

目录

1 引言	3
2 系统概述	3
2.1 背景	3
2.2 项目目标	3
2.3 项目范围	3
3 系统架构	4
3.1 用例图	4
3.2 时序图	4
3.3 活动图	9
3.4 状态图	15
3.5 类图	17
4 模块设计	19
4.1 数据库设计	19
4.1.1 数据库 er 图	19
4.1.2 数据库字段表	20
4.2 接口设计	22
4.2.1 用户相关接口	22
接口清单	22
4.2.2 管理员相关接口	29
4.3 界面设计	32
5 部署维护	45

6 变更管理和版本控制	46
7 安全设计	47
7.1 安全策略	47
7.2 安全机制	47
8 性能设计	48
8.1 性能设计考虑	48
8.2 性能设计策略	48
8.3 性能测试与优化	48
9 测试策略	49
9.1 测试用例	49
9.2 测试计划	50

1 引言

本报告的主要作用是确定各个项目模块的开发情况和主要的负责人，供各项目模块的负责人阅读，做到及时协调，按步有序进行项目的开发。减少开发中的不必要的损失，便于项目团队成员更好地了解项目情况，使项目工作开展的各个过程合理有序，因此以文件化的形式，把对于在项目生命周期内的工作任务范围、各项工作的任务分解、项目团队组织结构、各团队成员的工作责任、团队内外沟通协作方式、开发进度、经费预算、项目内外环境条件等内容做出的安排以书面的方式，作为项目团队成员以及项目干系人之间的共识与约定，项目生命周期内的所有项目活动的行动基础，项目团队开展和检查项目工作的依据。

2 系统概述

2.1 背景

随着网络技术的发展，计算机应用的日益普及和深化，各种软件，小程序使我们的生活更加方便快捷。为了满足学生们预约使用研讨室的使用需求，我们设计了一款在线研讨室预约小程序。该小程序能随时查看校园内的研讨室使用情况，并且按需预约，将极大的方便学生的学习。

2.2 项目目标

项目的总目标可按以下三个阶段目标来进行：

- （1）第一阶段目标:实现研讨室预约小程序的基本功能，小组各个成员进行各个模块的开发，形成初步的系统。
- （2）第二阶段目标:攻克技术上的难题，实现研讨室预约小程序的一些特殊功能，进一步完善系统。
- （3）第三阶段目标:让系统投入到实际运用中，做好系统的维护工作。

2.3 项目范围

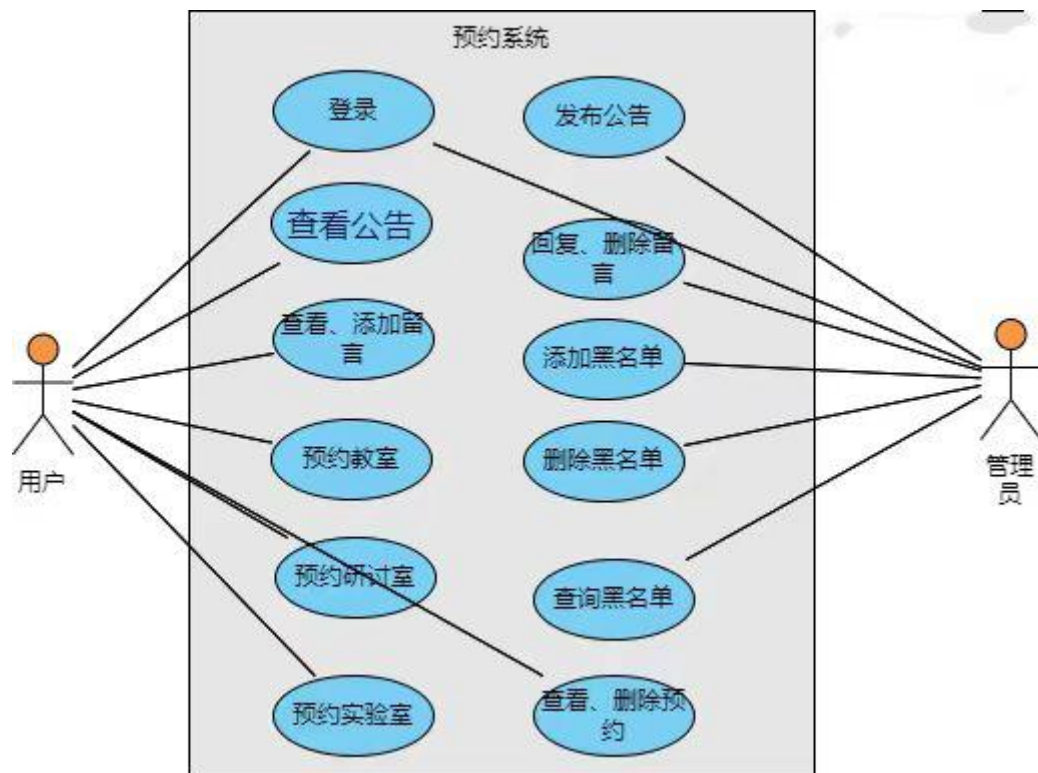
- （1）能预约研讨室
- （2）能查看研讨室的详尽情况

(3) 能查看过往预约记录

(4) 能查看公告

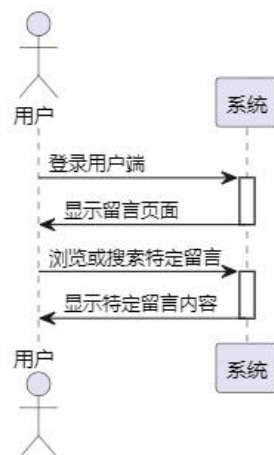
3 系统架构

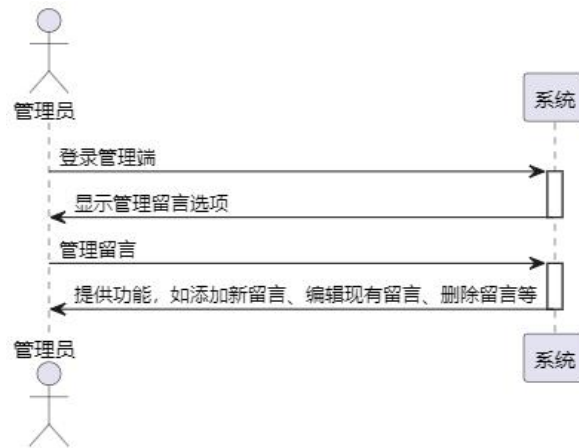
3.1 用例图



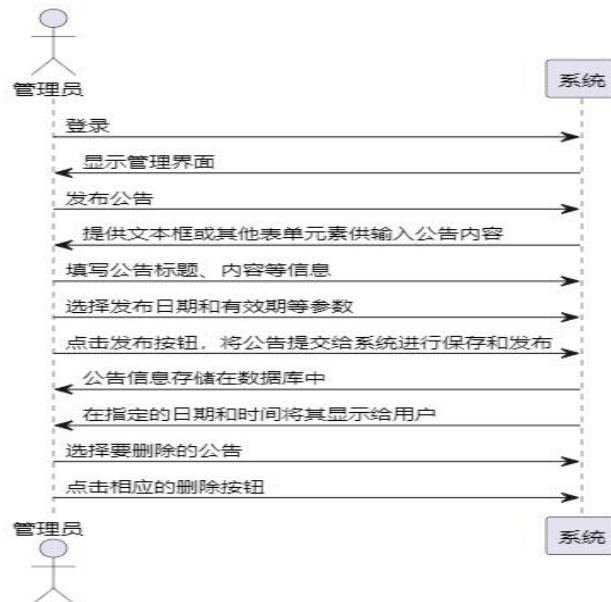
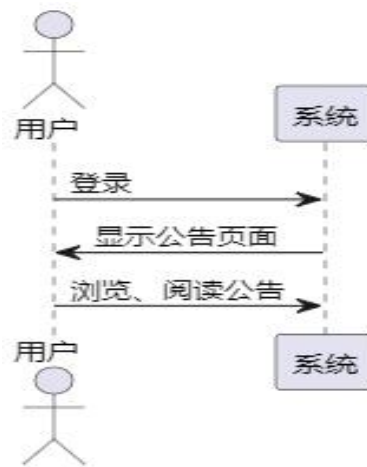
3.2 时序图

(1) 留言:

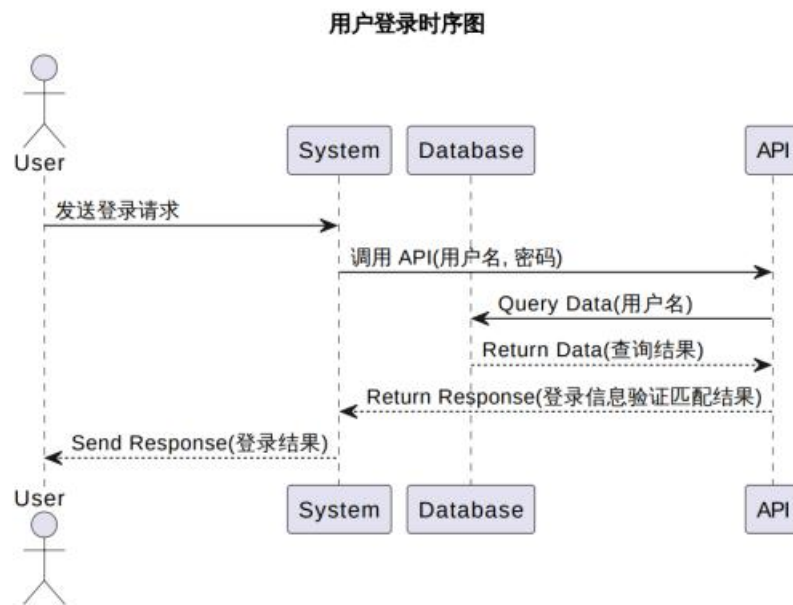




(2) 公告:

