**CSI语音技术吧**

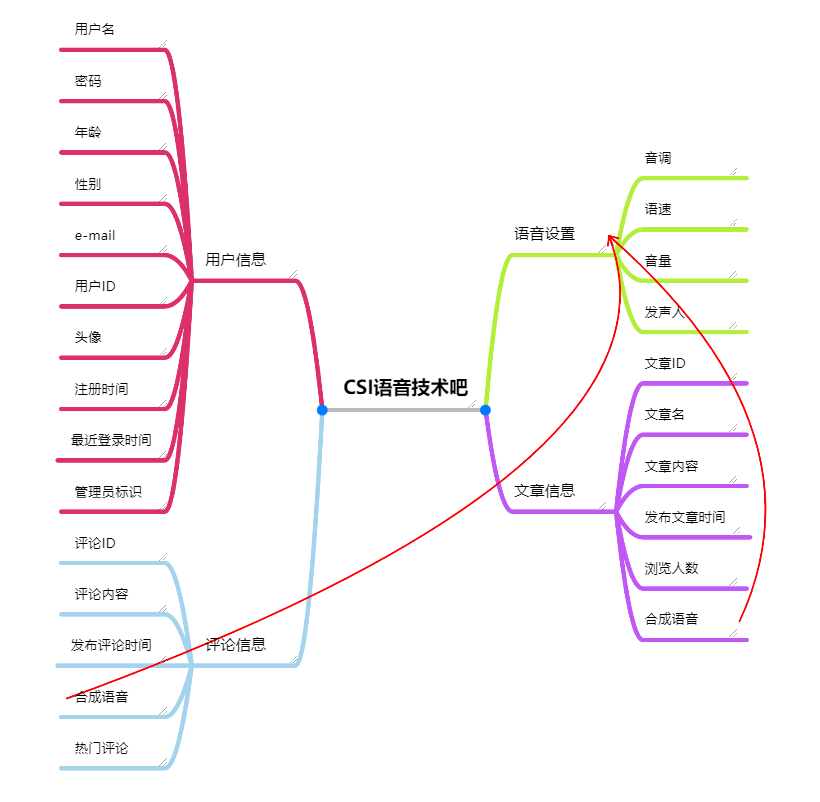
**开发设计文档**

**一、概述**

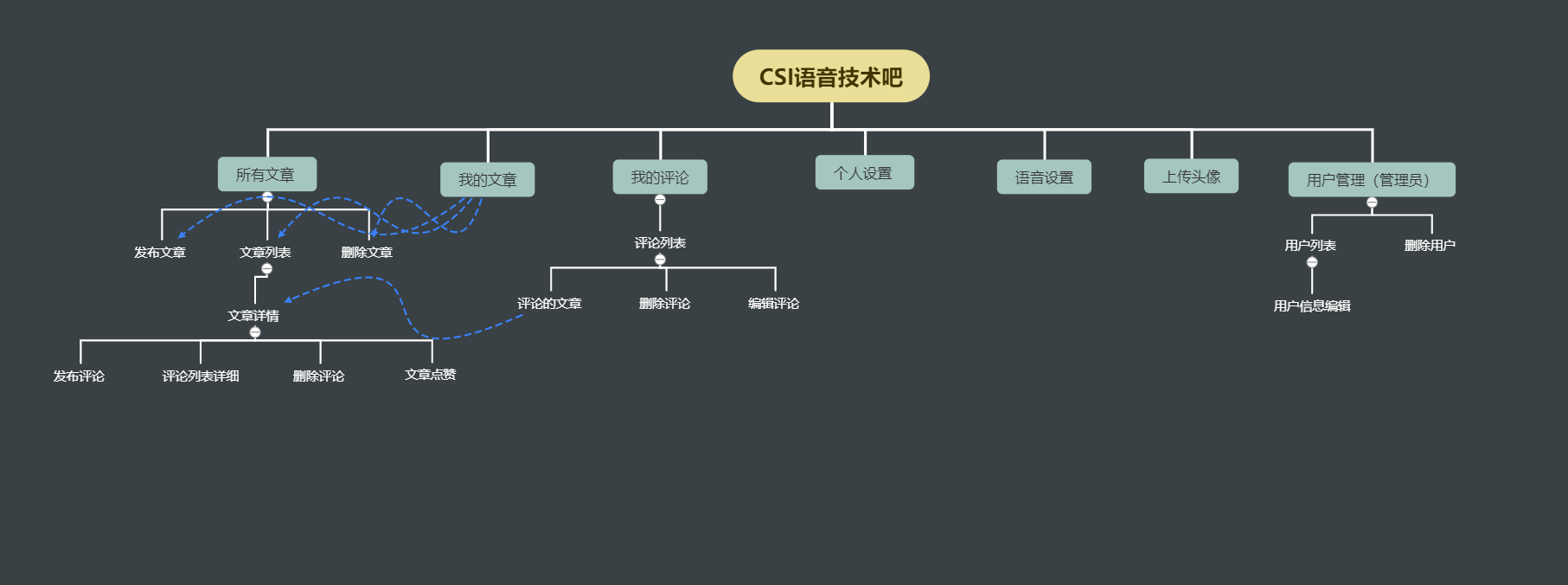
1.产品介绍

|  |  |
| --- | --- |
| 产品名称 | CSI语音技术吧 |
| 产品slogan | CSI语音技术吧，我们的精神角落 |
| 产品定位 | 一个在线交流平台，社区型信息沉淀方式，聚集志同道合的网友 |
| 产品描述 | CSI语音技术吧是内容创作者连接读者的工具，运营高品质论坛，为人们提供一个表达和交流思想的自由网络空间 |
| 关键词 | 贴吧、语音、交流 |
| 目标人群 | 有社交需求的用户、有兴趣爱好需求的用户、有问题需要解答的用户 |

1. 术语定义及缩略词含义
2. 语音合成：语音合成是通过机械的、电子的方法产生人造语音的技术，在本产品中把文章与文章评论合成为人造语音。
