2024-1-9

创新创业管理系统

李泽祥

目录

开发项目的背景与简介

学校实验室预约方式一直停留在线下纸质申请或者人工电话联系的阶段，预约流程繁琐费时，不便于学生们快速高效地使用校内实验室资源。同时，他人预约信息不透明，很难作出合理的自己的预约安排，这在某种程度上也浪费了校内实验室的资源。

针对这种情况，我们设计了一款实验室预约小程序服务于西海岸 58 创新工坊，旨在为学生提供便捷、高效、透明的实验资源预约服务，让学生可以方便地预约自己需要使用的资源。

页面结构和布局:

登录页面:

包括用户名输入框、密码输入框和登录按钮。

注册页面:

包括用户名输入框、密码输入框、确认密码输入框和注册按钮。

实验室预约页面:

展示可用的实验室选项和时间段，并提供创建预约和取消预约的功能。

预约详情页面：

展示用户已有的预约记录，并提供查看和取消预约的功能。

登录验证：

在用户点击登录按钮时，验证用户名和密码是否符合要求，如长度、字符类型等，并提示错误信息。

注册验证：

在用户点击注册按钮时，验证用户名、密码和确认密码是否符合要求，并提示错误信息。

预约时间选择：

提供日历或时间选择器，让用户能够方便地选择预约的时间段。

预约冲突检测：

在用户选择预约时间段后，检测该时间段内是否存在冲突的预约，并给出相应的提示信息。

预约创建和取消：

提供创建预约和取消预约的按钮，在用户点击时进行相应的操作，并更新预约状态。

实验室列表展示:

在实验室预约页面中展示可用的实验室列表，包括实验室名称、容量、设备等信息。

预约记录展示:

在预约管理页面中展示用户已有的预约记录，包括实验室名称、预约时间段等信息。

预约记录管理:

提供查看和取消预约的按钮，在用户点击时进行相应的操作，并更新预约状态。

响应式设计:

使用响应式布局，使得前端界面在不同设备上都能良好展示和操作，包括手机、平板和电脑等。

框架选择:

微信小程序框架：使用微信官方提供的小程序框架进行开发。该框架具有良好的兼容性和稳

定性，能够快速开发小程序应用。

UI 库选择:

Vant: Vant 是一套基于微信原生视觉体系的 UI 库，可以满足小程序的页面布局和样式需求。该库具有丰富的组件和样式类，能够快速搭建美观的小程序界面。
网络请求库选择:

微信官方 API:

微信小程序提供了内置的网络请求 API，可以直接使用 wx.request 进行网络请求。该 API 集成在微信小程序框架中，能够方便地与后端进行数据交互。

安全设计:

对用户输入的敏感信息进行合法性验证，防止恶意攻击。
使用 HTTPS 协议保证数据传输的安全性。
在前端进行基本的输入验证，防止 XSS 攻击等安全漏洞。

路由接口说明

/appointment/views.py
all(): 获取用户所有预约信息
请求方法：GET
请求 URL：/api/appointments
请求参数：无需参数
返回结果：返回当前用户所有预约信息的 JSON 格式列表。如果没有任何预约信息，则返回一个空列表。

add(): 创建预约
请求方法：POST
请求 URL：/api/appointments
请求参数：
user_id: 预约用户的 ID（必须）

equipment_id: 预约设备的 ID（必须）
start_time: 预约开始时间（必须）
end_time: 预约结束时间（必须）
返回结果：如果创建成功，返回创建的预约信息的 JSON 格式；否则返回错误信息。

delete(): 删除预约
请求方法：POST
请求 URL：/api/appointments/:id/delete （:id 为要删除的预约 ID）
请求参数：无需参数
返回结果：如果删除成功，返回一个空 JSON；否则返回错误信息。

edit(): 编辑预约
请求方法：POST
请求 URL：/api/appointments/:id/edit （:id 为要编辑的预约 ID）
请求参数：
user_id: 预约用户的 ID
equipment_id: 预约设备的 ID
start_time: 预约开始时间
end_time: 预约结束时间
返回结果：如果编辑成功，返回编辑后的预约信息的 JSON 格式；否则返回错误信息。

search_by_user(): 查询某个用户的所有预约信息
请求方法：GET
请求 URL：/api/appointments/user/:id （:id 为要查询的用户 ID）
请求参数：无需参数
返回结果：返回该用户的所有预约信息的 JSON 格式列表。如果该用户没有任何预约信息，则返回一个空列表。

