作品描述

「雪灾蔗大」是一个课程笔记共享平台，在这里，每个人既是内容的消费者，也是内容的发布者。

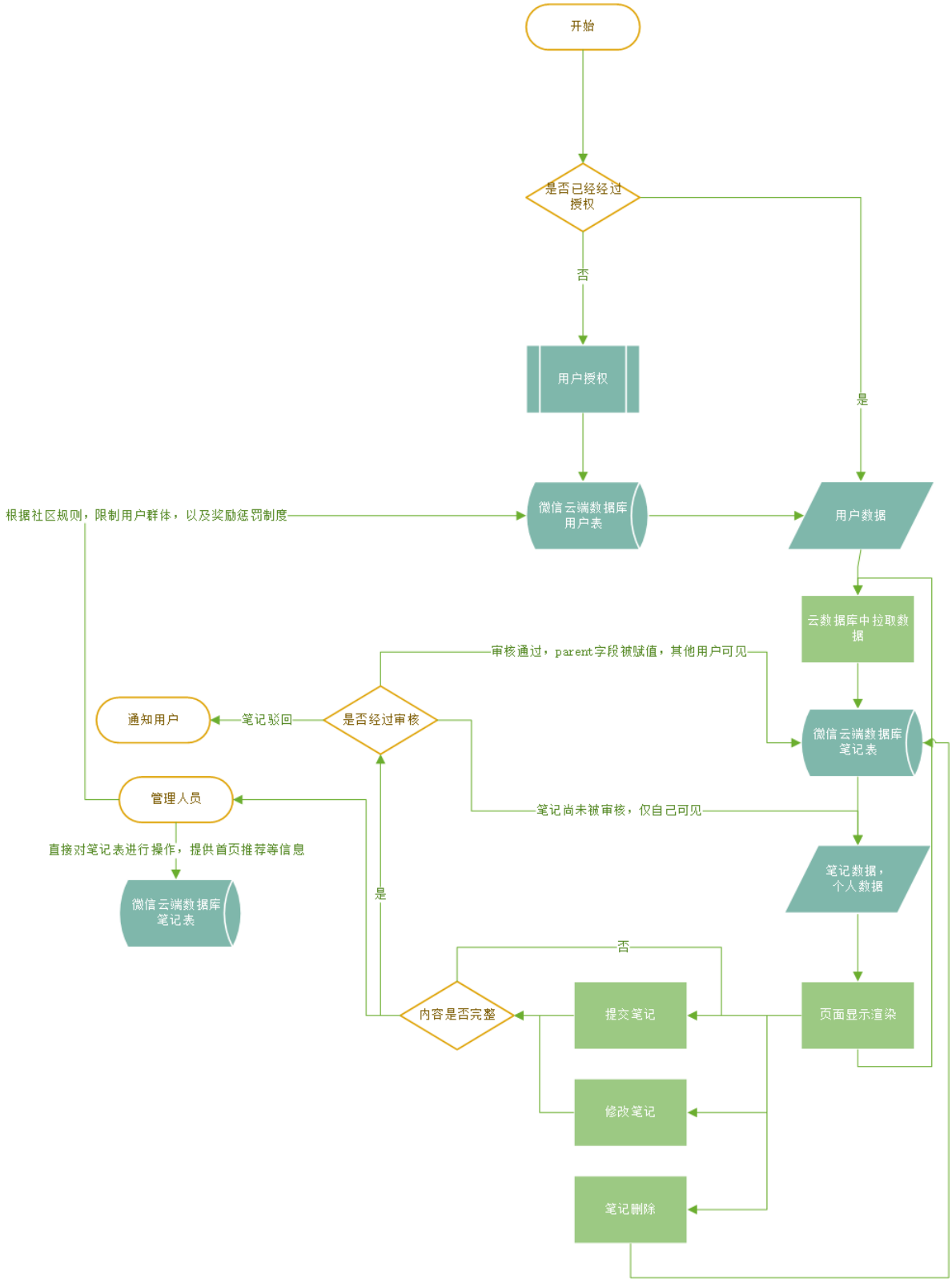
每个人都可以通过这个平台来上传自己的课程笔记，包括文字、图片等，同时对笔记拥有丰富的编辑方式。对于上传的课程笔记，我们利用了类目和tags进行分类归档，同时也可以通过搜索来找到需要的课程笔记。