3. 系统/贴吧：若未特别指出，统指CSI语音技术吧。
4. **产品结构**
5. 信息结构图



1. 功能结构图



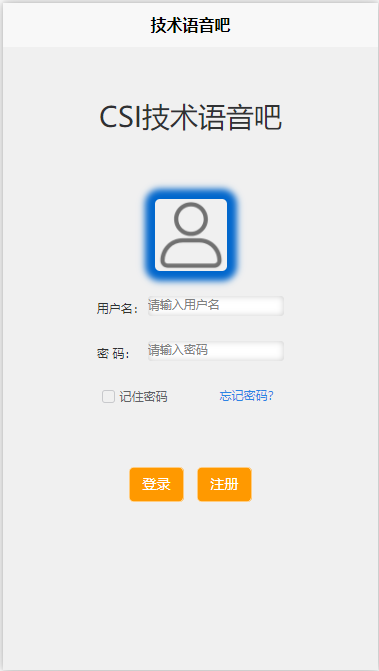
1. **用户角色说明**

|  |  |
| --- | --- |
| **系统角色** | **角色说明** |
| 普通用户 | * 可以访问操作所有文章模块（不能删除文章和评论） * 可以访问操作我的文章模块（不能删除单个评论） * 可以访问操作我的评论模块（进入所属文章界面后，不能删除文章和其他评论） * 可以访问操作语音设置模块 * 可以访问操作个人设置模块 * 可以访问操作上传头像模块 |
| 系统管理员 | * 可以访问操作所有文章模块 * 可以访问操作我的文章模块 * 可以访问操作我的评论模块 * 可以访问操作语音设置模块 * 可以访问操作个人设置模块 * 可以访问操作上传头像模块 * 可以访问操作用户管理模块 |

