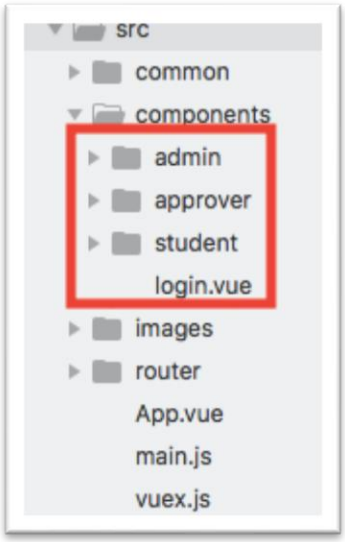
3) 使用 vue-resource 进行网络请求

```
main.js
1  // The Vue build version to load with the `import` command
2  // (runtime-only or standalone) has been set in webpack.ba
3  import Vue from 'vue'
4  // import store from './vuex.js'
5  import App from './App'
6  import router from './router'
7  import VueResource from 'vue-resource'
8
9  Vue.use(VueResource)
10 Vue.config.productionTip = false
11 Vue.prototype.HOST = '/api'
12
13 /* eslint-disable no-new */
14 new Vue({
15   el: '#app',
16   router,
17   components: { App },
18   template: '<App/>'
19 })
20
```

```
let apiStr = this.HOST + '/' + param + '/signin'
this.$http.post(apiStr, obj).then(res => {
  if (res.status === 200) {
    setCookie(pId, this.id, 1000 * 60)
    this.$router.push('/' + param)
  }
}).catch(err => {
  console.log(err)
  this.error = '密码错误'
})
}
```

2. 模块划分

1) 根据子系统模块划分, 可以划分为登录模块、学生模块(教室查询审批)、审批员模块(审批)、管理员模块(信息管理).



2) 登录模块

(1) 主要函数

```
login (param) {
  // 格式只判断是否为空,不校验其他
  if (this.id === '' || this.password === '') {
    this.error = '账号或密码不能为空！'
  } else {
    let obj = {}
    let pId = param.substring(0, 1).toUpperCase() + param.substring(1) + 'Id'
    let pPwd = param.substring(0, 1).toUpperCase() + param.substring(1) + 'Pwd'
    obj[pId] = parseInt(this.id)
    obj[pPwd] = this.password

    let apiStr = this.HOST + '/' + param + '/signin'
    this.$http.post(apiStr, obj).then(res => {
      if (res.status === 200) {
        setCookie(pId, this.id, 1000 * 60)
        this.$router.push('/' + param)
      }
    }).catch(err => {
      console.log(err)
      this.error = '密码错误'
    })
  }
},
```

(2)登陆页面

3) 学生模块

(1) 主要函数

[查询教室]

```
formSubmit (e) {
  this.showError = false
  e.preventDefault()
  if (this.date != '' && this.timeBegin != '' && this.timeEnd != '' && this.capacity != '') {
    if (this.date < this.currentDate()) {
      alert('日期选择错误！')
    } else if (this.compareTime(this.timeBegin) > this.compareTime(this.timeEnd)) {
      alert('时间选择错误！')
    } else {
      var beginTime = this.getDateTime(this.date, this.timeBegin)
      var endTime = this.getDateTime(this.date, this.timeEnd)
      var apiStr = this.HOST + '/api/classrooms/state'
      var cap = parseInt(this.capacity)

      this.$http.get(apiStr, {params: {ClassroomCampus: '东校园', ClassroomBuilding: '公教楼',
        StartTime: beginTime, EndTime: endTime, Capacity: cap}}).then(res => {
        console.log(res)
        res = res.body
        this.chooseList = res
        this.showList = true
      }).catch(err => {
        console.log(err)
        this.showError = true
      })
    }
  } else {
    alert('请选择日期、时间、容量！')
  }
```

