



《基于大小模型协同的在线文 档编辑器系统》 详细设计说明书

V1.2.0

神奇三人组



版本历史

版本/状态	作者	参与者	日期	备注
1.0.0	神奇三人组	李建康、姬舜赢、陈怡	2024.6.27	创建
1.1.0	神奇三人组	李建康、姬舜赢、陈怡	2024.6.28	修改
1.2.0	神奇三人组	李建康、姬舜赢、陈怡	2024.6.28	完成



目录

第一部分 引言.....	5
一、编写目的.....	5
二、项目背景.....	5
三、定义.....	5
1. 技术类.....	5
2. 业务类.....	6
四、参考资料.....	6
第二部分 项目概述.....	7
一、项目简介.....	7
二、术语与缩写解释.....	8
1. RTE.....	8
2. LLM.....	9
3. RAG.....	9
4. OCR.....	9
5. ASR.....	10
第三部分 总体设计.....	10
一、技术架构设计.....	10
二、开发结构设计.....	11
三、物理架构设计.....	13
四、Web 环境下的集成配置.....	13
第四部分 界面设计.....	15
一、界面设计.....	15
1. 界面框架设计.....	16
1.1 登陆与注册界面.....	16
1.2 个人信息界面.....	16
1.3 文档空间界面.....	17
1.4 文档编辑界面.....	17
2. PC 界面设计.....	18
2.1 登陆界面.....	18
2.2 注册界面.....	18
2.3 个人信息界面.....	19
2.4 文档空间界面.....	19
2.5 文档编辑页面.....	20
第五部分 单元模块设计.....	20



一、模型层设计	20
1. 类图设计	20
2. 类的详细描述	21
第六部分 数据库设计	26
一、数据库整体结构图	26
二、用户功能	26
一、系统管理（审核相关）	27
二、后台功能（保存用户数据）	27
第七部分 补充设计和说明	28
一、编译运行环境设计	28
1. 数据库连接设计	28
二、包路径与 WEB 目录结构设计	29
1. vue 目录结构设计	29
2. Django 目录结构设计	30



第一部分 引言

一、编写目的

编写本设计的目的是为了准确阐述智能编辑器的具体实现思路和方法，即系统的详细架构和实现逻辑，主要包括程序系统的结构以及各层次中每个程序的设计考虑。预期读者为项目全体成员，包括运行维护和测试人员。

二、项目背景

- 系统名称：系统编辑器
 - 任务提出者：李建康。
 - 开发者：李建康、姬舜赢、陈怡。
- 用户和运行该程序系统的计算中心：略。

三、定义

1. 技术类

Django: Django 是一个高级 Python Web 框架，促进快速开发和简洁的设计。

1) Django 负责请求的转发和视图管理：

Views.py: 接收用户的请求并返回相应的响应。

Urls.py: URL 路由管理，将 URL 映射到相应的视图。

Templates: 管理 HTML 模板，负责渲染动态内容。

2) Django ORM 负责数据访问层，封装数据库操作：

Models.py: 定义数据模型，映射到数据库表。

QuerySets: 用于执行数据库查询。

Migrations: 管理数据库模式的变更。

Vue3: Vue.js 是一个渐进式 JavaScript 框架，用于构建用户界面。



1) Vue3 组件负责前端 UI，页面交互：

组件化开发：每个功能模块封装成一个组件，方便复用。

Vue Router：管理前端路由，实现单页面应用。

2) Tiptap 是一款富文本编辑器框架：

集成到 Vue 组件中：提供强大的文本编辑功能，支持多种格式和插件扩展。

MySQL：MySQL 是一个关系型数据库管理系统，负责数据持久化，完成数据库的 CRUD 操作。

数据库设计：定义数据库表及其关系。

SQL 查询：通过 Django ORM 执行查询和更新操作。

2. 业务类

LLM：大型语言模型（LLM）是一类基础模型，经过大量数据训练，使其能够理解和生成自然语言和其他类型的内容，以执行各种任务。本项目使用的大语言模型是由百度开发的文心一言（ERNIE Bot）模型。文心一言（ERNIE Bot）是百度全新一代知识增强大语言模型，文心大模型家族的新成员，能够与人对话互动，回答问题，协助创作，高效便捷地帮助人们获取信息、知识和灵感。文心一言是知识增强的大语言模型，基于飞桨深度学习平台和文心知识增强大模型，持续从海量数据和大规模知识中融合学习具备知识增强、检索增强和对话增强的技术特色。