用户登录

原型：

登录：



注册：



忘记密码：



输入：

登录：用户名、密码、记住密码（可选）。

注册：用户名、密码、确认密码、性别、年龄、邮箱、用户权限（非管理员可不填）。

忘记密码：用户名、邮箱、新密码、确认密码。

输出及操作：

登录：

点击登录按钮，若填写正确，点击登录，进入“我的文章。”

若用户名和密码为空，输出“请先完善用户名和密码再登录”提示。

若输入密码长度不为6位，输出“您的密码格式不正确，请输入6位密码”提示。

若选中记住密码，下次登录密码自动填充。

点击注册按钮，进入注册页面。

点击忘记密码按钮，进入忘记密码页面。

注册：

点击提交按钮，若填写信息充分且格式无误，则弹窗确认，确认后注册新用户并返回登录界面。

若用户名、密码、确认密码、性别、年龄、邮箱任何一项为空，输出“请完善信息后再点击注册”提示。

若邮箱地址格式不正确，输出“您的邮箱格式不符合要求，请检查”提示。

若用户名已存在，输出“登录失败”。

若密码与确认密码不一致，输出“你的密码前后不一致”提示。

若密码长度不为6个字符，输出“您的密码格式不正确，请输入6位密码“提示。

点击取消，弹窗确认，确认后返回登录界面。

忘记密码：

点击提交按钮，若全部输入正确，点击确认按钮，弹窗确认，确认后修改用户密码，并返回到登录界面。

若用户名、邮箱、密码、确认密码等任何一项为空，应用均会提示“请完善信息后再点击注册”。

若邮箱地址格式不正确，输出“您的邮箱格式不符合要求，请检查“。

若密码与确认密码不一致，输出“你的密码前后不一致”。

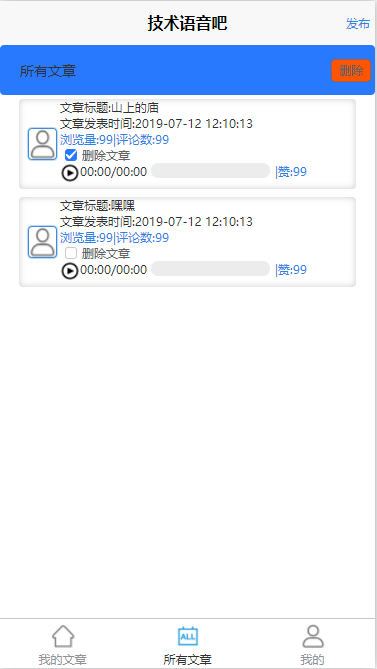
若密码长度不为6个字符，输出“您的密码格式不正确，请输入6位密码“。

点击取消按钮，弹窗确认，确认后返回登录界面

我的文章

原型：

文章列表：



新增文章：



文章详情：



编辑评论：



输入：

我的文章：要删除的文章（选择复选框，管理员权限）

新增文章：文章标题、文章正文

文章详情：要删除的评论（选择复选框，管理员权限）

添加评论：评论内容

输出及操作：

我的文章：

显示文章列表：展示当前用户发布的文章

查看作者信息：点击作者头像，跳转到我的基本信息界面

删除文章：点击勾选删除文章，再点击右上角删除按钮，弹窗确认，确认后删除勾选的所有文章。（管理员权限）。

播放音频：点击播放按钮，播放合成语音（不可拖动进度条）。

退出登录：点击退出按钮，退出到登录界面。

发布文章：点击发布按钮，进入到新增文章界面。

查看文章详情：点击文章卡片无其它交互功能的区域，进入该文章的详情页面。

新增文章：

点击取消按钮，弹窗并提醒取消后将会删除草稿，确认后删除草稿并返回到文章查阅页面

点击提交按钮，若文本为空，则输出“请先完善文章内容再提交”的提示。若文本不为空，弹窗确认，确认后发表文章，并返回到我的文章页面，并刷新文章列表

文章详情：

显示文章信息、5条热门评论、所有评论。

删除评论：点击勾选删除评论，再点击右上角删除按钮，弹窗确认，确认后删除勾选的所有文章。（管理员权限）。

播放音频：点击播放按钮，可以播放文章或评论的合成语音（不可拖动进度条）。

发表评论：点击文章卡片右下角的评论按钮，跳转到编辑评论界面

返回文章列表：点击左上角箭头按钮，返回文章列表。

编辑评论：

发表评论：点击提交，若文本为空，则输出“请先完善评论内容再提交”的提示。若文本不为空，则弹窗确认，确认后发表评论并返回到文章详情页面，并刷新评论列表。

取消评论：点击取消，确认退出编辑后，返回到文章详情页面。

所有文章

原型：

参见我的文章原型（区别在于，包含所有用户发表的文章）。

输入：

参见我的文章。

输出及操作：

参见我的文章（区别在于，仅管理员有权限删除文章，且点击其他用户的头像可以跳转对应的用户基本信息界面）。

语音设置

原型：



输入：性别（男/女）语速（1-5）、音量（1-5）、音调（1-5）。

输出及操作：

分别点击各选择栏，可相应选择各参数。

点击“提交”，弹出提示页面“确认要更改语音设置吗？”，点击确定更改语音设置并返回我的界面，否则取消。