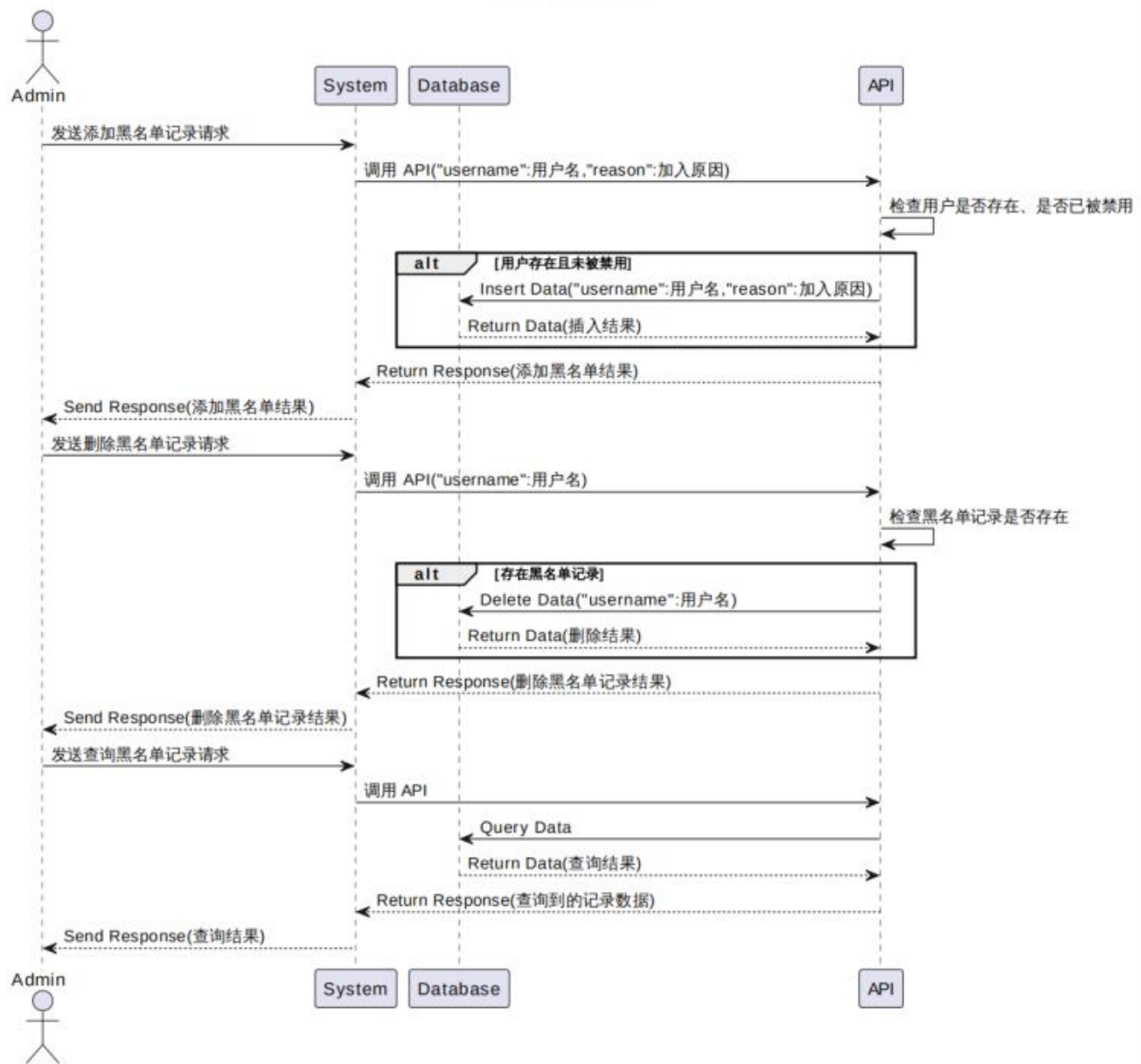


(3) 登录



(4) 黑名单管理

黑名单管理时序图

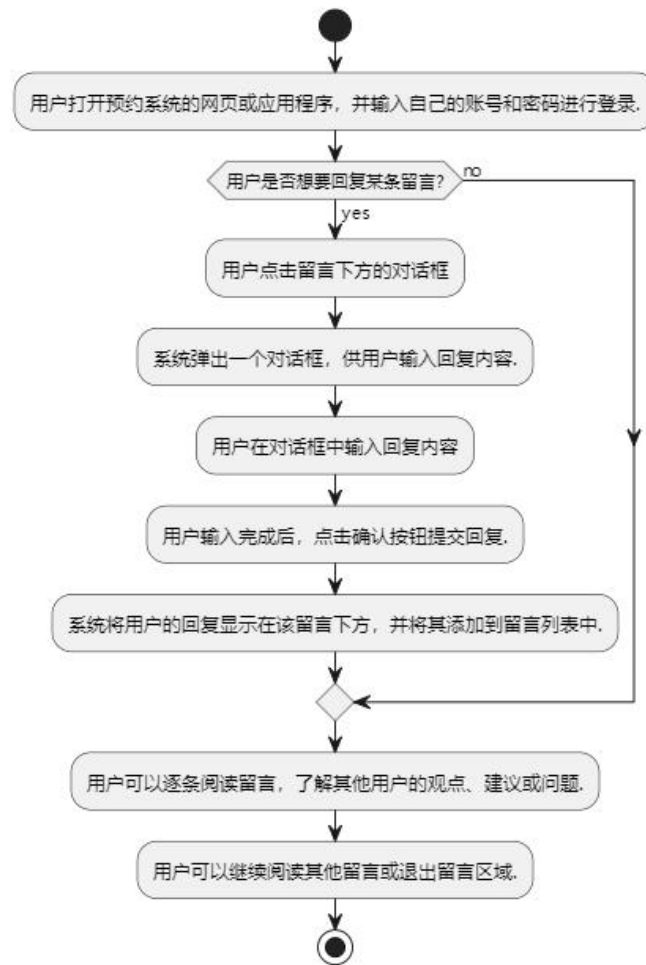


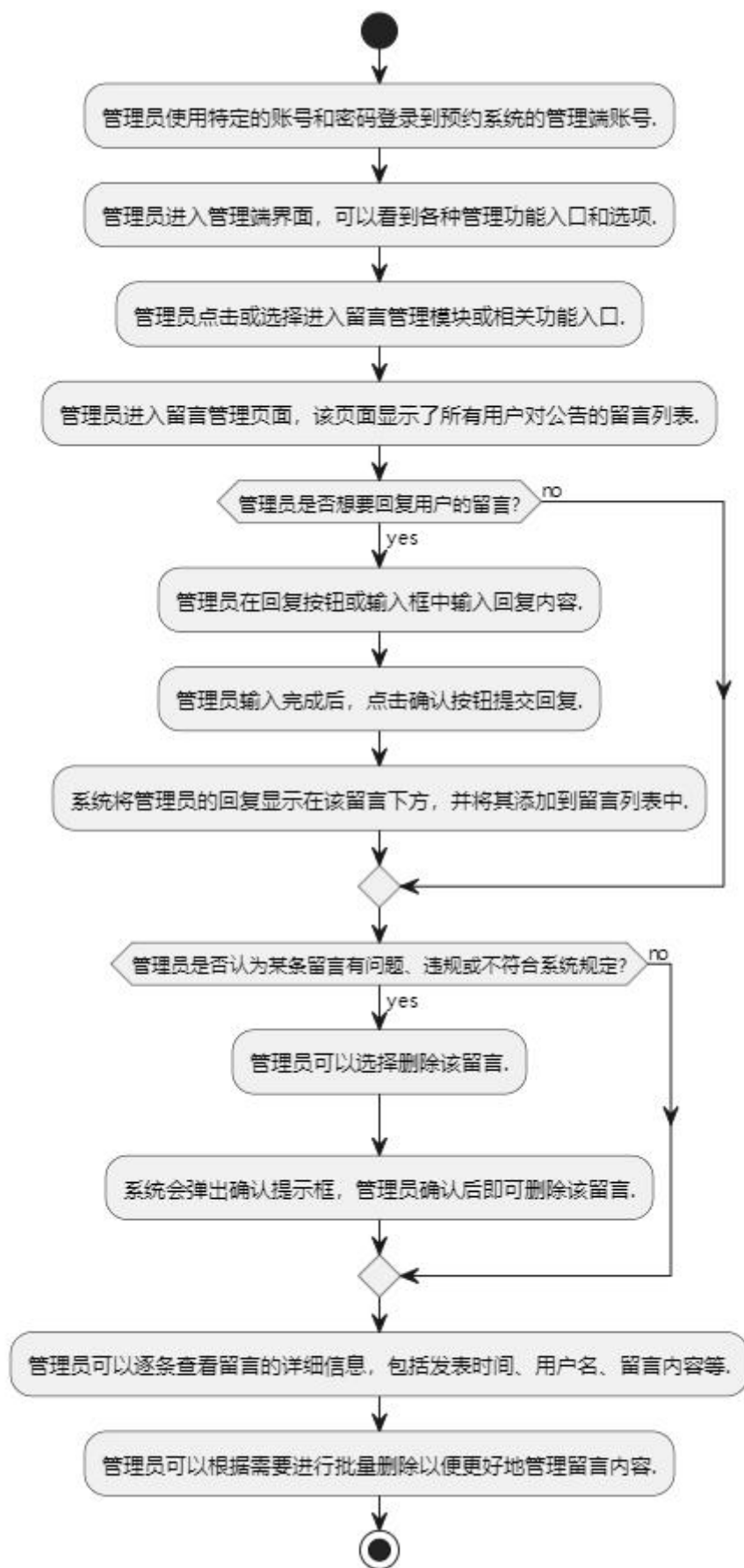
(5) 预约



3.3 活动图

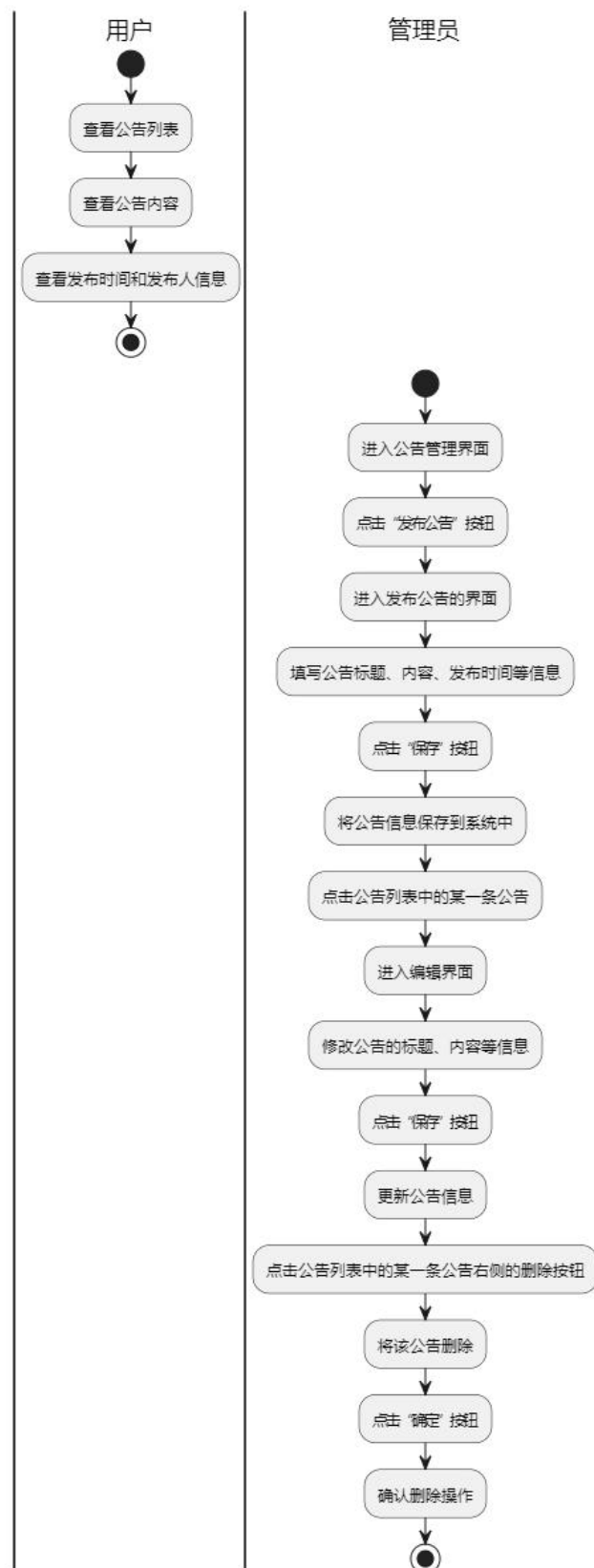
(1) 留言



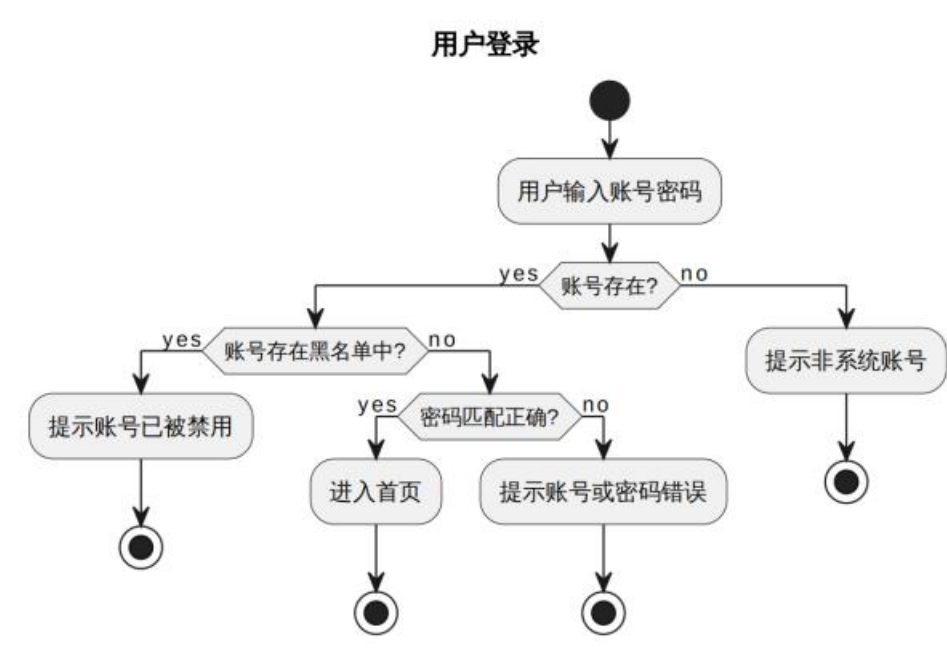


(2) 公告

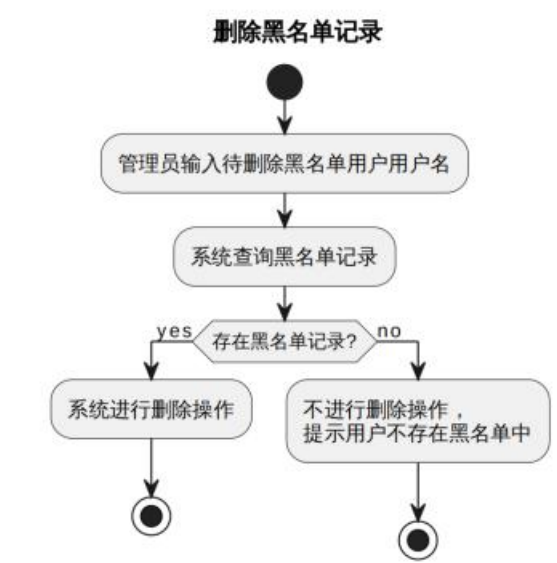
公告系统活动图



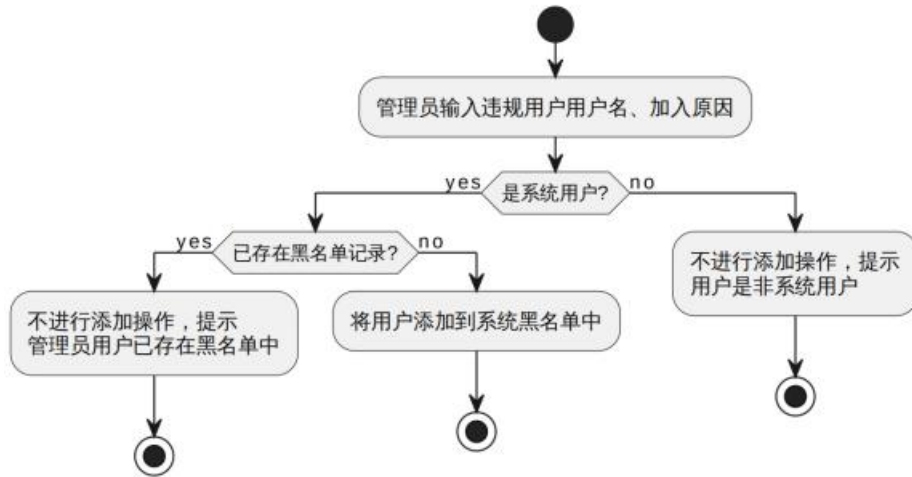
(3) 登录



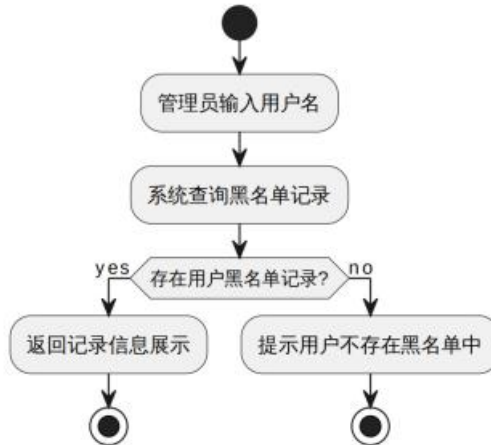
(4) 黑名单



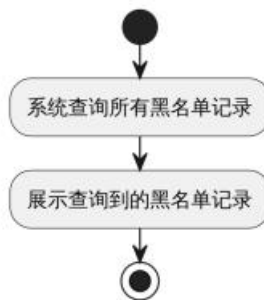
添加黑名单记录



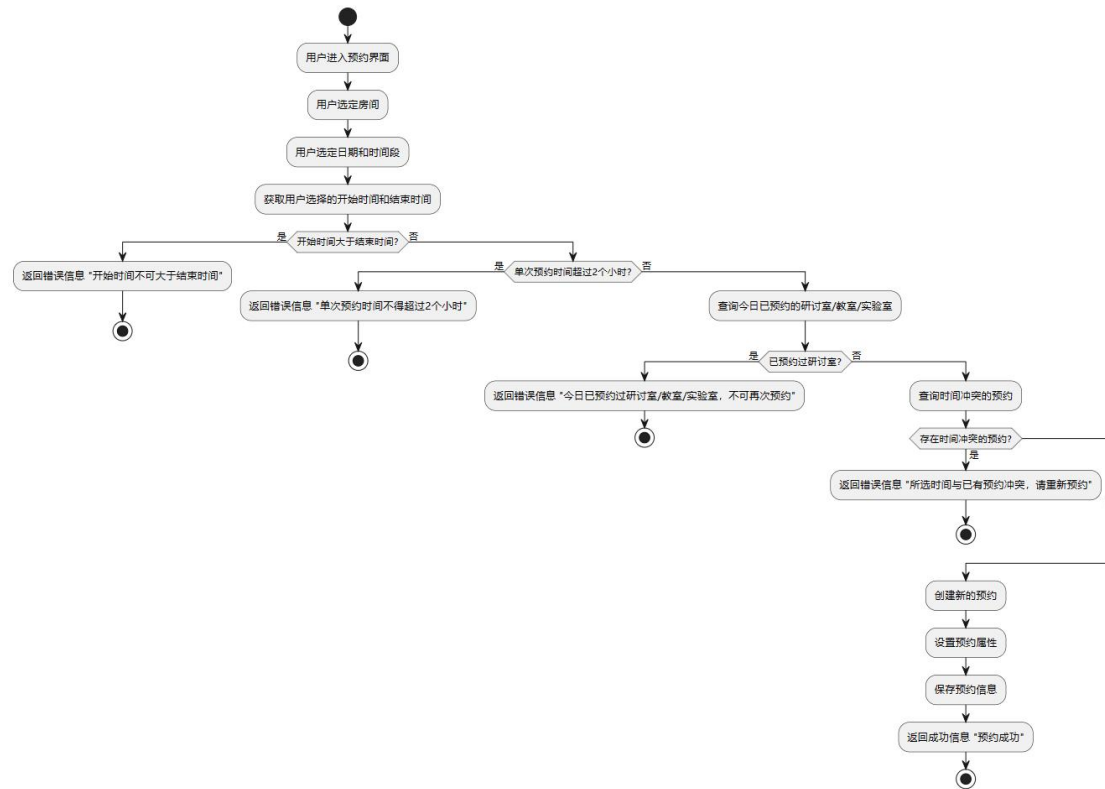
查询用户黑名单记录



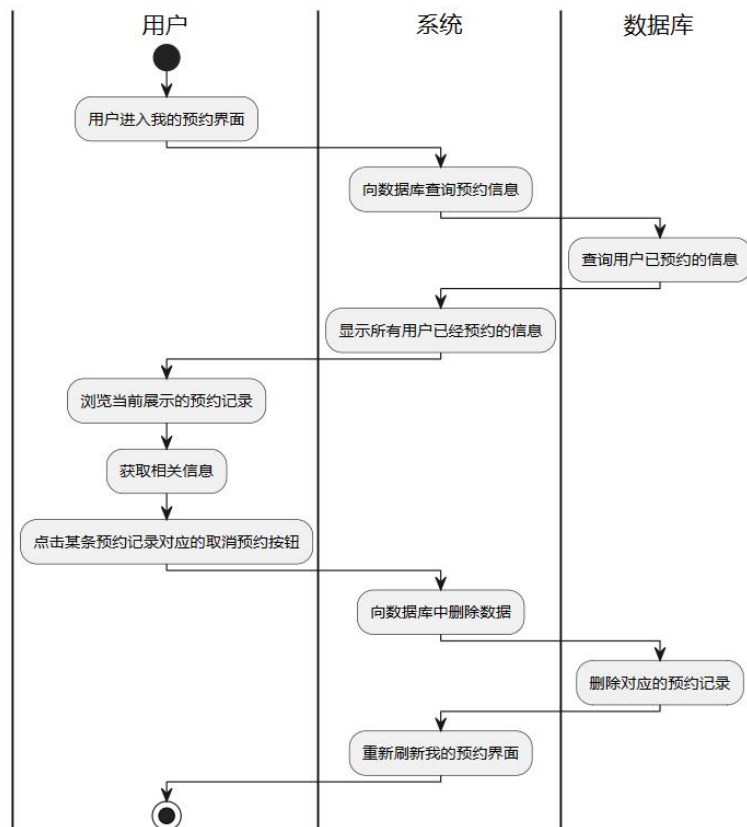
查看所有黑名单记录



(5) 用户预约研讨室/教室/实验室：

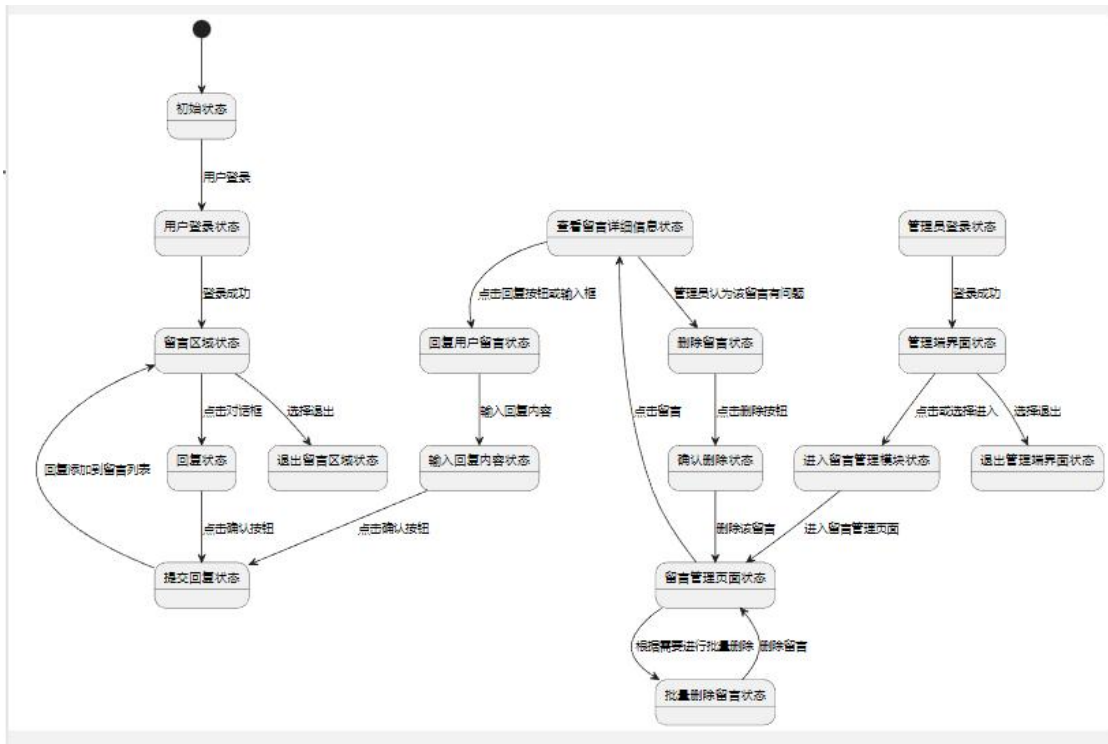


(6) 查看我的所有预约和取消预约：

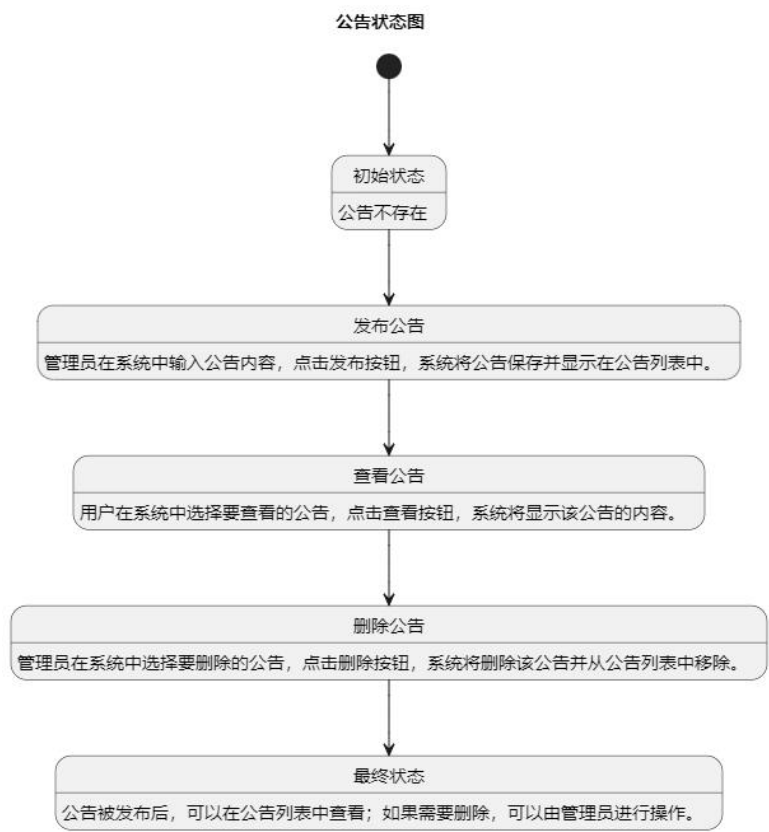


3.4 状态图

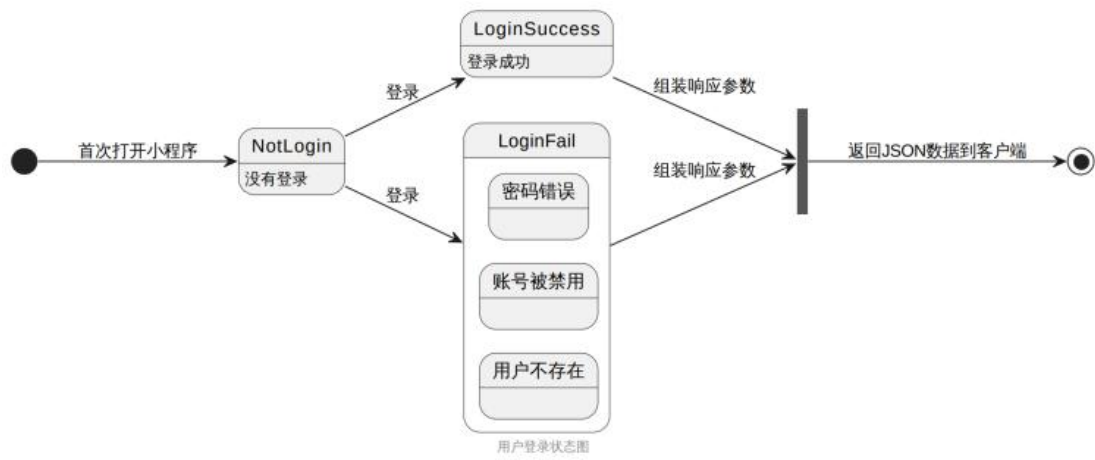
(1) 留言



(2) 公告