[提交教室申请]

```
applySubmit (e) {
  if (document.getElementById('organization').value == ' ' || document.getElementById('
    reservationInfo').value == '') {
    alert('请填写使用方隶属组织和申请教室用途！')
  } else {
    var apiStr = this.HOST + '/api/users/student/reservations'
    var obj = {}
    this.info.ResReason = this.reservationInfo
    this.info.OrganizationName = this.organization
    obj.StartTime = this.getDateTime(this.date, this.timeBegin)
    obj.EndTime = this.getDateTime(this.date, this.timeEnd)
    obj.ResReason = this.reservationInfo
    obj.ClassroomId = parseInt(this.info.ClassroomId)
    obj.OrganizationName = this.organization

    this.$http.post(apiStr, obj).then(res => {
      alert('submission succeeds!')
      this.chooseList.splice(this.info.indexer, 1)
      this.closeWindow()
      if (this.chooseList.length == 0) {
        this.showList = false
        this.showError = true
      }
    }).catch(err => {
      console.log(err)
      alert('submission fails!')
      this.closeWindow()
    })
  }
  e.preventDefault()
},
```

(2)学生页面

4) 审批模块

(1)主要函数

[通过审批、拒绝审批]

```
passApprove () {
  console.log(this.info.resId)
  let apistr = this.HOST + '/api/reservations/' + this.info.ResId
  this.$http.put(apistr, {ResState: '3', ApprovelNote: '通过'}).then(res => {
    console.log(res)
  }).catch(err => {
    console.log(err)
  })
  for (var item in this.waitApproveList) {
    var temp = this.waitApproveList[item]
    if (temp.ResId == this.info.ResId) {
      this.waitApproveList.splice(this.waitApproveList.indexOf(temp), 1)
    }
  }
  this.showDetail = false
},
failApprove () {
  let apistr = this.HOST + '/api/reservations/' + this.info.ResId
  this.$http.put(apistr, {ResState: '4', ApprovelNote: '活动不符合要求,不通过'}).then(res => {
    console.log(res)
  }).catch(err => {
    console.log(err)
  })
  for (var item in this.waitApproveList) {
    var temp = this.waitApproveList[item]
    if (temp.ResId == this.info.ResId) {
      this.waitApproveList.splice(this.waitApproveList.indexOf(temp), 1)
    }
  }
  this.showDetail = false
},
```

(2)审批页面

5) 管理员模块

(1)主要函数

[增加教室]

```
addClassSubmit () {
  if (this.addClassNum != '' && this.addCapacity != '') {
    if (confirm('确认增加')) {
      var apiStr = '/api/api/classrooms'
      var obj = {}
      obj.ClassroomCampus = '东校园'
      obj.ClassroomBuilding = '公教楼'
      obj.ClassroomNum = this.addClassNum
      obj.Capacity = parseInt(this.addCapacity)

      this.$http.post(apiStr, obj).then(res => {
        alert('成功添加！')
        this.tempClass.push(obj)
        this.closeClassSubmit()
      }).catch(err => {
        console.log(err)
        alert('添加失败！')
      })
    }
    this.addClassNum = ''
    this.addCapacity = ''
  } else {
    alert('请输入教室号和容量！')
  }
}
```

[删除教室]

```
classDeleteSubmit () {
  if (confirm('确认删除')) {
    var idx = parseInt(this.classId)
    var apiStr = '/api/api/classrooms/' + idx
    this.$http.delete(apiStr, {body: {Id: idx}}).then(res => {
      //      console.log(res)
      this.tempClass.splice(parseInt(this.classIndexer), 1)
      alert('删除成功！')
    }).catch(err => {
      console.log(err)
      alert('删除失败！')
    })
    this.classReviseCancel()
  }
},
```