我们的平台功能包括：

- 上传笔记：用户可以以富文本的形式编辑笔记并上传，经后台审核笔记可以公开
- 修改笔记：用户随时可以对已经提交的笔记进行修改发布
- 浏览笔记：用户可以在多个页面浏览丰富的笔记
- 收藏笔记：用户可以收藏心仪的笔记，并且在个人中心处浏览收藏的笔记
- 模糊搜索：小程序支持根据标签、课程名、笔记名的模糊搜索
- 聊天解答：用户可以在聊天室反馈自己的需求以及问题
- 评论、设置公开性、显示笔记被收藏次数等功能将在未来完善

我们希望，通过这个平台，能让有价值的笔记被每一个人看到，也能让每一个有需要的人能找到自己心仪的笔记。

开发方案



技术要点

组件

为了方便地重复使用某些类似的功能或页面，项目主要设置了`curri`、`recommand`、`editor`三个组件。

curri

文件夹显示组件，用于对于文件夹对象的显示及交互。

recommand

笔记显示组件，用于对于笔记对象的显示及交互。

富文本编辑器editor

笔记分享的一个重点就是笔记要如何生成，考虑到笔记的排版、公式、字体等要求，一般的文本输入不能满足笔记的要求。因此我们充分利用了微信小程序的富文本标签，编写了处理富文本输入的组件`editor`。

该富文本编辑器，可以将富文本的各种操作作为按钮，通过可视化的方式方便用户使用。

输入完成后，编辑器将最后的样式以富文本（字符串）返回并保存到数据库，在`notedetail`页面中将使用`<rich-text>`标签对富文本进行显示。

富文本按钮

数据库使用

三个集合：

- `NOTE`
  - 用来存储所有用户上传的笔记的相关信息，还有文件夹如“大雾”、“DS”；对笔记的增删查改都通过处理该数据库实现
  - `_id`：元素（笔记）的唯一标识
  - `_openid`：上传者的唯一标识
  - `classified`：笔记是否已经被后台工作人员分类，且被放在首页推荐上
  - `column`：笔记所属栏目
  - `creator`：上传者的昵称
  - `html`：笔记的内容，形式为富文本（其实就是字符串）
  - `image`：上传者的头像
  - `parent`：笔记所属文件夹的_id，即假如一份笔记的目录属于文件夹“大雾”且“大雾”的_id为‘dawuid’，那么这份笔记的parent就为‘dawuid’
  - `publicity`：笔记是否公开，后续更新可以利用
  - `tags`：笔记的标签，可以作为搜索的关键字
  - `title`：笔记的标题，可以作为搜索的关键字

    - `type`：0表示该元素为主文件夹，1表示该元素为子文件夹，2表示该元素为笔记
- `USER`
    - 该集合记录了所有使用过小程序的用户的相关信息。当一个用户登录，小程序会检测用户是否为新用户，如果是，则为其在该集合中创建一个元素
    - `_id`：元素的唯一标识
    - `_openid`：用户的唯一标识
    - `favorlist`：用户的收藏列表，记录每个笔记的`_id`
- `chatroom`
    - 用来实现用户与客服的对话，方便用户提出增加笔记分类等请求
    - 记录聊天记录

模糊搜索

为了方便用户找到心仪的笔记，小程序需要有搜索功能。如果直接利用*Javascripts*提供的字符串匹配，那么对于字符串匹配程度的要求过高，便利性与实用性太差。查询相关资料后，我们通过构造正则表达式来实现对特定字段的模糊搜索。

要实现模糊搜索，我们需要借助微信小程序的数据库操作符，通过`db.command`获取。小程序中，我们利用了`db.command.or(..., ..., ...)`操作，其对括号的内容做或操作。

同时，我们还需要构造正则表达式，利用`db.RegExp`：

```
db.RegExp({
          regexp: '.*' + that.data.input,
          options: 'i'
        })
```

上例中，假设`that.data.input`中已经有了需要搜索的字段，`'.*'`表示单个字符匹配任意次；options的i表示执行对大小写不敏感的匹配。

由此，我们还需要确保搜出来的是分类过的（`parent`不为空字符串）笔记（`type==2`），在标题、标签、栏目中搜索有：

```
db.collection('NOTE').where(
      _.or([{
        tags: db.RegExp({
          regexp: '.*' + that.data.input,
          options: 'i'
        })
      },
      {
        title: db.RegExp({
          regexp: '.*' + that.data.input,
          options: 'i'
        })
      }, {
        column: db.RegExp({
          regexp: '.*' + that.data.input,
          options: 'i'
        })
      }])
    ).where({
      parent: _.neq(''),
```

```
      type: 2
    }).get({
      success: res => {
        console.log(res.data)
        that.setData({
          searchlist: res.data
        })
        console.log('get object', that.data.searchlist)
      }
    })
```