search_by_equipment(): 根据设备 ID 查询当前已经预约的时间段
请求方法：GET
请求 URL：/api/appointments/equipment/:id （:id 为要查询的设备 ID）
请求参数：无需参数
返回结果：返回该设备已经预约的时间段的 JSON 格式列表。如果该设备在当前时间段没有任何预约信息，则返回一个空列表。

allow(): 显示每种设备在当天还可以预约的时间段
请求方法：GET
请求 URL：/api/appointments/allow
请求参数：无需参数
返回结果：返回每种设备在当天还可以预约的时间段的 JSON 格式列表。如果某种设备在当天已经被预约满了，则不会包含在返回结果中。网 IP 的方式。

/collect/views.py
all_award()：
请求方法：GET
请求 URL：/all_award
请求参数：无需参数
返回结果：返回当前用户所有获奖信息的 JSON 格式列表。如果没有任何获奖信息，则返回一个空列表。

commit()：
请求方法：POST
请求 URL：/commit
请求参数：
name: 获奖名称（必须）
date: 获奖日期（必须，格式为"%Y-%m-%d"，例如："2022-01-01"）
level: 获奖等级（必须）
institution: 颁奖机构（必须）
user_name: 获奖人姓名（必须）
返回结果：如果提交成功，返回一个包含 "msg" 字段的 JSON；否则返回包含错误信息的 JSON。

/equipment/views.py
/all 接口：
请求方法：GET
请求 URL：/all
请求参数：无
返回结果示例：

```
{
  "msg": "获取成功",
  "data": [
    {
      "id": 1,
      "name": "设备 1",
      "location": "地点 1",
      "kind": "类型 1",
      "capacity": 100,
      "image_url": "http://example.com/image1.jpg"
    },
    {
      "id": 2,
      "name": "设备 2",
      "location": "地点 2",
      "kind": "类型 2",
      "capacity": 200,
      "image_url": "http://example.com/image2.jpg"
```

```
    }
  ]
}
```

/add 接口：
请求方法：POST
请求 URL：/add
请求参数：

```
{
  "name": "设备名称",
  "location": "设备位置",
  "kind": "设备类型",
  "capacity": 100,
  "image_url": "http://example.com/image.jpg"
}
```

返回结果示例：

```
{
  "msg": "添加成功",
  "equipment_id": 3,
  "equipment_name": "设备名称"
}
```

/delete 接口：
请求方法：POST
请求 URL：/delete
请求参数：

```
{
  "equipment_id": 3
}
```

返回结果示例：

```
{
  "msg": "成功删除设备, 设备名: 设备名称"
}
```

/edit 接口：

请求方法：POST
请求 URL：/edit
请求参数：

```
{
  "equipment_id": 3,
  "name": "新设备名称",
  "location": "新设备位置",
  "kind": "新设备类型",
  "capacity": 150,
```

```
  "image_url": "http://example.com/new_image.jpg"
}
```
返回结果示例：
```
{
  "msg": "成功修改设备信息",
  "id": 3,
  "name": "新设备名称",
  "location": "新设备位置",
  "kind": "新设备类型",
  "capacity": 150,
  "image_url": "http://example.com/new_image.jpg"
}
```

/upload_image 接口：
请求方法：POST
请求 URL：/upload_image
请求参数：包含一个名为 image 的文件字段，值为上传的图片文件
返回结果示例：
```
{
  "msg": "上传成功",
  "image_url": "http://example.com/uploaded_image.jpg"
}
```

/upload_inside_image 接口：
请求方法：POST
请求 URL：/upload_inside_image
请求参数：包含一个名为 image 的文件字段，值为上传的图片文件
返回结果示例：
```
{
  "msg": "上传成功",
  "image_url": "http://example.com/uploaded_inside_image.jpg"
}
```

/disable 接口：
请求方法：POST
请求 URL：/disable
请求参数：
```
{
  "equipment_id": 3
}
```
返回结果示例：
```
{
  "msg": "成功禁用设备，设备名: 设备名称"
}
```