四、参考资料

列出有关的参考资料，如：

1. 本项目的经核准的计划任务书或合同、上级机关的批文；

《技术服务合同》

《项目章程》

《里程碑计划》

《开发计划》



《交付物清单》

2. 属于本项目的其他已发布的文件；

《智能编辑器需求规格说明》

《智能编辑器项目概要设计》

3. 本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《python 风格规范——Google 开源项目风格指南》

第二部分 项目概述

一、项目简介

在数字化时代，传统的文档编辑工具往往只能处理单一模式的信息，如纯文本或简单的表格，这种方式在处理大量的多模态信息时显得效率低下，用户体验较差。无论是撰写项目书、设计分工表，还是记录会议纪要，用户在使用过程中常常感到力不从心。

目前，尽管一些文档编辑器已经引入了 AI 功能，但这些功能通常仅限于基础的纠错和简单的文本处理，无法满足用户对图片、表格、代码和语音视频等多模态信息的全面处理需求。这不仅导致了大量时间的浪费，也因为功能的局限性，用户在使用时的效率低、准确性差，甚至影响了工作和学习的质量。

随着信息内容和复杂度的增加，用户对文档编辑工具的需求也变得更加多样和苛刻。他们希望能够拥有一个强大的工具，不仅可以自动化地处理多种类型的信息，还能通过智能化的方式提升工作效率，优化用户体验。同时，随着团队协作和跨设备工作的增加，用户还需要工具在数据同步和安全性方面具备更高的能力。

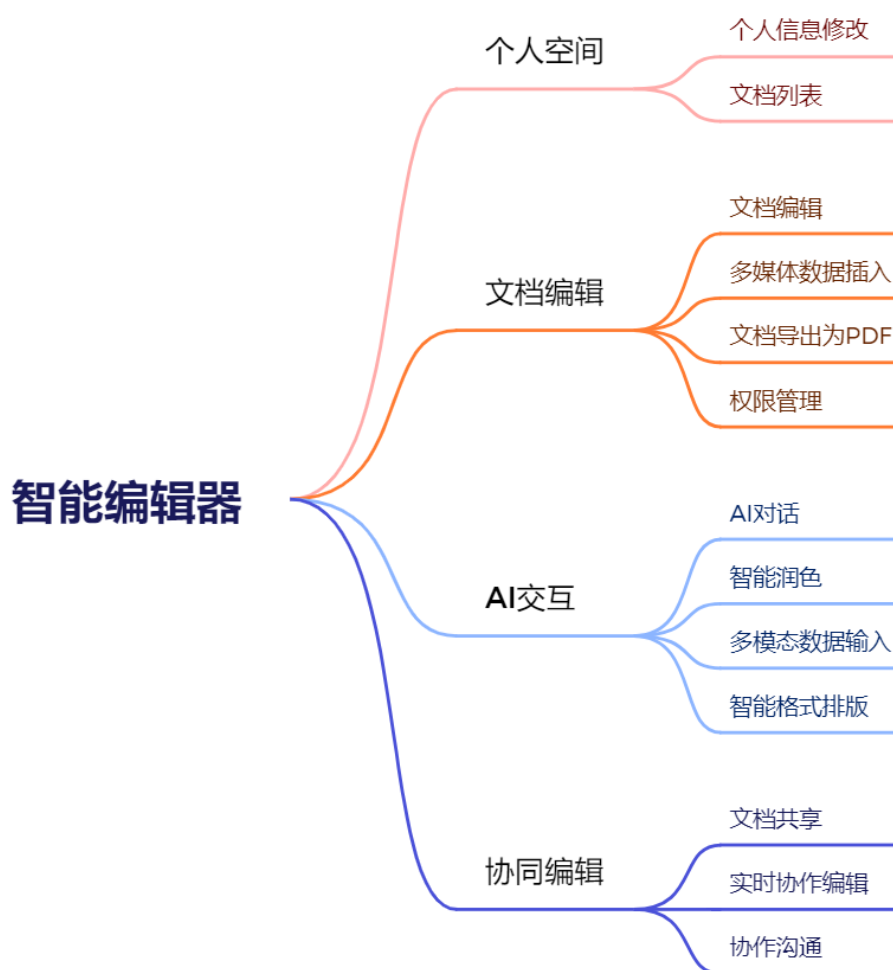
基于大小模型协同的在线编辑器将结合生成式 AI 技术和多模态信息处理能力，帮助用户高效地编辑和创作内容，还支持多人协作和云同步功能。