获取授权

根据微信小程序的最新要求，项目需要获取用户的授权。

通过部署调用云函数`login`自动获取当前用户的`_openid`并存入**app**的全局变量中，方便之后由用户对数据库进行操作。

```
const cloud = require('wx-server-sdk')

// 初始化 cloud
cloud.init({
  // API 调用都保持和云函数当前所在环境一致
  env: cloud.DYNAMIC_CURRENT_ENV
})

/**
 * 这个示例将经自动鉴权过的小程序用户 openid 返回给小程序端
 *
 * event 参数包含小程序端调用传入的 data
 *
 */
exports.main = async (event, context) => {
  console.log(event)
  console.log(context)

  // 可执行其他自定义逻辑
  // console.log 的内容可以在云开发云函数调用日志查看

  // 获取 WX Context (微信调用上下文)，包括 OPENID、APPID、及 UNIONID（需满足 UNIONID 获取条件）等信息
  const wxContext = cloud.getWXContext()

  return {
    event,
    openid: wxContext.OPENID,
    appid: wxContext.APPID,
    unionid: wxContext.UNIONID,
    env: wxContext.ENV,
  }
}
```

在**index**页面的`onload`的中会调用云函数`login`。

index页面的`onload`函数中会通过`getSetting`函数获取用户的相关信息：

```
wx.getSetting({
      success: res => {
        if (res.authSetting['scope.userInfo']) {
          // 已经授权，可以直接调用 getUserInfo 获取头像昵称，不会弹框
          wx.getUserInfo({
            success: res => {
              this.setData({
                avatarUrl: res.userInfo.avatarUrl,
                userInfo: res.userInfo
              })
              app.globalData.avatarUrl= res.userInfo.avatarUrl
              app.globalData.userInfo=res.userInfo
            }
          })
        }
      }
    })
```

调用该函数获取用户信息需要用户主动授权，因此我们需要对授权进行校验，因此在**app.js**的`onlaunch`函数中对该用户是否已经进行过授权进行判断。

```
wx.getSetting({
      success(res) {
        if (res.authSetting['scope.userInfo']) {
          wx.switchTab({
            url: '/pages/index/index',
          })
        }
      }
    })
```

如果该用户已经经过过授权，则直接进入**index**页面；如果该用户还未进行过授权，则进入授权页面。

授权页面中使用按钮来进行授权认证：

```
<button open-type="getUserInfo" bindgetuserinfo='getUser' class='button'
style="width: 200px;">获取用户信息(授权登录)</button>
```

获取Swiper点击事件

当用户点击**swiper**时，小程序需要知道用户点击的是哪一图片，并跳转到相应的笔记详情页。

首先，我们需要知道用户点击的是**swiper**的哪一面。查询开发者文档可知，页面切换时会自动触发`binchange`函数。因此我们先在**data**段设置一个`swipercur`的变量用来记录当前**swiper**展示的是哪一面。

调用**swiper**标签：

```
<swiper class="swiper" indicator-dots="true" autoplay="true" circular="true"
indicator-active-color="#eeeeee" indicator-color="#999999" current="
{{swipercur}}" bindchange="swiperchange" interval="3000">
```

然后在js文件里给`swipercur`赋值：

```
  swiperchange: function(e){
    this.setData({
      swipercur: e.detail.current
    })
  }
```

最后，如果用户点击了**swiper**的内容，则根据`swipercur`判断要进行何种处理：

```
  swipertap: function(){
    console.log(this.data.swipercur)
    if(this.data.swipercur==0){
      console.log('current == 0')
    }
    else if(this.data.swipercur==1){
      console.log('current == 1')
    }
    else if(this.data.swipercur==2){
      console.log('current == 2')
    }
  },
```

编辑器插入图片

为了满足笔记编辑的需求，编辑器需要有插入图片的功能。用户在编辑器中点击图片按钮（尝试插入图片）会触发`editor.js`中的`insertimage()`函数。