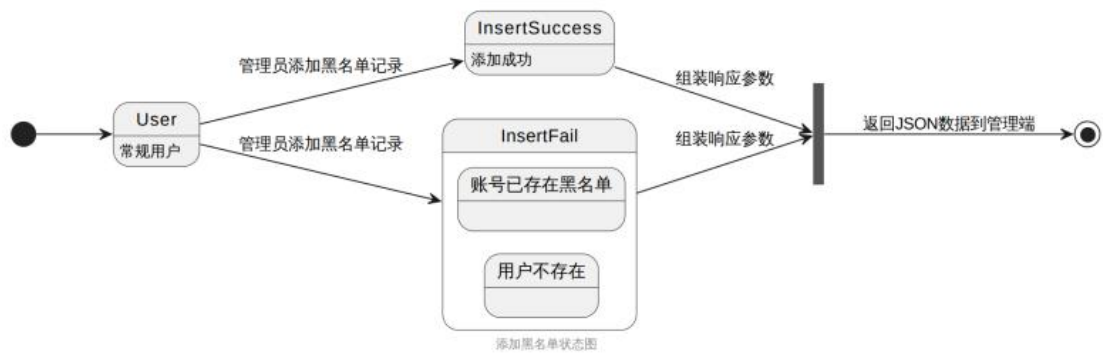


(3) 登录



(4) 黑名单管理

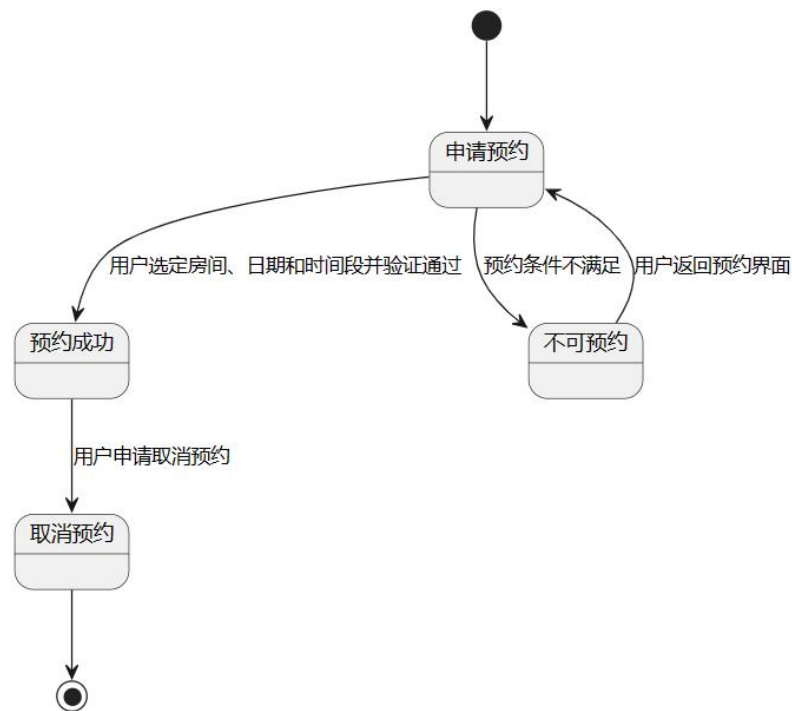
添加



删除

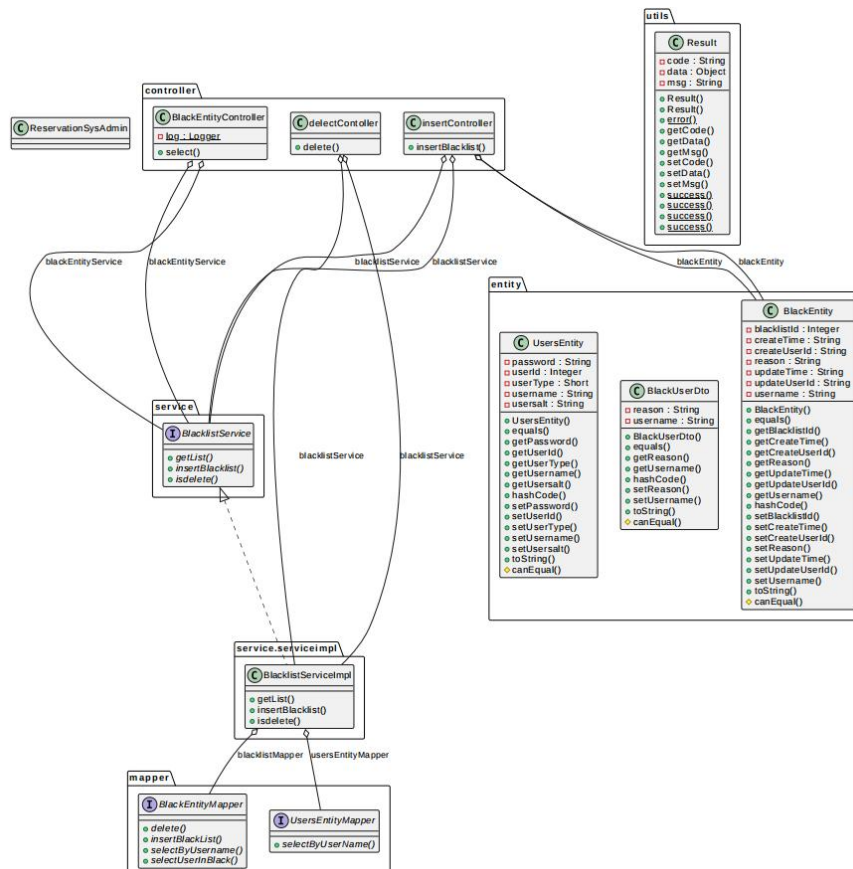


(5) 预约

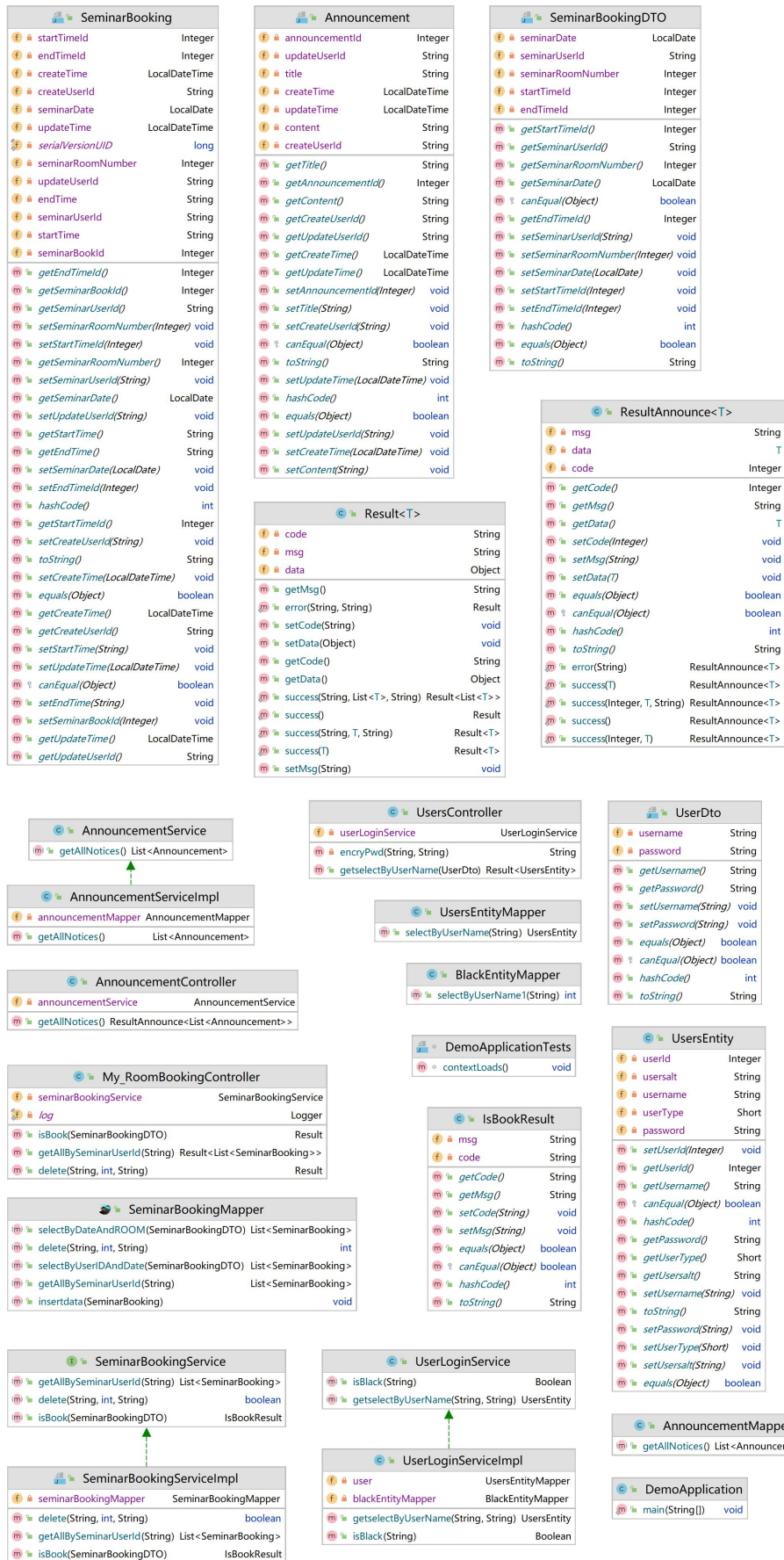


3.5 类图

黑名单管理类结构图



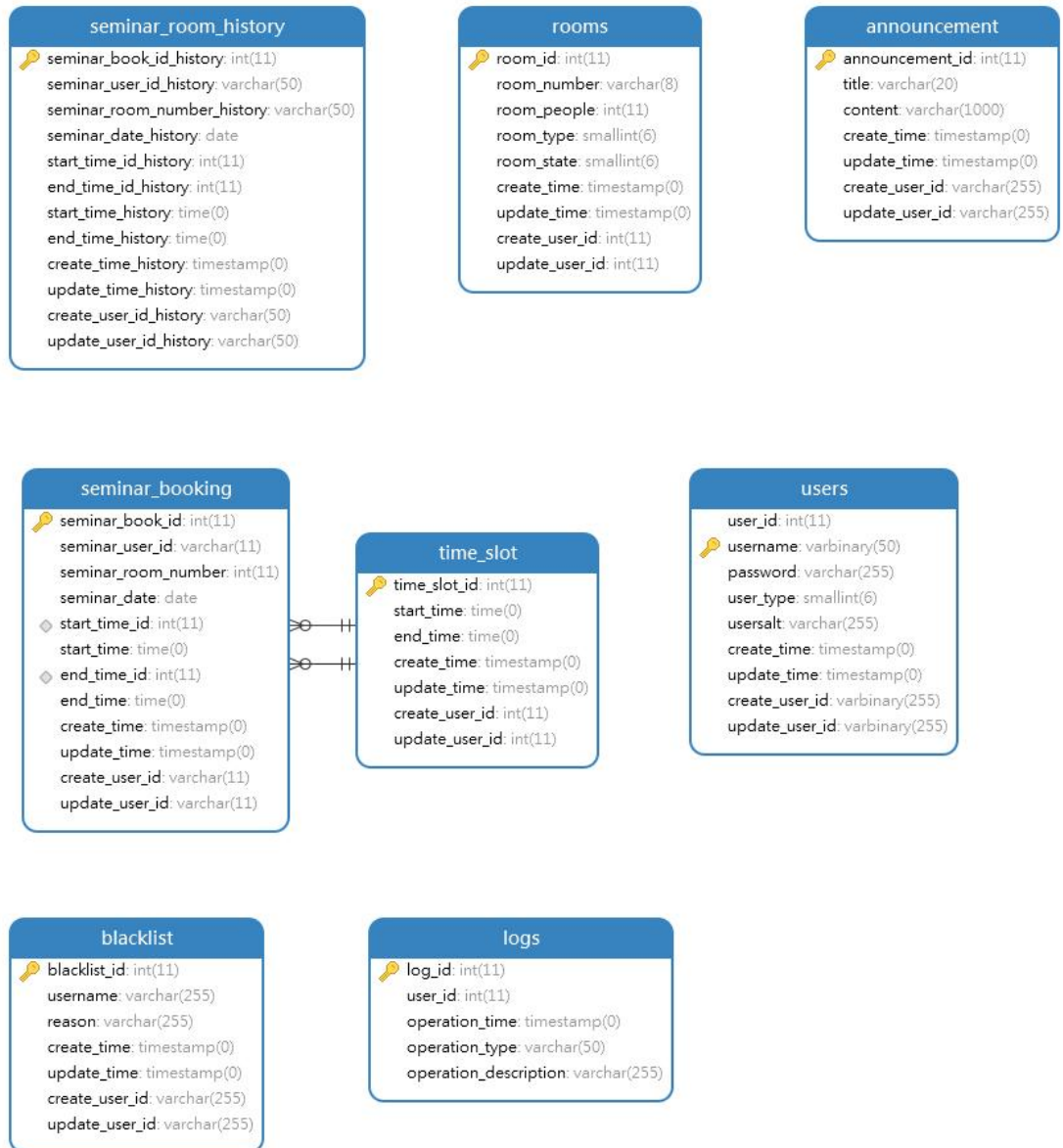
研讨室预约系统类结构图



4 模块设计

4.1 数据库设计

4.1.1 数据库 er 图



4.1.2数据库字段表

(1) 公告表

列名	数据类型	字段类型	长度	是否为空	默认值
announcement_id	int(11)	int	(Null)	NO	(Null)
▶title	varchar(20)	varchar	20	NO	(Null)
content	varchar(1000)	varchar	1000	NO	(Null)
create_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
update_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
create_user_id	varchar(255)	varchar	255	NO	(Null)
update_user_id	varchar(255)	varchar	255	NO	(Null)

(2) 黑名单表

列名	数据类型	字段类型	长度	是否为空	默认值
▶blacklist_id	int(11)	int	(Null)	NO	(Null)
username	varchar(255)	varchar	255	NO	(Null)