用户可以利用大模型实现对文章的智能排版，自动调整格式和布局，并通过多种模板选择丰富文档的展示效果。同时，该系统将支持多模态输入和理解，自动提取和组织结构化知识，构建用户专属的知识库，提升内容的丰富度和准确性。

系统主要包括个人空间、文档编辑、AI 交互、协同编辑四大部分。

功能模块图如下：



二、术语与缩写解释

1.RTE



富文本编辑器 (Rich Text Editor, RTE) 是一种可内嵌于浏览器, 所见即所得的在线文本编辑器。它提供类似于 Office Word 的编辑功能, 方便那些不太懂 HTML 用户使用, 富文本编辑器的应用非常广泛, 它的历史与图文网页诞生的历史几乎一样长。

2.LLM

大型语言模型 (LLM) 是一类基础模型, 经过大量数据训练, 使其能够理解和生成自然语言和其他类型的内容, 以执行各种任务。本项目使用的大语言模型是由百度开发的文心一言 (ERNIE Bot) 模型。文心一言 (ERNIE Bot) 是百度全新一代知识增强大语言模型, 文心大模型家族的新成员, 能够与人对话互动, 回答问题, 协助创作, 高效便捷地帮助人们获取信息、知识和灵感。文心一言是知识增强的大语言模型, 基于飞桨深度学习平台和文心知识增强大模型, 持续从海量数据和大规模知识中融合学习具备知识增强、检索增强和对话增强的技术特色。

3.RAG

大型语言模型 (LLM) 相较于传统的语言模型具有更强大的能力, 然而在某些情况下, 它们仍可能无法提供准确的答案。为了解决大型语言模型在生成文本时面临的一系列挑战, 提高模型的性能和输出质量, 研究人员提出了一种新的模型架构: 检索增强生成 (RAG, Retrieval-Augmented Generation)。该架构巧妙地整合了从庞大知识库中检索到的相关信息, 并以此为基础, 指导大型语言模型生成更为精准的答案, 从而显著提升了回答的准确性与深度。

4.OCR

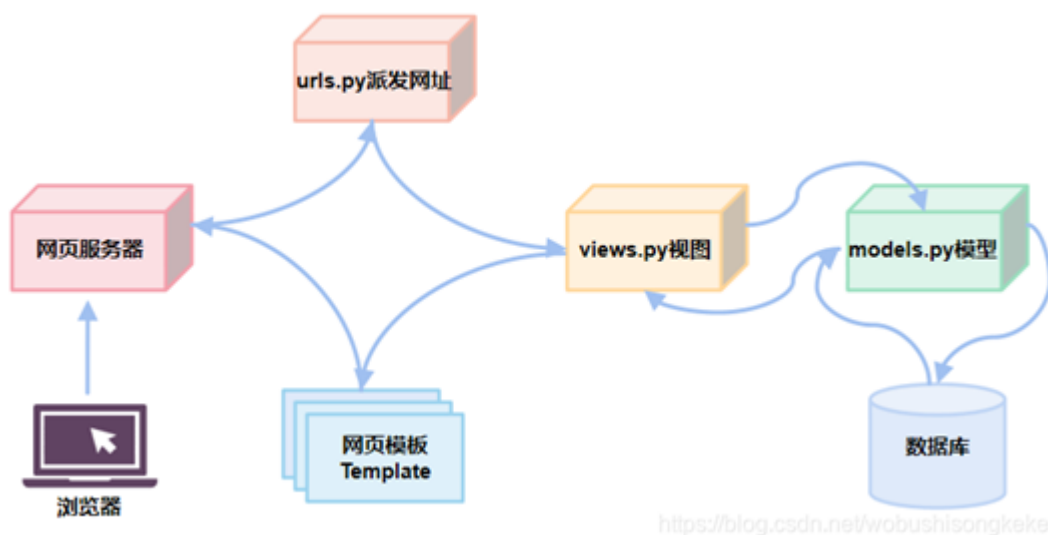
OCR 识别 (Optical Character Recognition, 光学字符识别) 是一种技术, 用于将印刷或手写文本从图像、扫描件或照片中提取并转换为可编辑和搜索的数字文本。该技术使用图像处理和模式识别算法, 将文本从静态的视觉形式转换为计算机可处理的数据, 从而实现自动化的数据输入和文档管理。OCR 识别广泛应用于文档数字化、数据录入自动化以及信息检索等领域。