该函数首先会调用`wx.chooseImage`将用户选择的图片放入临时路径。

其后如果选择图片成功返回则会调用`wx.cloud.uploadFile()`函数将该函数上传到云存储中，并将该云存储的**fileID**插入到富文本中。

```
insertImage() { //图片上传插入示例
      var that = this;
      console.log(111)
      wx.chooseImage({
        count: 1,
        sizeType: ['original', 'compressed'],
        sourceType: ['album', 'camera'],
        success: (res) => {
          const tempFilePath = res.tempFilePaths[0]
          const cloudPath =  "img/" + new Date().getTime() +"-"+
Math.floor(Math.random() * 1000);

          wx.cloud.uploadFile({
            cloudPath: cloudPath,
            filePath: tempFilePath,
            success: res => {
              console.log('fileid', res.fileID);
              var src=res.fileID
              that.editorCtx.insertImage({
                src,
                data: {
```

```
                id: 'abcd',
                role: 'god'
              },
              width: '80%',
              success: function () {
                console.log('insert image success')
              }
            })
            console.log('fileid', res.fileID);
          }
        })
        // wx.uploadFile({ //调用图片上传接口
        //   url: 'http://192.168.1.61:3000/upload',
        //   filePath: tempFilePath,
        //   name: 'imgFile',
        //   success: res => {
        //     let imgUrl = JSON.parse(res.data).url
        //     console.log(imgUrl);
        //     that.editorCtx.insertImage({
        //       imgUrl,
        //       data: {
        //         id: 'abcd',
        //         role: 'god'
        //       },
        //       width: '80%',
        //       success: function () {
        //         console.log('insert image success')
        //       }
        //     })
        //   }
        // })
      },
      fail: (res) => {},
      complete: (res) => {},
    })

  }
```

构造`cloudPath`时需要使用时间和随机函数生成随机的文件名，否则会导致文件名重复，可能导致程序出错：

`const cloudPath =  "img/" + new Date().getTime() +"-"+ Math.floor(Math.random() * 1000);`

收藏功能

程序设置了收藏功能，方便用户集中查看重要的笔记。

该部分首先需要用到数据库中每个用户的`favorlist`。当用户点击收藏按钮时，首先利用用户的`_openid`获取该用户的信息。如果用户未收藏当前笔记，则加入该笔记的`_id`并更新其收藏列表；若已经收藏，则从列表中删去相应的笔记并更新收藏列表。

最后，刷新当前页面，使得收藏图片变为实心或空心。

```
collectOrNot: function () {
```

```
      console.log(this.data.isCollect)

    var that = this;
    var userfavorlist = [];
    const db = wx.cloud.database();
    // get favorlist
    db.collection('USER').where({
      _openid: app.globalData.openid
    }).get({
      success: res => {
        userfavorlist = res.data[0].favorlist
        if (that.data.isCollect) {
          userfavorlist.splice(userfavorlist.indexOf(that.data.decoded), 1);
        }
        else {
          userfavorlist.push(that.data.decoded);
        }

        // renew favorlist
        db.collection('USER').where({
          _openid: app.globalData.openid
        }).update({
          data: {
            favorlist: userfavorlist
          },
          success:res=>{
            wx.showToast({
              title: that.data.isCollect?'取消收藏':"已收藏",
              icon: 'success',
              duration: 1000,
              mask: true,
              success: function () {
                that.data.isCollect = that.data.isCollect ? 0 : 1;
                console.log(that.data.isCollect)
                that.reloadEvent();
              }
            })

          }
        })
      }
    })
  }
```

笔记分类

数据表`note`中共有三类记录，即：

- 笔记
- 子文件夹
- 主文件夹

由`type`字段进行区分：

- `type==0`：主文件夹，即所有科目归档的第一级文件夹

- `type==1`：子文件夹，即所有的次级文件夹(包括主文件夹的次级文件夹以及子文件夹的次级文件夹)
- `type==2`：笔记

子文件夹与笔记均拥有一个字段即`parent`用来存储包含该子文件夹或笔记的文件夹的`_id`。

在分类页面`notefilter`中，程序会首先拉取`type==0`的记录（所有的主文件夹）并显示，等待用户交互。用户点击一个文件夹后，程序获得该文件夹对应的`_id`，拉取数据库中`parent`字段等于`_id`的记录并显示。这时，程序会对拉取的记录的`type`进行判断，`type==1`则重用`curri`组件；`type==0`则重用`recommand`组件。