reason	varchar(255)	varchar	255	NO	(Null)
create_time	timestamp	timestamp	(Null)	NO	CURRENT_TIMESTAMP
update_time	timestamp	timestamp	(Null)	NO	CURRENT_TIMESTAMP
create_user_id	varchar(255)	varchar	255	NO	(Null)
update_user_id	varchar(255)	varchar	255	NO	(Null)

(3) 日志表

列名	数据类型	字段类型	长度	是否为空	默认值
▶log_id	int(11)	int	(Null)	NO	(Null)
user_id	int(11)	int	(Null)	NO	(Null)
operation_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
operation_type	varchar(50)	varchar	50	YES	(Null)
operation_description	varchar(255)	varchar	255	YES	(Null)

(4) 房间表

列名	数据类型	字段类型	长度	是否为空	默认值
room_id	int(11)	int	(Null)	NO	(Null)
room_number	varchar(8)	varchar	8	NO	(Null)
room_people	int(11)	int	(Null)	NO	(Null)
room_type	smallint(6)	smallint	(Null)	NO	(Null)
room_state	smallint(6)	smallint	(Null)	NO	(Null)
create_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
update_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
create_user_id	int(11)	int	(Null)	NO	(Null)
update_user_id	int(11)	int	(Null)	NO	(Null)

(5) 研讨室预约表

列名	数据类型	字段类型	长度	是否为空	默认值
seminar_book_id	int(11)	int	(Null)	NO	(Null)
seminar_user_id	varchar(11)	varchar	11	NO	(Null)
seminar_room_number	int(11)	int	(Null)	NO	(Null)
seminar_date	date	date	(Null)	NO	(Null)
start_time_id	int(11)	int	(Null)	NO	(Null)
start_time	time	time	(Null)	YES	(Null)
end_time_id	int(11)	int	(Null)	NO	(Null)
end_time	time	time	(Null)	YES	(Null)
create_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
update_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
create_user_id	varchar(11)	varchar	11	NO	(Null)
update_user_id	varchar(11)	varchar	11	NO	(Null)