5. ASR

语音识别也称为自动语音识别（ASR）、计算机语音识别或语音转文本，它能够让一个程序来处理将人类语音转变为书面格式的任务。语音识别专注于将语音从口头格式转换为文本，进而实现将文本输入到大模型中的功能。

第三部分 总体设计

一、技术架构设计



M——Model——models.py

Model 是 Django 表示数据的模式，以 Python 的类为基础在 models.py 中设置数据项与数据格式，基本上每个类对应一个数据库中的数据表。在后端 Django 会自动把这个类中的设置对应到数据库系统中，不管使用的是哪一种在数据库。因此，定义每个数据项时，除了数据项名称外，也要定义此项目的格式以及这张表和其他表格相互之间的关系（即数据关联）。定义完毕后，网站的其他程序就可以使用 Python 语句来操作这些数据内容，不用关心实际使用的 SQL 指令以及使用的是哪一种数据库。在 models.py 中定义所有需要用到的数据格式，一般是以数据库的形式来存储的，定义后的 Model 数据类要把它 import 到 views.py 中。



T——Template——template 文件夹

把取得的数据用美观且有弹性的方式输出，是在 Template 中处理。使用 templates 来做每个网页的外观框架，送至 template 中要被使用的数据尽量是可以直接显示的简单形式，不要试图在 template 文件中使用复杂的方法处理这些送进来的变量，如果需要对变量进行更复杂的运算，那么这些工作应该放在 views.py 中完成。

V——View——views.py

把数据取出来，或是如何存进去等程序逻辑，则是在 View 中，也就是在 view.py 中处理。View 是 Django 最重要的程序逻辑所在的地方，网站大部分程序设计都放在这里。这里放了许多要操作的数据，以及安排哪些数据需要被显示出来的函数，在函数中把这些数据传送给网页服务器或交由 Template 的渲染器后再送到网页服务器中。这些放在 views.py 中的函数，再由 urls.py 中的设计进行对应和派发。

在智能在线文档编辑器系统中，我们采用 Django 作为后端框架，负责数据管理和业务逻辑处理，并通过 Django 的 Model-View 架构定义数据模型和处理逻辑；前端采用 Vue 框架，负责构建用户界面和处理用户交互。通过这种前后端分离的设计，后端通过 API 接口（如 Django REST framework）向前端提供数据支持，前端通过 Vue 组件渲染和操作数据，实现高效、灵活的系统架构。