最终成果

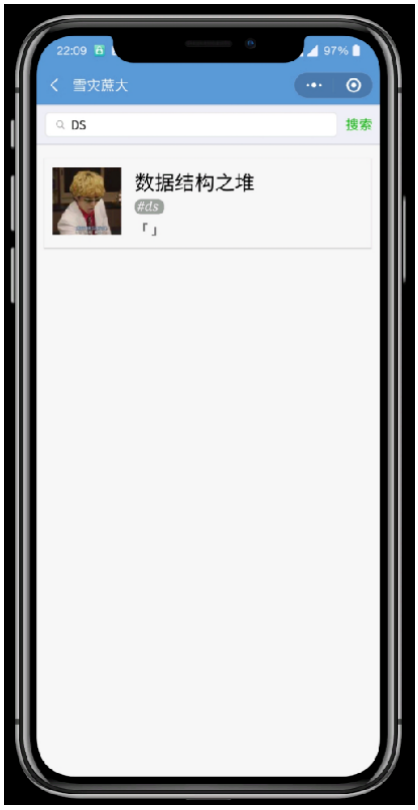
首页推荐


10:00
100%
雪灾盛大
请输入搜索内容
搜索
草稿
微观经济学
DS
大学物理实验
马克思主义基本原理概论
首页
分类
聊天室
个人中心

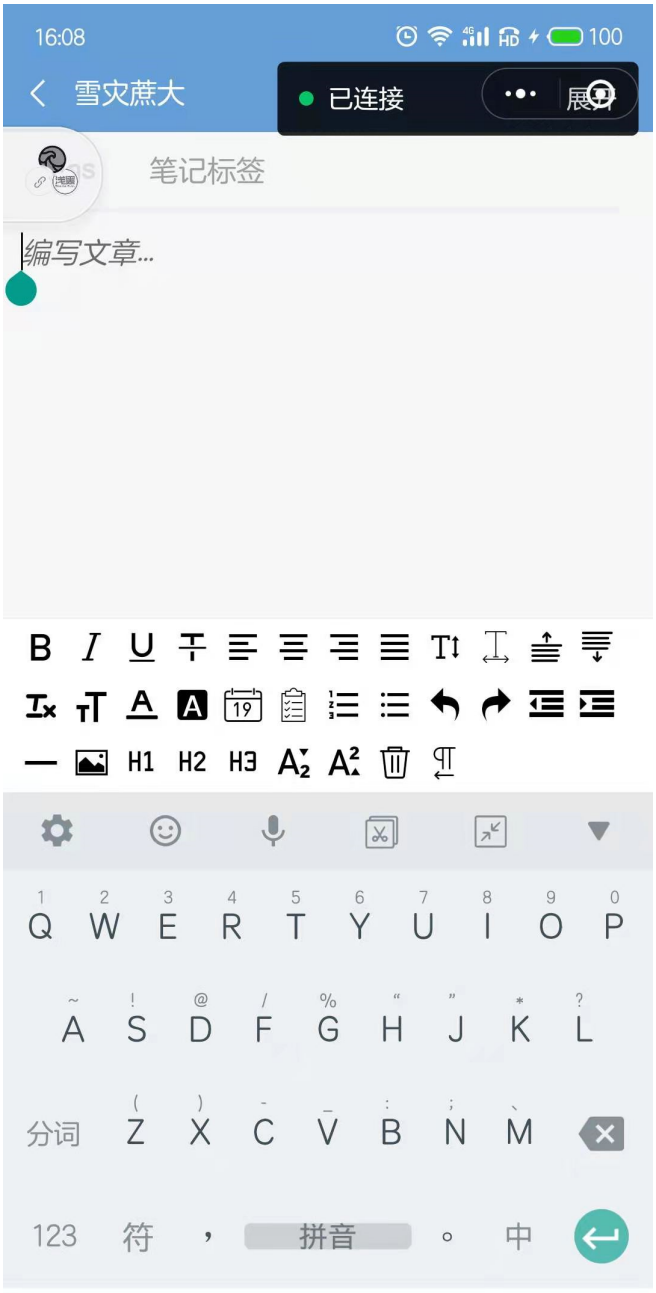
笔记分类


10:00
100%
雪灾盛大
如何撰写文献综述
沟通技巧
引言
（一）为什么要写综述：综述的目的
不管是想读研的同学，还是正在读研的同学，还是出了国的同学，读综述，写综述（Literature review），都是研究生涯中不可或缺的一环。很多人一开学，导师就会让你从写综述开始认识你的课题。
因为整个学术界的逻辑就是，任何发现都要建立在严谨的科学证据之上，任何新的发现，也只能建立在旧的发现的基础上。
回顾之前的研究，对任何一个新开始研究生涯的学术都尤为重要。而写综述，就是这样一个过程。

笔记详情



模糊搜索



笔记编辑

改进思路

- 模糊搜索栏中增加筛选条件，如将搜索范围限定在“大雾”课程。这一点可以通过增加一个`picker`标签，并相应地修改正则表达式来实现。
- 显示每一个笔记的热度，热度通过笔记被收藏的次数显示。这一点首先需要在数据库中为每条笔记增加一个记录被收藏次数的字段`favors`；然后，每当该笔记被收藏或被取消收藏，修改`favors`的值；最后，还需要相应地修改显示笔记的组件，让其显示该笔记被收藏的次数。
- 在新建笔记与编辑笔记处增加一个`picker`标签，方便用户设置笔记的公开性。
- 增设激励机制，激励用户产出有价值的笔记。

团队情况

团队共三人，分别为：

-
-
-

分工：

打分：