(6) 历史记录表

列名	数据类型	字段类型	长度	是否为空	默认值
seminar_book_id_history	int(11)	int	(Null)	NO	(Null)
seminar_user_id_history	varchar(50)	varchar	50	YES	(Null)
seminar_room_number_history	varchar(50)	varchar	50	YES	(Null)
seminar_date_history	date	date	(Null)	YES	(Null)
start_time_id_history	int(11)	int	(Null)	YES	(Null)
end_time_id_history	int(11)	int	(Null)	YES	(Null)
start_time_history	time	time	(Null)	YES	(Null)
end_time_history	time	time	(Null)	YES	(Null)
create_time_history	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
update_time_history	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
create_user_id_history	varchar(50)	varchar	50	YES	(Null)
update_user_id_history	varchar(50)	varchar	50	YES	(Null)

(7) 时间段表

列名	数据类型	字段类型	长度	是否为空	默认值
time_slot_id	int(11)	int	(Null)	NO	(Null)
start_time	time	time	(Null)	NO	(Null)
end_time	time	time	(Null)	NO	(Null)
create_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
update_time	timestamp	timestamp	(Null)	YES	CURRENT_TIMESTAMP
create_user_id	int(11)	int	(Null)	NO	(Null)
update_user_id	int(11)	int	(Null)	NO	(Null)

(8) 用户表

列名	数据类型	字段类型	长度	是否为空	默认值
user_id	int(11)	int	(Null)	NO	(Null)
username	varbinary(50)	varbinary	50	NO	
password	varchar(255)	varchar	255	NO	(Null)
user_type	smallint(6)	smallint	(Null)	NO	(Null)
usersalt	varchar(255)	varchar	255	NO	(Null)
create_time	timestamp	timestamp	(Null)	NO	CURRENT_TIMESTAMP
update_time	timestamp	timestamp	(Null)	NO	CURRENT_TIMESTAMP
create_user_id	varbinary(255)	varbinary	255	NO	(Null)
update_user_id	varbinary(255)	varbinary	255	NO	(Null)

4.2 接口设计

4.2.1 用户相关接口

接口清单

序号	接口 URL	描述	方法	传参方式
1	/login	用户登录	POST	json body 参数
2	/notice	查询所有公告	GET	无
3	/seminar/appointments/{seminar_user_id}	查询用户所有预约记录	GET	path 参数
4	/seminar/appointments	添加预约记录	POST	json body 参数
5	/seminar/appointments	取消预约	GET	query 参数

(1) 用户登录验证密码

接口路由： /login

请求方法： POST

请求参数： json 格式 Body 参数

名称	位置	类型	必选	说明
username	body	string	是	用户名
password	body	string	是	用户密码

请求参数示例：

```
{
  "username" : "xxx",
  "password" : "xxx"
}
```

接口响应内容格式：json

响应内容数据结构：

HTTP 状态码：200

名称	类型	必选	说明
code	number	true	200 登录成功 411 用户不存在 412 用户名或密码错误 413 账号已被禁用
data	string	false	none
msg	string	false	none

响应内容示例：

```
HTTP 状态码：200
{
  "code": 200,
  "msg": "登录成功",
  "data": ""
}
{
  "code": 411,
  "msg": "用户不存在",
  "data": ""
}
{
  "code": 412,
  "msg": "用户名或密码错误",
  "data": ""
}
{
  "code": 413,
  "data": "",
  "msg": "账号已被禁用，请联系管理员"}
}
```

接口请求地址：

本地测试请求示例：<http://localhost:8082/login>

服务器：<http://8.134.201.230:8082/login>

(2) 查询所有公告

接口路由： /notice

请求方法： GET

请求参数： 无

接口响应内容格式： json

响应内容数据结构：

HTTP 状态码： 200

名称	类型	必选	说明
code	integer	true	202 成功查询，返回的数据不为空且正确 402 返回的数据为空
data	[object]	true	公告数据列表
announcementId	integer	true	公告 id
title	string	true	标题
content	string	true	内容
createTime	string	true	创建时间
updateTime	string	true	更新时间
createUserId	integer	true	创建用户 id
updateUserId	integer	true	更新用户 id
msg	string	false	none

响应内容示例：

```
HTTP 状态码： 200
{
  "code": 202,
  "data": [
    {
      "announcementId": 1,
      "createTime": "2023-10-25 16:24:59",
      "content": "1. 在规定时间内签到 2. xxx",
      "title": "研讨室预约注意事项",
      "createUserId": 1,
      "updateTime": "2023-10-25 16:24:59",
      "updateUserId": 1
    },
    {
      "announcementId": 2,
      "createTime": "2023-08-18 20:25:46",
      "content": "每天 8: 00-22: 00",
      "title": "研讨室开放时间",
      "createUserId": 1,
      "updateTime": "2023-08-18 20:25:46",
      "updateUserId": 1
    }
  ],
  "msg": "查询公告成功"}

```

接口请求地址：

本地测试请求示例：<http://localhost:8082/notice>

服务器：<http://8.134.201.230:8082/notice>

(3) 查询我的所有预约

接口路由： /seminar/appointments/{seminar_user_id}

请求方法： GET

请求参数： path 参数

名称	位置	类型	必选	说明
seminar_user_id	path	string	是	用户名

请求参数示例：

本地测试请求示例：http://localhost:8082/seminar/appointments/{seminar_user_id}

接口响应内容格式: json

响应内容数据结构:

HTTP 状态码: 200

名称	类型	必选	说明
code	string	true	203 表示成功返回的数据不为空且正确 403 返回数据为空
data	[object]	true	预约记录信息列表
seminar_user_id	string	true	预约人
seminar_room_number	number	true	预约房间号
seminar_date	string	true	预约日期
start_time	number	true	开始时间
end_time	number	true	结束时间
msg	string	false	查询成功

响应内容示例:

```
HTTP 状态码: 200
{
  "code": "203",
  "data": [
    {
      "seminar_user_id": "11",
      "seminar_room_number": 37,
      "seminar_date": "2023-04-14",
      "start_time": 7,
      "end_time": 8
    },
    {
      "seminar_user_id": "11",
      "seminar_room_number": 55,
      "seminar_date": "2003-12-12",
      "start_time": 1,
      "end_time": 2
    }
  ],
  "msg": "查询成功"}
```

(4) 添加预约记录

接口路由：/seminar/appointments

请求方法：POST

请求参数：json body 参数

名称	位置	类型	必选	说明
body	body	object	否	json body 参数
seminarUserId	body	string	是	用户 id
seminarRoomNumber	body	integer	是	房间号
seminarDate	body	string	是	预约日期
startTimeId	body	integer	是	开始时间对应的时间段表 id
endTimeId	body	integer	是	结束时间对应的时间段 id

请求参数示例：

```
{
  "seminarUserId": "12",
  "seminarRoomNumber": 1,
  "seminarDate": "2023-09-10",
  "startTimeId": 1,
  "endTimeId": 3
}
```

接口响应内容格式：json

响应内容数据结构：

名称	类型	必选	说明
code	integer	true	240 预约成功 440 开始时间不可大于结束时间 441 时间与已有预约冲突，请重新预约 442 今日已预约过研讨室，不可再次预约 443 单次预约时间不可超过 2 小时
data	string	false	none
msg	string	false	none

响应内容示例：

```
HTTP 状态码: 200
{
  "code": 240,
  "msg": "预约成功",
  "data": ""
}
```

接口请求地址：

本地测试请求示例：<http://localhost:8082/seminar/appointments>

服务器：<http://8.134.201.230:8082/seminar/appointments>

(5) 取消预约

接口路由：/seminar/appointments

请求方法：GET

请求参数：query 参数

名称	位置	类型	必选	说明
username	query	string	是	用户名
room	query	integer	是	房间号
date	query	string	是	预约日期

请求参数示例：

本地请求示例：

<http://localhost:8082/seminar/appointments?username=xx&room=xx&date=2023-12-10>

接口响应内容格式：json

响应内容数据结构：

名称	类型	必选	说明
» code	integer	true	205 预约取消成功 405 取消预约失败
» msg	string	false	none
» data	string	false	none

响应内容示例：

```
HTTP 状态码: 200
{
  "code": 205,
  "data": "",
  "msg": "预约取消成功"}
```

接口请求地址：

本地测试请求示例：

<http://localhost:8082/seminar/appointments/21xxx/401/2023-11-26>

服务器示例：

<http://8.134.201.230:8082/seminar/appointments/21xxx/402/2021-11-23>

4.2.2 管理员相关接口

接口清单

序号	接口 URL	描述	方法	传参方式
1	/login	管理员登录	POST	json body 参数
2	/blacklist/insert	添加黑名单记录	GET	json body 参数
3	/blacklist/select	查询黑名单记录	GET	无
4	/blacklist/delete	删除黑名单记录	GET	path 参数

(1) 添加黑名单记录

接口路由：/blacklist/insert

请求方法：POST

请求参数：json body 参数

名称	位置	类型	必选	说明
body	body	object	否	json body 参数
» username	body	string	是	待添加用户名
» reason	body	string	是	添加原因

请求参数示例：

```
{
  "username" : "xxx",
  "reason" : "xxx"
}
```

接口响应内容格式：json

响应内容数据结构：

名称	类型	必选	说明
» code	number	true	210 添加成功 450 用户不存在 451 用户已存在黑名单中
» data	string	false	none
» msg	string	true	none

响应内容示例：

```
HTTP 状态码: 200
{
  "code": 210,
  "data": "",
  "msg": "成功加入黑名单"
}
```

接口请求地址：

本地示例：<http://localhost:8081/blacklist/insert>

服务器示例：<http://8.134.201.230:8081/blacklist/insert>

(2) 删除黑名单记录

接口路由：/blacklist/delete

请求方法：GET

请求参数：query 参数

名称	位置	类型	必选	说明
username	query	string	是	用户名

请求参数示例：

本地请求示例：<http://8.134.201.230:8081/blacklist/delete?username={username}>

接口响应内容格式：json

响应内容数据结构：

名称	类型	必选	说明
» code	number	true	206 删除黑名单成功 406 出错了，请检查输入的用户是否正确
» data	string	true	none
» msg	string	true	none

响应内容示例：

```
HTTP 状态码: 200
{
  "code": 206,
  "data": "",
  "msg": "删除黑名单成功"}
{
  "code": 406,
  "data": "",
  "msg": "出错了，请检查输入的用户是否正确"}
```

接口请求地址：

本地示例：<http://localhost:8081/blacklist/delete?username={username}>

服务器示例：
<http://8.134.201.230:8081/blacklist/delete?username={username}>

(3) 查询黑名单

接口路由：/blacklist/select

请求方法：GET

请求参数：无

请求参数示例：

本地测试请求示例：<http://localhost:8081/blacklist/select>

接口响应内容格式：json

响应内容数据结构：

HTTP 状态码: 200

名称	类型	必选	说明
» code	number	true	none
» data	string	true	none
» msg	string	false	none

响应内容示例：

```
HTTP 状态码: 200
{
  "code": "241",
  "msg": "查询成功",
  "data": [
    "20yt2",
    "20yt1",
    "20yjma"
  ]
}
```