/enable 接口：
请求方法：POST
请求 URL：/enable
请求参数：

```
{
  "equipment_id": 3
}
```

返回结果示例：

```
{
  "msg": "成功解除禁用设备，设备名: 设备名称"
}
```

/sign/views.py
/all 获取所有设备信息接口：
请求方法：GET
请求 URL：/all
请求参数：无
返回结果示例：

```
{
  "msg": "获取成功",
  "data": [
    {
      "id": 1,
      "name": "设备 1",
      "location": "地点 1",
      "kind": "类型 1",
      "capacity": 100,
      "image_url": "http://example.com/image1.jpg"
    },
    {
      "id": 2,
      "name": "设备 2",
      "location": "地点 2",
      "kind": "类型 2",
      "capacity": 200,
      "image_url": "http://example.com/image2.jpg"
    }
  ]
}
```

/add 添加设备接口：
请求方法：POST
请求 URL：/add

请求参数：

```
{
  "name": "设备名称",
  "location": "设备位置",
  "kind": "设备类型",
  "capacity": 100,
  "image_url": "http://example.com/image.jpg"
}
```

返回结果示例：

```
{
  "msg": "添加成功",
  "equipment_id": 3,
  "equipment_name": "设备名称"
}
```

删除设备接口 (/delete)：

请求方法：POST
请求 URL：/delete
请求参数：

```
{
  "equipment_id": 3
}
```

返回结果示例：

```
{
  "msg": "成功删除设备, 设备名: 设备名称"
}
```

/edit 修改设备信息接口：
请求方法：POST
请求 URL：/edit
请求参数：

```
{
  "equipment_id": 3,
  "name": "新设备名称",
  "location": "新设备位置",
  "kind": "新设备类型",
  "capacity": 150,
  "image_url": "http://example.com/new_image.jpg"
}
```

返回结果示例：

```
{
  "msg": "成功修改设备信息",
  "id": 3,
  "name": "新设备名称",
```

```
  "location": "新设备位置",
  "kind": "新设备类型",
  "capacity": 150,
  "image_url": "http://example.com/new_image.jpg"
}
```

/upload_image 上传设备图片接口：
请求方法：POST
请求 URL：/upload_image
请求参数：包含一个名为 image 的文件字段，值为上传的图片文件
返回结果示例：

```
{
  "msg": "上传成功",
  "image_url": "http://example.com/uploaded_image.jpg"
}
```

上传设备内部图片接口 (/upload_inside_image)：
请求方法：POST
请求 URL：/upload_inside_image
请求参数：包含一个名为 image 的文件字段，值为上传的图片文件
返回结果示例：

```
{
  "msg": "上传成功",
  "image_url": "http://example.com/uploaded_inside_image.jpg"
}
```

/disable 禁用设备接口：
请求方法：POST
请求 URL：/disable
请求参数：

```
{
  "equipment_id": 3
}
```

返回结果示例：

```
{
  "msg": "成功禁用设备，设备名: 设备名称"
}
```

/enable 解除禁用设备接口：
请求方法：POST
请求 URL：/enable
请求参数：

```
{
  "equipment_id": 3
```

}
返回结果示例：
{
  "msg": "成功解除禁用设备，设备名: 设备名称"
}

/user/views.py
login() - 处理用户登录请求
请求方法：POST
请求 URL：/user/login
请求参数：表单数据
返回结果：JSON 格式的响应数据

register() - 处理用户注册请求
请求方法：POST
请求 URL：/user/register
请求参数：表单数据
返回结果：JSON 格式的响应数据

login_out() - 处理退出登录请求
请求方法：GET
请求 URL：/user/login_out
请求参数：无
返回结果：JSON 格式的响应数据

get_image_code() - 获取图片验证码
请求方法：GET
请求 URL：/user/get_image_code
请求参数：无
返回结果：带有图片验证码的 HTTP 响应

send_code_by_email() - 通过邮箱发送验证码
请求方法：POST
请求 URL：/user/send_code_by_email
请求参数：JSON 数据，包含邮箱地址
返回结果：JSON 格式的响应数据

reset_password() - 找回密码（未登录的情况）
请求方法：POST
请求 URL：/user/reset_password
请求参数：表单数据
返回结果：JSON 格式的响应数据

change_password() - 修改密码（已登录的情况）

请求方法：POST
请求 URL：/user/change_password
请求参数：表单数据
返回结果：JSON 格式的响应数据

change_userinfo() - 修改用户信息（已登录的情况）
请求方法：POST
请求 URL：/user/change_userinfo
请求参数：表单数据
返回结果：JSON 格式的响应数据