点击取消：弹出提示页面“确认要取消并退出编辑吗”，点击确定按钮退出到我的界面

我的评论

原型：

我的评论：



编辑评论：参见我的文章→文章详情→编辑评论原型。



**输入：**

**我的评论：要删除的文章（选择复选框，管理员权限）。**

**编辑评论：参见我的文章→文章详情→编辑评论。**

**输出及操作：**

**我的评论：**

**点击头像可切换至“基本信息”页面。再点击左上角返回按钮可返回我的评论页面**

**点击每条评论的播放键，播放合成语音。**

**点击“作者“或”标题“可跳转”文章详情“，再点击左上角返回按钮可返回我的评论页面。**

**点击“点赞数“可实现点赞，点赞数增加。（目前暂不实现点赞这一功能）。**

**点击编辑评论可跳转至编辑评论页面，再点击左上角返回按钮可返回我的评论页面。**

**勾选删除评论，再点击右上方删除键可以删除评论，该操作不可逆。**

**用户管理（管理员权限）**

**原型：**

**用户管理：**

****

**编辑用户：**

****

**输入：**

**用户管理：**

**删除用户：点击勾选删除用户，再点击右上角删除按钮，弹窗确认，确认后删除勾选的所有文章。。**

**编辑用户：**

**用户名、密码、性别、年龄、邮箱、用户权限**

**输出和操作：**

**用户管理：**

**编辑用户：**

**提交编辑：点击提交按钮，若任意信息为空，输出“请先完善评论内容再提交”的提示。若密码长度不为6位，输出“您的密码格式不正确，请输入6位密码”的提示。若邮箱输入格式不正确，输出“您的邮箱格式不符合要求，请检查”的提示。若所填信息无异常，弹窗确认，确认后将修改该用户的信息。**

**取消编辑：点击取消按钮，弹窗确认，确认后返回用户管理界面。**

1. **接口**

1.user部分

接口：user/login

描述：用户登录

方法：POST

Request: userName str 用户名

password str 密码

Response: access\_token str

token\_type str

@router.post("/login", tags=["users"], response\_model=token)  
async def login(text: OAuth2PasswordRequestForm = Depends()):  
 userName = text.username  
 password = text.password  
 # loginTime = text.loginTime  
 credentials\_exception = HTTPException(  
 status\_code=status.HTTP\_401\_UNAUTHORIZED,  
 detail="Could not validate credentials",  
 headers={"WWW-Authenticate": "Bearer"},  
 )  
 result = userRetrieve.login(userName, password)  
 if not result:  
 raise credentials\_exception  
 expire = datetime.utcnow() + timedelta(minutes=ACCESS\_TOKEN\_EXPIRE\_MINUTES)  
 print('登录：', result['admin'], result['userName'])  
 payload = {'sub': result['userName'], 'exp': expire, 'admin': result['admin'], 'id': result['id']}  
 print('payload:', payload)  
 accessToken = jwt.encode(payload, SECRET\_KEY, algorithm=ALGORITHM)  
 return {'access\_token': accessToken, 'token\_type': 'bearer'}