接口请求地址：

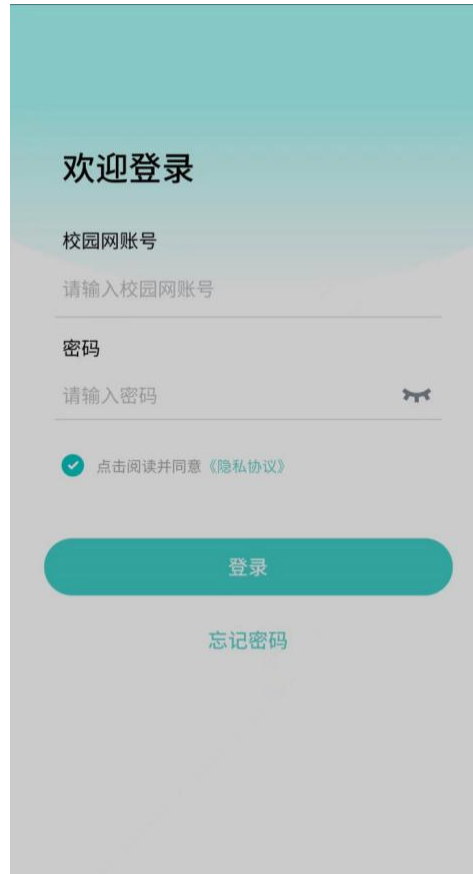
本地测试请求示例：<http://localhost:8081/blacklist/select>

服务器：<http://8.134.201.230:8081/blacklist/select>

4.3 界面设计

一、用户端模块：

1. 登陆模块



The image shows a mobile app login screen. At the top, there's a teal header with the text '欢迎登录' (Welcome to Login). Below the header, there are two input fields: '校园网账号' (Campus Network Account) and '密码' (Password). The '校园网账号' field has a placeholder text '请输入校园网账号' (Please enter campus network account). The '密码' field has a placeholder text '请输入密码' (Please enter password) and a toggle icon for password visibility. Below the password field, there's a checkbox with a green checkmark and the text '点击阅读并同意《隐私协议》' (Click to read and agree to the Privacy Policy). At the bottom, there's a large teal button labeled '登录' (Login) and a link labeled '忘记密码' (Forgot Password).

用户打开小程序时，页面进入登陆界面。账号输入框与密码输入框为**必填**内容，否则用户将无法登陆小程序。**账号和密码都可以是数字，下滑线（英文版本，中文版本不允许），大小写字母，不允许出现其他字符。**登陆该小程序的用户都为拥有汕大校园账号的师生，外校人员无法登陆，**没有注册选项**。登陆该小程序的用户**不可以**进行忘记密码操作，如需修改密码，需要在学校专有修改网页进行修改。用户的账号和密码都是正确的才允许进入首页，如果账号错误**或者**密码错误，提示“账号和密码信息有误，请重新输入”。同时账号和密码正确后，还需要用户**勾选**“点击阅读并同意《隐私协议》”，才允许跳转至首页界面，否则提示用户“请勾选点击阅读并同意《隐私协议》”。完成登陆操作后，小程序跳转至首页界面。

2.首页模块

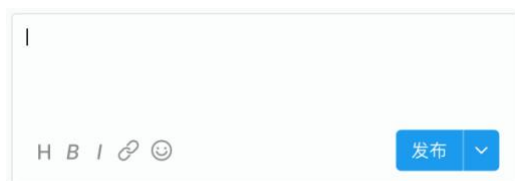


- ① 用户可以通过两种方式跳转到首页：用户在成功登录之后会直接跳转到首页；

或者当用户最下方导航栏的  时可以从其他页面跳转到首页。



- ② 这里展示的是公告信息，按时间排序，最晚发布的公告排在最上面，用户可以通过上下滑动来阅读更多的公告。每条公告由两部分内容组成，第一部分是公告的标题，第二部分是公告的具体内容，当标题或者具体内容因字数太多而不能完全展示时，只展示一部分文字，最后用省略号表示。当用户点击公告的标题或者是内容时，可以进入公告详情页面查看对应公告的全部信息。





- ③ 用户在此处输入文字/链接/表情，点击发布即可发表留言。（H、B、I用于调节字体的格式，H表示高亮，B表示加粗，I表示斜体，是否实现此功能待定）

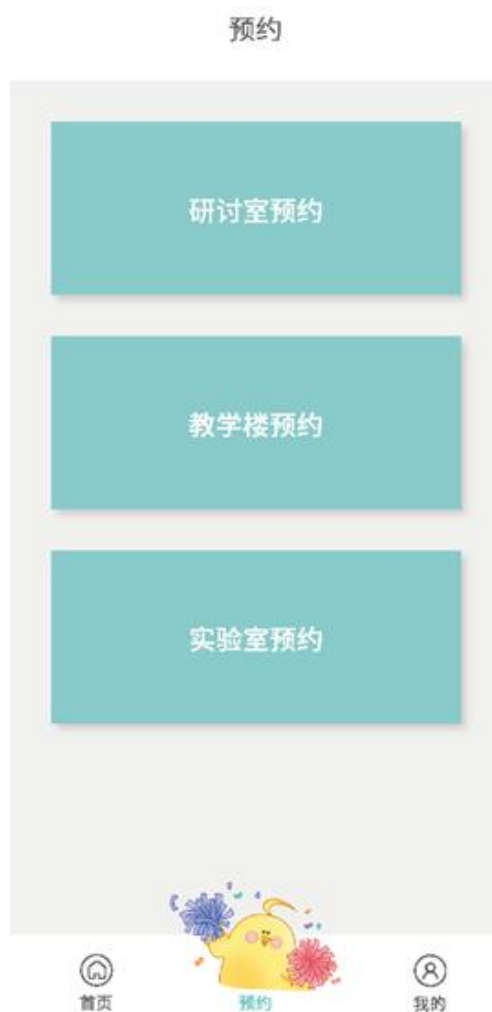


- ④ 此处展示历史留言信息，仅展示发布时间最晚的两条留言，每条留言展示用户的头像、用户的账号姓名、留言内容和发布时间。用户点击“查看历史”，可以展示所有的留言，用户通过上下滑动来阅读留言。



- ⑤ 当用户点击  时，可以跳转到预约界面；当用户点击  时，可以跳转到我的界面。

3.预约模块



用户点击“预约”按钮，小程序跳转至预约界面，该界面从上到下为：研讨室预约、教学楼预约、实验室预约三个部分。用户可以根据需求选择自己需要预约的类型。

研讨室预约

4人研讨室



6人研讨室



10人研讨室



用户点击研讨室预约跳转至研讨室预约界面，用户需要进行研讨室大小以及研讨室编号的选择。

研讨室按照可容纳人数分为 4 人、6 人、10 人三类，通过左右滑动可以查看对应类别下更多的研讨室。用户可以根据需要选择使用研讨室的人数，选择所需的研讨室，直接点击对应研讨室的图标即可查看对应的研讨室详情。比如当用户点击“101”，小程序跳转至下图的界面。

研讨室预约

Room 101

2023 07月

2023 08月

2023 09月

2023 10月

2023 11月

2023 12月

周日

周一

周二

周三

周四

周五

周六

12

13

14

15

16

17

18

有位

有位

有位

有位

有位

有位

有位

请选择当日预约开始时间段

剩余 7

8:00
有位

9:00
我的预约时间

10:00
约满

11:00
约满

13:00
有位

14:00
约满

15:00
约满

16:00
有位

18:00
约满

19:00
约满

20:00
约满

21:00
有位

○ 预约人数

4人

○ 预约时间

9:00-10:00

○ 预约人

XXX

首页

预约

我的

用户点击研讨室后进入上图所示界面，界面显示当前用户选择的房间号，可选预约时间日期（最多可以提前预约 7 天，8：00-22：00），界面最下方可查看当前房间容量等等。用户可以根据最上方的日期栏选择需要预约的月份及日期，点击对应日期的数字，用户即可选中并查看当天的研讨室预约情况。仅可以预约今天到 7 天后的预约（如今天为 1 号，则用户最多可以预约 1 号-7 号的研讨室），如果用户选择不

在这个范围的日期则

请选择当日预约开始时间段

剩余 7

8:00
有位

9:00
我的预约时间

10:00
约满

11:00
约满

13:00
有位

14:00
约满

15:00
约满

16:00
有位

18:00
约满

19:00
约满

20:00
约满

21:00
有位

清空，并在此显示“当前日期不可以预约”。

用户选中日期（如 12 月 11 日）之后，可以选择有空缺的时间段，该时间区按照小时为最小单位，显示每个时间段是否已经被预约。如果某个小时段没有空缺，则该时间

下方会显示“约满”，若该时间段还有空缺位置，则界面中显示“有位”。（用户可以在该区域右上角，**查看当前研讨室的剩余时间段容量**。建议删除该功能，理由：这个功能对于我们的小程序重要性不大，且实施逻辑困难，需要对已经预约的时间段进行统计，统计没有预约了的时间段，需要对每个房间，12 个时间段都有记录，与我们当前开发状况不符。）

用户点击任意的开始时间段按钮，确定预约使用人数，并选择所需的预约时间段（如 9:00-10:00），例如用户在上一个界面选择了九点，在弹出的时间选择框的开始时间可以选择九点或者九点半，在点击预约时间之后，如下图所示，用户需在弹出的窗口滚动中选择具体的预约时间段，确定开始时间以及结束时间，并点击“**确认**”按钮，确认选中该时间段，并**确定**预约该研讨室。在此过程中，如果用户想更改其他的预约设置，可以点击“**取消**”按钮，重新回到日期选择的界面。预约开始时间和预约结束时间不可以相等，且预约开始时间要早于预约结束时间至少半个小时，预约时间段不能超过 2 个小时（即预约时间段为半个小时至 2 个小时），用户与用户的预约时间至少要间隔半个小时才允许预约，如 A 已经预约 9:00-10:00，则 B 如果要预约的话，至少要从 10:30 开始预约，预约时间可以为 10:30-11:00，但是不可以为 10:00-11:00（用户与用户的预约时间至少要间隔半个小时）。

如果当天用户已经预约过研讨室，则不允许再次进行预约，提示“预约失败, 已经预约了研讨室，无法再次进行预约”。



用户按照以上预约流程操作之后，系统会根据预约的情况给予用户反馈，若用户预约成功，则如下图显示预约成功弹窗。至此，用户完成了对研讨室的预约。



4.我的模块：



“我的”模块，主要为用户提供个人信息查看管理功能，用户点击页面右下角“我的”进入“我的”页面，展示用户账号信息，提供查看用户个人预约信息、留言反馈、联系客服、设置修改。

当用户点击“我的预约”将跳转到用户预约记录界面，当用户点击“留言反馈”，跳转到“留言”输入界面，当用户点击“联系客服”将跳转到客服联系方式界面，当用户点击“设置”将跳转到设置界面。

二、管理员模块

1. 登录模块





The image shows a login interface for administrators. It features a light purple background with a white login box. The title '欢迎登录' (Welcome to Login) is centered at the top. Below it, there are two input fields: '管理员账号' (Admin Account) and '密码' (Password). The 'Admin Account' field has a placeholder text '请输入管理员账号' (Please enter admin account). The 'Password' field has a placeholder text '请输入密码' (Please enter password) and a toggle icon for visibility. Below the password field, there is a checkbox with a blue checkmark and the text '点击阅读并同意《隐私协议》' (Click to read and agree to the Privacy Policy). At the bottom, there is a large blue button with the text '登录' (Login).