算法说明

canlendar.js 实现了一个日历生成工具
包括了一个名为 dyCalendar 的对象，其中定义了一个 draw 方法。该方法接受一个 option 对象作为参数，用于指定生成日历的一些选项，比如目标位置、日期格式、是否高亮显示今天等。
在 draw 方法内部，首先根据选项中的年份和月份计算出该月的第一天是星期几，然后根据该信息创建出该月份的表格，并依次填充每个单元格，从 1 号开始到该月最后一天。同时，如果选项中指定了要高亮显示今天或者其他特定日期，也会在相应的单元格上添加 CSS 类名以实现高亮显示。
整个生成过程都是通过调用一系列辅助函数来完成的。除了绘制月份表格的函数外，还包括了一些用于处理日期、点击事件和 CSS 样式的函数。通过将这些功能函数封装在一个对象中，并提供一个简单的 API 来调用它们，可以使得用户能够灵活地定制自己需要的日历样式和功能。

header.js 实现了一个简单的导航栏点击切换功能
它首先通过 document.querySelectorAll('.nav>a')选择所有导航栏中的链接元素，并将它们放入一个 NodeList 对象中。然后，通过一个 for 循环遍历每个链接元素，并为它们绑定了一个 onclick 事件。当用户点击某个链接时，对应的事件处理函数就会被调用。在事件处理函数内部，首先通过 document.getElementsByClassName('active')[0]找到当前处于激活状态的链接元素，然后移除其 active 类名，以便在接下来的操作中能够正确地添加到新的激活元素上。接着，通过 event.target 获取到当前被点击的链接元素，并给它添加 active 类名，以此将其标记为激活状态。

home.js 用于创建日历的 JavaScript 库
它包括了一些函数来创建月份表格、绘制日历月份表格、获取日历详细信息、绘制日历以及处理用户点击事件的功能。
在这段代码中，作者使用了自执行函数来创建一个全局对象 dycalendar，并向

这个对象添加了一些方法，比如 draw 方法用于绘制日历，getData 方法用于获取某一天的预约数据，以及一些辅助函数用于处理日历的绘制和用户交互。

news.js
首先，通过 document.querySelector 和 document.querySelectorAll 获取到左右按钮、小圆点和图片容器等 DOM 元素。然后，定义变量 index 和 last 用于记录当前显示的图片索引和上一张图片的索引。
position 函数用于设置图片容器的 left 样式，使其根据当前索引显示对应的图片。
add 函数用于增加索引，并判断是否需要循环到第一张图片。
desc 函数用于减小索引，并判断是否需要循环到最后一张图片。
timer 函数是一个定时器，用于每隔 3 秒自动切换图片。
dot 函数用于设置当前小圆点的样式。接下来，给左右按钮添加点击事件监听器，在点击事件中调用相应的函数进行切换图片和更新样式，并清除定时器重新启动。最后，调用 timer 函数启动定时器。

script.js 用于显示已预约的时间，并提供签到功能
showReservedTime 函数用于显示已预约的时间。首先，它会清空原先的预约时间列表。然后，根据当前日期和时间，构造起始时间和结束时间的请求体。使用 fetch 函数向服务器发送 POST 请求，获取已预约的时间数据。然后，根据一定的规则对时间进行排序，包括是否过期、是否已签到等。排序完成后，将时间数据渲染到预约时间列表中。对于每个预约时间，创建一个 tr 元素，并填充相关信息，如设备名称、申请人姓名、开始时间、结束时间等。如果预约时间已过期，则显示"已过期"按钮，并禁用点击事件。如果已签到，则显示"已签到"按钮，并禁用点击事件。如果未审核，则显示"未审核"按钮，并禁用点击事件。如果可以签到，则显示"签到"按钮，并绑定点击事件。
handleSignIn 函数用于处理签到逻辑。当用户点击"签到"按钮时，向服务器发送签到请求，并在成功后刷新页面并显示签到成功的提示。
init 函数用于初始化页面。它禁用了打印功能，并在页面加载完成后调用 showReservedTime 函数显示已预约的时间。同时，还设置了定时器，每隔 5 分钟刷新一次预约时间列表。

小组成员及分工

李泽祥（组长）:
1. 开发小程序前端
2. 打通前后端
3. 开发部分后端代码

李迅:
1. 开发小程序前端

2. 部署数据库和项目代码至服务器

李申帅:

1. 开发静态签到页面
2. 开发部分后端代码