接口：user/createUser

描述：用户注册

方法：POST

Request: userName str 用户名

password str 密码

regAge int 年龄

regSex str 性别

regEmail str 邮箱

managerKey str 管理员密钥

Response: code

message

data

@router.post("/createUser", tags=["users"])  
async def createUsers(text: createUser):  
 userName = text.userName  
 password = text.password  
 regAge = text.regAge  
 regSex = text.regSex  
 regEmail = text.regEmail  
 managerKey = text.managerKey  
 if managerKey == MANAGER\_KEY:  
 result = userCreate.userRegister(userName, password, regAge, regSex, regEmail, '1')  
 result = userCreate.userRegister(userName, password, regAge, regSex, regEmail, '0')  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "注册",  
 'data': result,  
 }  
 )

接口：user/forgetPassword

描述：找回密码

方法：POST

Request: userName str 用户名

regEmail str 邮箱

newPassword str 新密码

Response: code

message

data

@router.post("/forgetPassword", tags=["users"])  
async def forgetPassword(text: dropPassword):  
 userName = text.userName  
 regEmail = text.regEmail  
 newPassword = text.newPassword  
 result = userUpdate.updateUserPassword(userName, regEmail, newPassword)  
 print(result)  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "忘记密码",  
 'data': result,  
 }  
 )

接口：user/getUserInfor

描述：展示用户基本信息

方法：POST

Request: userName str 用户名

Response: code

message

data

@router.post("/getUserInfor", tags=["users"])  
async def showUserInfor(text: showUserInformation):  
 userName = text.userName  
 result = userRetrieve.getUserInfor(userName)  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "展示用户基本信息",  
 'data': result,  
 }  
 )

接口：user/updateUser

描述：修改用户基本信息

方法：POST

Request: oldName str 旧用户名

newName str 新用户名

regAge int 年龄

regSex str 性别

regEmail str 邮箱

Response: code

message

data

@router.post("/updateUser", tags=["users"])  
async def updateUser(text: updateUserInformation):  
 oldName = text.oldName  
 newName = text.newName  
 regAge = text.regAge  
 regSex = text.regSex  
 regEmail = text.regEmail  
 result = userUpdate.updateUserInfor(oldName, newName, regAge, regSex, regEmail)  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "修改基本信息",  
 'data': result,  
 }  
 )

接口：user/updatePassword

描述：修改密码

方法：POST

Request: userName str 用户名

oldPassword str 旧密码

newPassword str 新密码

Response: code

message

data

@router.post("/updatePassword", tags=["users"])  
async def updatePassword(text: userPassword):  
 userName = text.userName  
 oldPassword = text.oldPassword  
 newPassword = text.newPassword  
 result = userUpdate.updatePassword(userName, oldPassword, newPassword)  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "修改密码",  
 'data': result,  
 }  
 )

接口：user/getAllUserInfor

描述：得到用户列表

方法：POST

Request:

Response: code

message

data

@router.get("/getAllUserInfor", tags=["users"])  
async def getAllUser(token: str = Depends(getAdmin)):  
 result = userRetrieve.getAllUserInfor()  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "用户列表",  
 'data': result,  
 }  
 )

接口：user/deleteUser

描述：删除用户

方法：POST

Request: userName str 用户名

Response: code

message

data

@router.post("/deleteUser", tags=["users"])  
async def deleteUser(text: deleteUsers, token: str = Depends(getAdmin)):  
 userName = text.userName  
 result = userDelete.deleteUserInfor(userName)  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "删除用户",  
 'data': result,  
 }  
 )