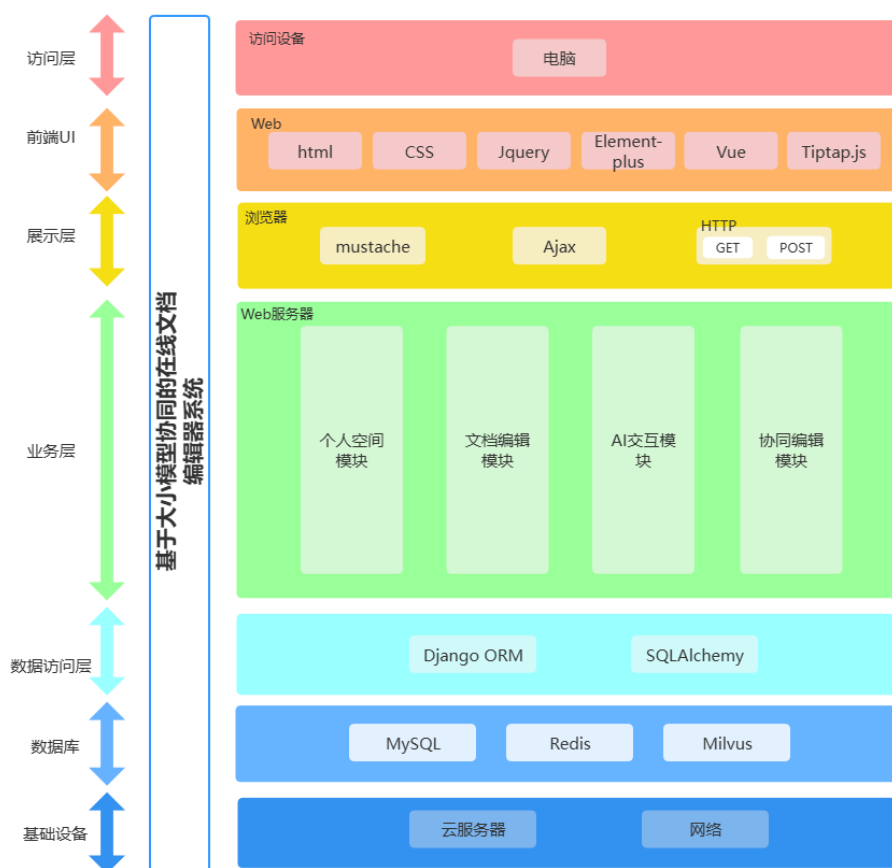
二、开发结构设计

本系统采用 Django 作为后端框架，Vue 作为前端框架，并使用 Tiptap.js 实现富文本编辑功能。前端通过 HTML、CSS、JQuery、Element-Plus、Vue 和 Tiptap.js 构建用户界面和交互逻辑，确保动态、高效的页面展示和丰富的编辑功能。后端 Django 负责数据管理和业务逻辑，通过 Django ORM 和 SQLAlchemy 进行数据的持久化操作。系统的数据存储采用 MySQL、Redis 和 Milvus，分别处理主要数据、缓存和 session 管理，以及向量数据的存储和相似度搜索。

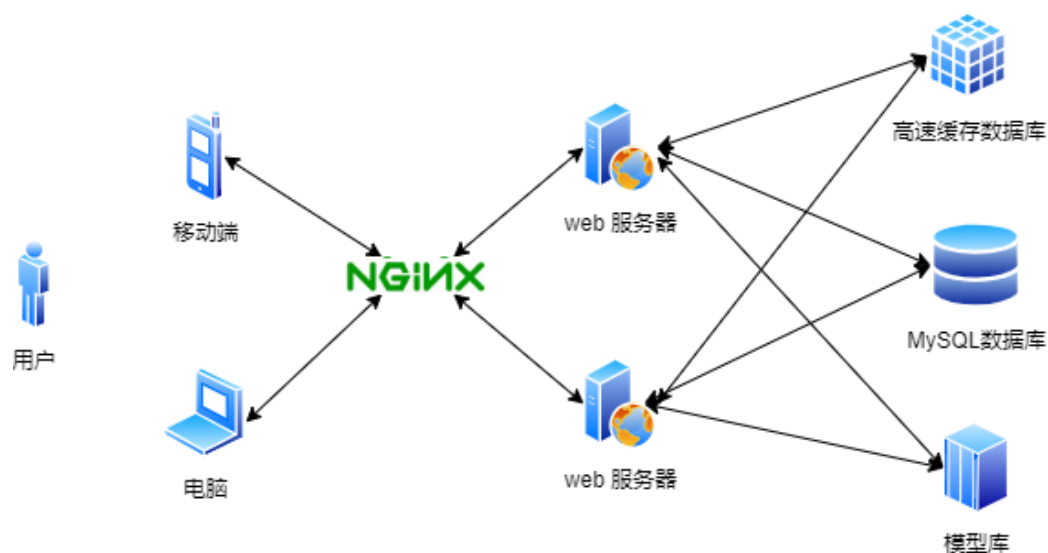
系统使用 Mustache 模板引擎进行数据渲染，通过 Ajax 进行异步数据请求和交互，并通过 HTTP 的 GET 和 POST 方法与后端进行通信。业务层包含个人空间模块、文档编辑模块、AI 交互模块和协同编辑模块，分别提供用户个人信息管理、文档创建和编辑、智能推荐和自动完成，以及多用户协同编辑功能。