管理员打开管理员小程序时，页面进入登陆界面。账号输入框与密码输入框为**必填**内容，否则管理员将无法登陆小程序。**账号和密码都可以是数字，下滑线（英文版本，中文版本不允许），大小写字母，不允许出现其他字符。**登陆该小程序的用户都为拥有汕大校园账号的管理员，外校人员无法登陆，**没有注册选项。**登陆该小程序的用户**不可以**进行忘记密码操作，如需修改密码，需要在学校专有修改网页进行修改。用户的账号和密码都是正确的才允许进入首页，如果账号错误**或者**密码错误，提示“账号和密码信息有误，请重新输入”。同时账号和密码正确后，还需要用户**勾选**“点击阅读并同意《隐私协议》”，才允许跳转至首页

界面，否则提示用户“请勾选点击阅读并同意《隐私协议》”。完成登陆操作后，小程序跳转至首页界面。




2. 公告管理模块



管理员进入小程序之后，点击  可以进入公告管理页面，在文本框中输入需要添加的公告内容，点击  按钮即可向全体用户发送公告。



3. 留言管理模块



管理员点击  即可进入留言管理界面，该界面会显示用户发送的留言，管理员可以查留言内容并对其进行审核，如留言存在内容违规、不友善的内容，可以点击  按钮对其进行删除，点击  ，管理员可以对留言进行回复。

4. 用户管理模块



点击  按钮，进入用户管理界面。该界面会显示一个黑名单表格，表格中会显示已经被添加到黑名单的用户，该表格一次性可以显示四行四列的黑名单用户，管理员可以通过上下滑动查看更多黑名单中的用户。点击表格中的  用户框，会弹出一个提示窗口，



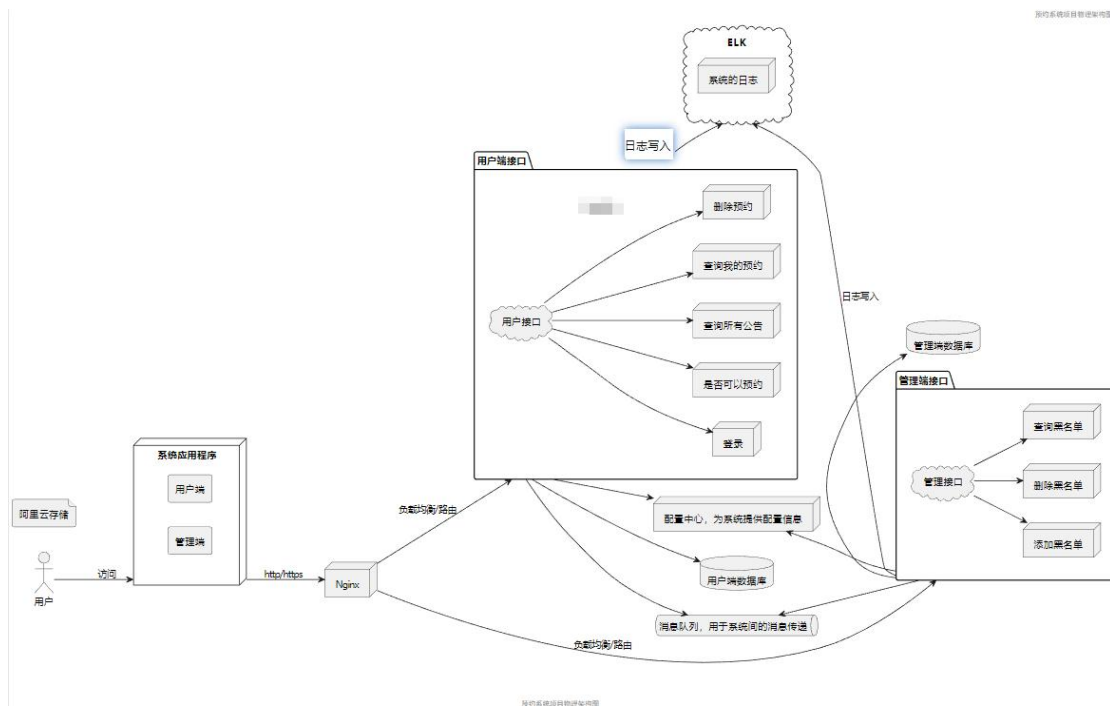
根据实际情况的需求，管理员可以选择将已经加入黑名单的用户移出黑名单。

在表格下方，界面会显示一个搜索框，



管理员可以在此处输入将要加入黑名单的用户，并点击将其加入黑名单中。

5 部署维护



6 变更管理和版本控制

1. 版本控制工具和配置

- (1) 使用 GitHub 作为版本控制工具。
- (2) 创建 GitHub 仓库用于托管代码。
- (3) 配置仓库的分支策略，所有开发成员将代码提交到主分支。

2. 分支管理策略

- (1) 使用主分支作为稳定版本的发布分支。
- (2) 所有开发成员直接在主分支上进行开发和提交更改。

3. 代码提交规范

- (1) 提供代码提交的规范和约定。
- (2) 使用语义化版本控制规范（Semantic Versioning）作为提交消息的格式，

例如："feat: 添加用户注册功能"。

- (3) 提供开发人员如何使用 Git 命令行或图形界面工具进行代码提交的说明。

4. 代码合并流程

- (1) 开发人员使用 GitHub 的 Pull Request 功能进行代码合并。
- (2) 每个功能或修复都应创建一个 Pull Request。
- (3) 邀请其他成员进行代码审查，并解决审查意见。
- (4) 确保通过自动化测试，并解决可能的冲突。
- (5) 完成代码审查和测试后，由负责合并的人员进行合并操作。

5. 备份和恢复策略

- (1) 确保对 GitHub 仓库进行定期备份。
- (2) 配置自动化备份策略，以保护代码仓库和相关资源。
- (3) 准备应急方案，包括处理仓库损坏或意外删除的情况。

6. 安全性考虑

- (1) 使用 GitHub 的访问控制功能，限制对代码仓库的访问权限。
- (2) 遵循最佳实践，保护敏感信息，如 API 密钥和凭证。
- (3) 加密和安全存储敏感数据，如数据库连接字符串。

7. 培训和文档

- (1) 提供培训和文档资源，帮助团队成员学习和掌握 GitHub 和版本控制工具的使用。
- (2) 提供教程、指南和参考文档的链接，以帮助开发人员了解版本控制的最佳实践。

7 安全设计

7.1 安全策略

- 1.身份验证: 用户必须通过登录才能使用系统，我们采用用户名和密码的方式进行用户身份验证。
- 2.权限管理: 根据用户角色和权限的不同，系统会为用户分配不同的操作权限，确保只有拥有相应权限的用户才能对系统进行操作。
- 3.数据加密: 所有存储在服务器上的数据都会进行加密处理，以防止数据被窃取或篡改。
- 4.防 SQL 注入: 所有用户输入的数据都会进行防 SQL 注入的过滤和校验，防止恶意用户通过输入特殊字符进行攻击。
- 5.防跨站脚本攻击: 对所有用户输入的数据进行跨站脚本攻击的预防和检查，防止用户浏览器被注入恶意脚本。

7.2 安全机制

- 1.身份验证机制: 系统提供注册、登录功能，并使用密码哈希等方式存储用户密码，保证用户信息的安全。
- 2.权限管理机制: 系统采用角色权限管理机制，根据用户角色分配不同的操作权

限，确保只有拥有相应权限的用户才能对系统进行操作。

3.数据加密机制：所有存储在服务器上的数据都会进行加密处理，采用 AES 等加密算法对数据进行加密存储，确保数据在传输和存储过程中的安全性。

4.安全防护机制：系统部署在安全性能较高的服务器上，安装防火墙、入侵检测等安全防护工具，防止恶意攻击和入侵。

5.安全审计机制：系统部署安全审计系统，对所有操作行为进行记录和审计，及时发现并处理安全问题。

8 性能设计

8.1 性能设计考虑

1.响应时间：响应时间是用户使用小程序的重要指标之一，我们需要确保系统能够快速响应用户的请求。可以通过优化数据库查询、缓存技术、异步处理等方式来提高响应速度。

2.资源利用：系统需要合理利用服务器资源，避免资源浪费和过度消耗。可以通过动态调整服务器资源分配、使用负载均衡等技术来提高资源利用率。

3.负载均衡：随着用户数量的增加，系统需要能够应对高负载情况。可以通过添加服务器节点、使用负载均衡器等技术来分担负载，确保系统在高负载情况下能够稳定运行。

8.2 性能设计策略

1.数据库优化：使用合适的数据模型和查询语句，减少数据库查询时间；使用缓存技术，减少对数据库的访问次数。

2.异步处理：将一些非实时性的任务进行异步处理，减少对用户界面的响应时间。

动态资源分配：根据系统的负载情况动态调整服务器资源分配，确保系统在高负载情况下能够稳定运行。

3.负载均衡器：添加服务器节点，并使用负载均衡器将请求分发到不同的服务器节点上，确保系统能够应对高负载情况。

8.3 性能测试与优化

1.进行性能测试：在系统开发过程中，进行性能测试，确保系统在各种场景下都能够满足性能要求。

- 2.监控与日志：实时监控系统的性能指标，如响应时间、资源利用率等，并及时记录日志，以便于发现问题并进行优化。
- 3.代码优化：对代码进行优化，减少代码的运行时间，提高系统的性能。例如，优化算法、减少重复计算等。
- 4.反馈与改进：根据测试结果和用户反馈，不断改进和优化系统性能，提高用户体验。

9 测试策略

- 1.测试范围：根据需求文档和设计文档，确定测试范围，包括功能测试、性能测试、安全测试等方面。
- 2.测试环境：搭建与生产环境一致的测试环境，确保测试结果的准确性。
- 3.测试方法：采用黑盒测试、白盒测试等多种测试方法，确保系统功能和质量。

9.1 测试用例

1.功能测试：

- a. 注册功能：验证用户注册信息的正确性，包括用户名、密码等。
- b. 登录功能：验证用户登录信息的正确性，包括用户名、密码、验证码等。
- c. 预约功能：验证用户预约流程的正确性，包括选择实验室、时间、人数等。
- d. 取消预约功能：验证用户取消预约流程的正确性。
- e. 查询已预约功能：验证用户查询已预约信息的正确性。

2.性能测试：

- a. 响应时间测试：测试系统在不同负载下的响应时间，确保系统能够满足用户需求。
- b. 负载测试：模拟大量用户同时使用系统的情况，测试系统的稳定性和性能。
- c. 压力测试：通过不断增加系统负载，测试系统的各项指标是否符合要求。

3.安全测试：

- a. 登录安全：测试密码加密存储、密码找回等功能的安全性。
- b. 权限管理：验证权限分配的合理性，防止越权操作。
- c. 数据加密：验证数据传输和存储过程中的安全性。

9.2 测试计划

1.测试阶段：将测试分为单元测试、集成测试、系统测试、验收测试等阶段，确保测试的全面性和准确性。

2.人员分配：根据测试内容和要求，合理分配测试人员和时间，确保测试进度和质量。

3.风险控制：及时发现并处理潜在的风险和问题，确保系统质量和稳定性。

测试用例执行：按照测试用例的要求和顺序执行测试，确保每个功能点和细节都得到覆盖。

4.缺陷跟踪：对发现的缺陷和问题及时记录和跟踪，直至解决和关闭。

5.总结与优化：对测试结果进行总结和分析，针对存在的问题提出优化建议，为下一次测试提供参考和借鉴。