接口：user/ManagerGetUserInfor

描述：向管理员展示用户所有基本信息

方法：POST

Request: userName str 用户名

Response: code

message

data

@router.post("/ManagerGetUserInfor", tags=["users"])  
async def showUserInfor(text: showUsers, token: str = Depends(getAdmin)):  
 userName = text.userName  
 result = userRetrieve.ManagerGetUserInfor(userName)  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "展示用户基本信息",  
 'data': result,  
 }  
 )

接口：user/editUserInfor

描述：管理员编辑用户

方法：POST

Request: newName str 新用户名

password str 密码

regSex str 性别

regAge int 年龄

regEmail str 邮箱

admin str 管理员标识

oldName str 旧用户名

Response: code

message

data

@router.post("/editUserInfor", tags=["users"])  
async def editUsers(text: editUser, token: str = Depends(getAdmin)):  
 newName = text.newName  
 password = text.password  
 regSex = text.regSex  
 regAge = text.regAge  
 regEmail = text.regEmail  
 admin = text.admin  
 oldName = text.oldName  
 result = userUpdate.ManagerEditUserInfor(newName, password, regSex, regAge, regEmail, admin, oldName)  
 return JSONResponse(  
 content={  
 'code': 200,  
 'message': "删除用户",  
 'data': result,  
 }  
 )

接口：/user/api/updatePhoto

描述：修改头像

方法：POST

Request: UploadFile

userID

2.item部分

接口：item/getMyArticle

描述：展示我的文章

方法：POST

Request: userName str 用户名

Response: postID int 文章id

postTitle str 文章标题

postTime str 发表时间

status bool 状态

@router.post("/getMyArticle", tags=["articles"])

async def getMyArticle(text:myArticle):

userName = text.userName

result = articleRetrieve.getArticle(userName)

return JSONResponse(

content={

'code':200,

'message':"我的文章",

'data':result,

}

)

接口：item/getAllarticle

描述：展示所有文章

方法：POST

Request: userName str 用户名

Response: postID int 文章id

postTitle str 文章标题

postTime str 发表时间

status bool 状态

@router.post("/getAllarticle", tags=["articles"])

async def getAllArticle(text:allArticle):

userName = text.userName

result = articleRetrieve.getAllArticle(userName)

return JSONResponse(

content={

'code':200,

'message':"所有文章",

'data':result,

}

)

接口：item/readArticle

描述：查看文章详情

方法：POST

Request: postID int 文章id

Response: postID int 文章id

postTitle str 文章标题

postAudio str 文章语音地址

postTime str 发表时间

status bool 状态

class article(BaseModel):

postID:int

@router.post("/readArticle", tags=["articles"])

async def readArticle(text:article):

postID = text.postID

result = articleRetrieve.readAnArticle(postID)

return JSONResponse(

content={

'code':200,

'message':"文章详情",

'data':result,

}

)

接口：item/postArticle

描述：发表文章

方法：POST

Request: userName str 用户名

postTitle str 文章标题

postText str 文章内容

postTime str 发表时间

Response: status bool 状态

postID int 文章id

postTitle str 文章标题

postAudio str 文章语音地址

status bool 状态

@router.post("/postArticle", tags=["articles"])

async def postArticle(text:postArticleInfor):

userName = text.userName

postTitle = text.postTitle

postText = text.postText

postTime = text.postTime

result = articleCreate.uploadArticle(userName,postTitle,postText,postTime)

print(result)

return JSONResponse(

content={

'code': 200,

'message' :"发布文章",

'data' : result,

}

)

接口：item/editArticle

描述：修改文章

方法：POST

Request: userName str 用户名

postTitle str 文章标题

postText str 文章内容

postID int 文章id

Response: status bool 状态

postID int 文章id

postTitle str 文章标题

postAudio str 文章语音地址

@router.post("/editArticle", tags=["articles"])

async def editArticle(text:updateArticles):

userName = text.userName

postID = text.postID

postTitle = text.postTitle

postText = text.postText

result = articleUpdate.updateArticle(userName,postID,postTitle,postText)

print(result)

return JSONResponse(

content={

'code': 200,

'message' :"修改文章",

'data' : result,

}

)