为解决系统的并发问题和提高响应速度，系统使用 Nginx 负载均衡服务器，将 HTTP 请求分发到不同的 Web 服务器。此外，Redis 作为高速缓存服务器，不仅用于缓存和 session 管理，还解决了多个 Web 服务器之间 session 数据不共享的问题。系统部署在云服务器上，通过稳定的网络连接，确保系统的可扩展性、可访问性和高可靠性。通过这种前后端分离的设计，结合 Nginx 和 Redis 技术，系统形成了一种高效的开发模式，提升了开发效率和系统性能。

本项目基于 Django+vue3+Mysql 架构，进行了重新的分层，同时共享了业务代码。简化了开发、增强了与 Nginx 技术、Redis 技术的结合。提供了一种更高效的开发模式。



三、物理架构设计



四、Web 环境下的集成配置

1. Django 配置文件

```
1  """
2  Django settings for backend project.
3
4  Generated by 'django-admin startproject' using Django 5.0.6.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/5.0/topics/settings/
8
9  For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/5.0/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18 # Quick-start development settings - unsuitable for production
19 # See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
20
21 # SECURITY WARNING: keep the secret key used in production secret!
22 SECRET_KEY = "django-insecure-h4cd$esr*8il81*xcfg)b*4jba1j*+b!+d8x9vcdn)ln#jdo"
23
24 # SECURITY WARNING: don't run with debug turned on in production!
25 DEBUG = True
26
27 ALLOWED_HOSTS = []
28
29 # Application definition
30
31 INSTALLED_APPS = [
32     "django.contrib.admin",
33     "django.contrib.auth",
34     "django.contrib.contenttypes",
35     "django.contrib.sessions",
36     "django.contrib.messages",
37     "django.contrib.staticfiles",
38 ]
```



```
40     'corsheaders',
41 ]
42
43 MIDDLEWARE = [
44     "django.middleware.security.SecurityMiddleware",
45     "django.contrib.sessions.middleware.SessionMiddleware",
46     'corsheaders.middleware.CorsMiddleware',
47     "django.middleware.common.CommonMiddleware",
48     "django.middleware.csrf.CsrfViewMiddleware",
49     "django.contrib.auth.middleware.AuthenticationMiddleware",
50     "django.contrib.messages.middleware.MessageMiddleware",
51     "django.middleware.clickjacking.XFrameOptionsMiddleware",
52 ]
53
54 CORS_ALLOW_ALL_ORIGINS = True
55 ROOT_URLCONF = "backend.urls"
56
57 TEMPLATES = [
58     {
59         "BACKEND": "django.template.backends.django.DjangoTemplates",
60         "DIRS": [BASE_DIR / 'templates']
61         ,
62         "APP_DIRS": True,
63         "OPTIONS": {
64             "context_processors": [
65                 "django.template.context_processors.debug",
66                 "django.template.context_processors.request",
67                 "django.contrib.auth.context_processors.auth",
68                 "django.contrib.messages.context_processors.messages",
69             ],
70         },
71     },
72 ]
73
74 WSGI_APPLICATION = "backend.wsgi.application"
75
76
77 # Database
78 # https://docs.djangoproject.com/en/5.0/ref/settings/#databases
```

```
79
80 DATABASES = {
81     "default": {
82         "ENGINE": "django.db.backends.sqlite3",
83         "NAME": BASE_DIR / "db.sqlite3",
84     }
85 }
86
87
88 # Password validation
89 # https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators
90
91 AUTH_PASSWORD_VALIDATORS = [
92     {
93         "NAME": "django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
94     },
95     {
96         "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
97     },
98     {
99         "NAME": "django.contrib.auth.password_validation.CommonPasswordValidator",
100     },
101     {
102         "NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",
103     },
104 ]
105
106
107 # Internationalization
108 # https://docs.djangoproject.com/en/5.0/topics/i18n/
109
110 LANGUAGE_CODE = "en-us"
111
112 TIME_ZONE = "UTC"
113
114 USE_I18N = True
115
116 USE_TZ = True
117
```