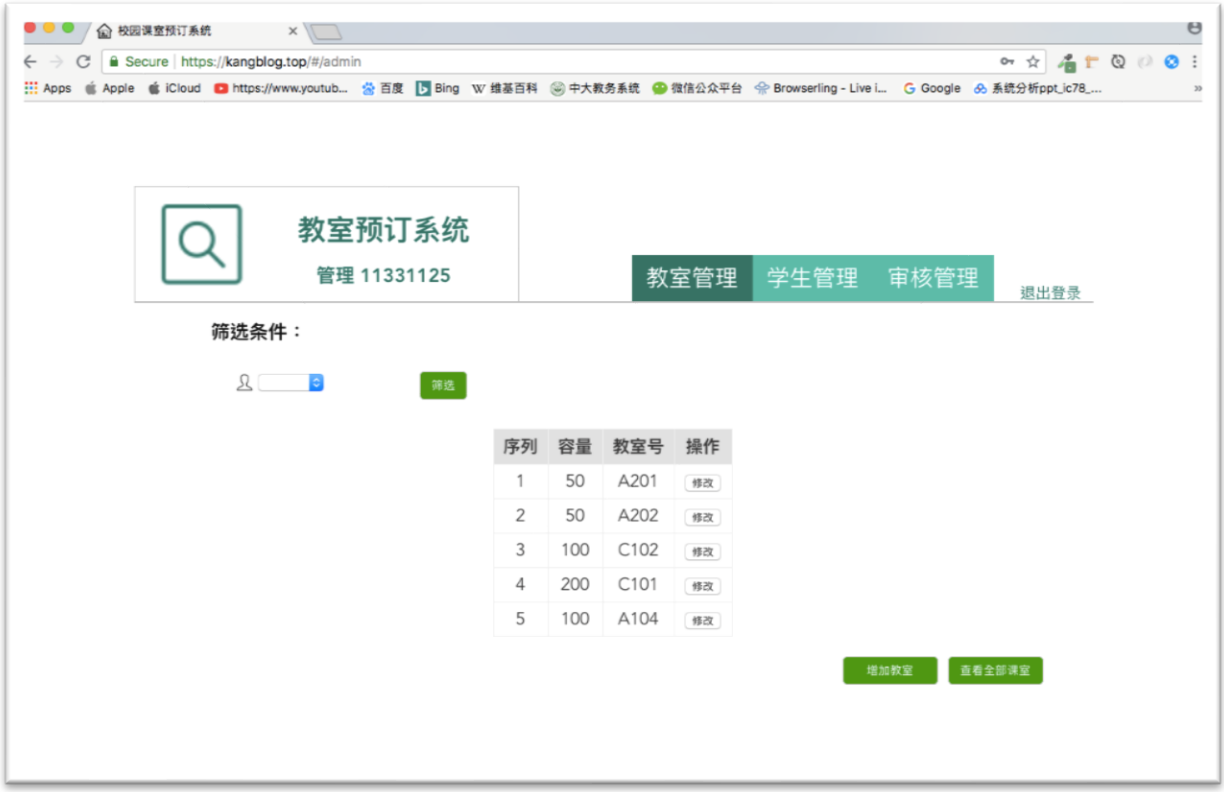
[查询教室]

```
displayAllClass () {
    var apiStr = '/api/api/classrooms'
    this.$http.get(apiStr).then(res => {
      res = res.body
      this.fullClass = res
      this.tempClass = this.fullClass
      this.showClass = true
    }).catch(err => {
      console.log(err)
      this.showError = true
    })
  },
```

[修改教室]

```
classReviseSubmit () {
  if (this.classCapacity != '') {
    if (confirm('确认修改')) {
      var cap = parseInt(this.classCapacity)
      var apiStr = '/api/api/classrooms/' + this.classId
      this.$http.put(apiStr, {ClassroomCampus: '东校园', ClassroomBuilding: '公教楼', Capacity: c
        , ClassroomNum: this.classNum}).then(res => {
        //  console.log(res)
        alert('revision succeeds!')
      }).catch(err => {
        console.log(err)
        alert('revision fails!')
      })
    }
    this.classReviseCancel()
  } else {
    alert('请输入容量！')
  }
},
```

(2)管理员页面


校园课室预订系统
Secure | https://kangblog.top/#/admin
Apps Apple iCloud https://www.youtub... 百度 Bing 维基百科 中大教务系统 微信公众平台 Browserling - Live i... Google 系统分析ppt_ic78_...
教室预订系统
管理 11331125
教室管理 学生管理 审核管理
退出登录
筛选条件：
筛选
序列 容量 教室号 操作
1 50 A201 修改
2 50 A202 修改
3 100 C102 修改
4 200 C101 修改
5 100 A104 修改
增加教室
查看全部课室