接口：item/deleteArticle

描述：删除文章

方法：POST

Request: userName str 用户名

postID int 文章id

admin str 管理员权限判断

Response: status bool 状态

@router.post("/deleteArticle", tags=["articles"])

async def deleteArticle(text:deleteArticles, token: str = Depends(getAdmin)):

postID = text.postID

userName = text.userName

admin = token

result = articleDelete.deleteArticle(postID, userName, admin)

print(result)

return JSONResponse(

content={

'code': 200,

'message' :"删除文章",

'data' : result,

}

)

接口：item/getMyComment

描述：展示我的评论

方法：POST

Request: userID int 用户id

Response: status bool 状态

cmID int 评论id

postID int 文章id

cmText str 评论内容

cmAudio str 评论语音地址

CmTime str 评论时间

@router.post("/getMyComment", tags=["comments"])

async def getMyComment(text:myComment):

userID = text.userID

result = commentRetrieve.getMyComments(userID)

print(result)

return JSONResponse(

content={

'code': 200,

'message' :"我的评论",

'data' : result,

}

)

接口：item/postComment

描述：发表评论

方法：POST

Request: userID int 用户id

postID int 文章id

cmText str 评论内容

cmTime str 评论时间

userName str 用户名

Response: status bool 状态

cmID int 评论id

cmText str 评论内容

cmAudio str 评论语音地址

cmTime str 评论时间

@router.post("/comment", tags=["comments"])

async def comment(text:postComment):

postID = text.postID

userID = text.userID

cmText = text.cmText

cmTime = text.cmTime

userName = text.userName

result = commentCreate.postComment(postID,userID,cmText,cmTime,userName)

print(result)

return JSONResponse(

content={

'code': 200,

'message' :"发表评论",

'data' : result,

}

)

接口：item/updateComment

描述：修改评论

方法：POST

Request: cmID int 评论id

cmText str 评论内容

userName str 用户名

Response: status bool 状态

cmID int 评论id

cmText str 评论内容

cmAudio str 评论语音地址

@router.post("/updateComment", tags=["comments"])

async def updateComment(text:changeComment):

cmID = text.cmID

cmText = text.cmText

userName = text.userName

result = commentUpdate.updateComments(cmID,cmText,userName)

print(result)

return JSONResponse(

content={

'code': 200,

'message' :"修改评论",

'data' : result,

}

)

接口：item/deleteComment

描述：删除

方法：POST

Request: cmID int 评论id

Response: status bool 状态

@router.post("/deleteComment", tags=["comments"])

async def deleteComments(text:deleteComment):

cmID = text.cmID

result = commentDelete.deleteComments(cmID)

print(result)

return JSONResponse(

content={

'code': 200,

'message' :"删除评论",

'data' : result,

}

)

接口：item/voiceSetting

描述：展示语音设置

方法：POST

Request: userName str 用户名

Response: status bool 状态

auSetSpd int 语速

auSetVol int 音量

auSetPit int 音调

auSetVoiPer int 发音人

@router.post("/voiceSetting", tags=["comments"])

async def voiceSetting(text:showVoiceSetting):

userName = text.userName

result = commentRetrieve.getVoiceSetting(userName)

return JSONResponse(

content={

'code': 200,

'message' :"展示用户语音设置",

'data' : result,

}

)

接口：item/setVoice

描述：设置语音设置

方法：POST

Request: userName str 用户名

auSetSpd int 语速

auSetVol int 音量

auSetPit int 音调

auSetVoiPer int 发音人

Response: status bool 状态

spd int 语速

vol int 音量

pit int 音调

per int 发音人

@router.post("/setVoice", tags=["articles"])

async def voiceSet(text:voice):

userName = text.userName

spd = text.auSetSpd

vol = text.auSetVol

pit = text.auSetPit

per = text.auSetVoiPer

result = commentUpdate.setVoice(userName,spd,vol,pit,per)

return JSONResponse(

content={

'code': 200,

'message' :"语音设置",

'data' : result,

}

)