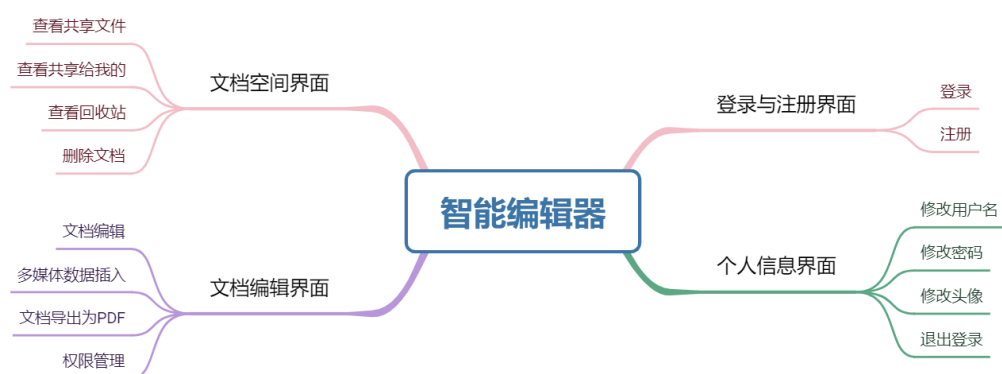
```
117  
118  
119 # Static files (CSS, JavaScript, Images)  
120 # https://docs.djangoproject.com/en/5.0/howto/static-files/  
121  
122 STATIC_URL = "static/"  
123  
124 # Default primary key field type  
125 # https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field  
126  
127 DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"
```

2.Vue 配置文件

```
1 import { defineConfig } from 'vite'  
2 import vue from '@vitejs/plugin-vue'  
3  
4 // https://vitejs.dev/config/  
5 无用法  CharonJay  
6 export default defineConfig( config: {  
7   plugins: [vue()],  
8 } )
```

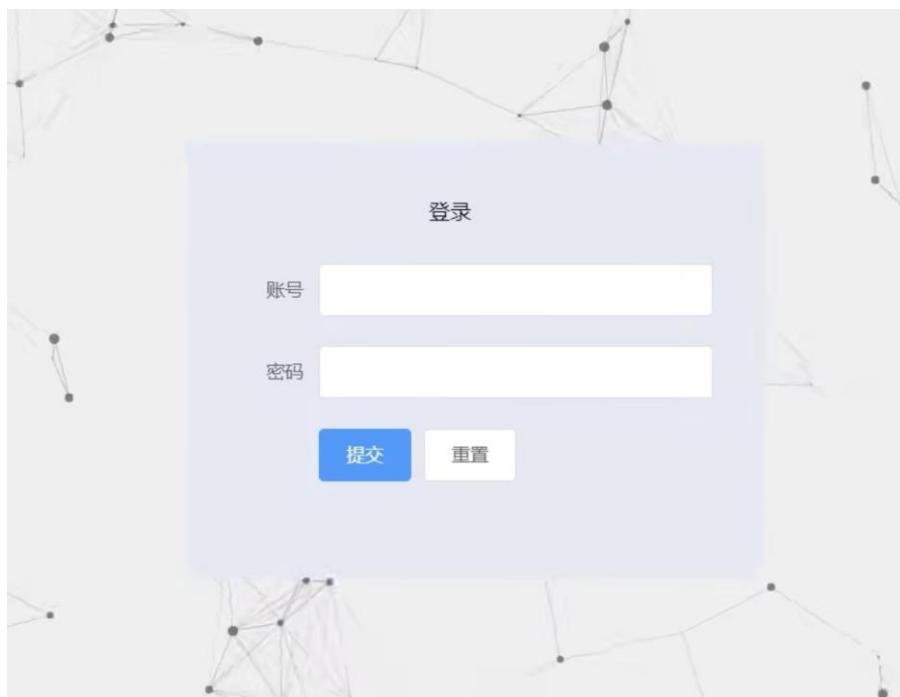
第四部分 界面设计

一、界面设计



1. 界面框架设计

1.1 登陆与注册界面



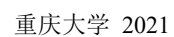
The image shows a login and registration interface. It features a light blue background with a network diagram pattern. A central white box contains the title "登录" (Login). Below the title are two input fields: "账号" (Account) and "密码" (Password). At the bottom of the box are two buttons: "提交" (Submit) in blue and "重置" (Reset) in white.

1.2 个人信息界面



The image shows a personal information interface. On the left is a circular profile picture placeholder labeled "个人头像" (Personal Avatar). To the right is a large blue rectangular area. At the top of this area is a table with two rows of information. Below the table is a large blue rectangular area labeled "基本信息区域" (Basic Information Area).

信息1	内容	修改
信息2	内容	修改



创建

文档列表

目录名称

Name	最近修改日期	文档信息
文档1		
文档2		



2. PC 界面设计

2.1 登陆界面

文心一编
WEN XIN EDITOR

文心一编
WEN XIN EDITOR

一款基于大小模型协同的智能文档编辑器

邮箱

密码

[重置密码](#)

[没有密码? 快速注册](#)

2.2 注册界面

文心一编
WEN XIN EDITOR

注册

邮箱

密码

确认密码

短信验证码

[返回登录界面](#)



2.3 个人信息界面

返回个人空间



一位酷炫的用户

文心一编
WEN XIN EDITOR

个人信息

头像



编辑

昵称

一位酷炫的用户

编辑

账号

12345678

邮箱

12345678@qq.com

更改

手机号

sxjy*****

更改

密码

188***8888

重置

2.4 文档空间界面

文心一编
WEN XIN EDITOR

搜索文档

一位酷炫的用户

创建文档

最近文件

共享给我

我的空间

回收站

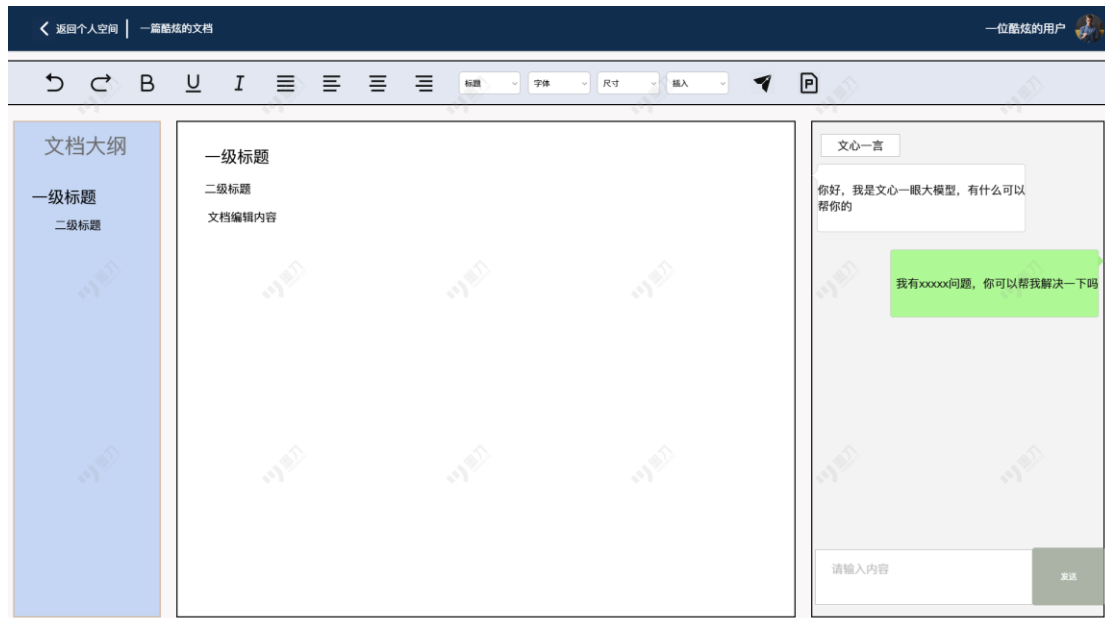
我的容量

100MB/1G

最近文档

文件名	创建者	打开时间
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前
test文档	一位酷炫的用户	1分钟前

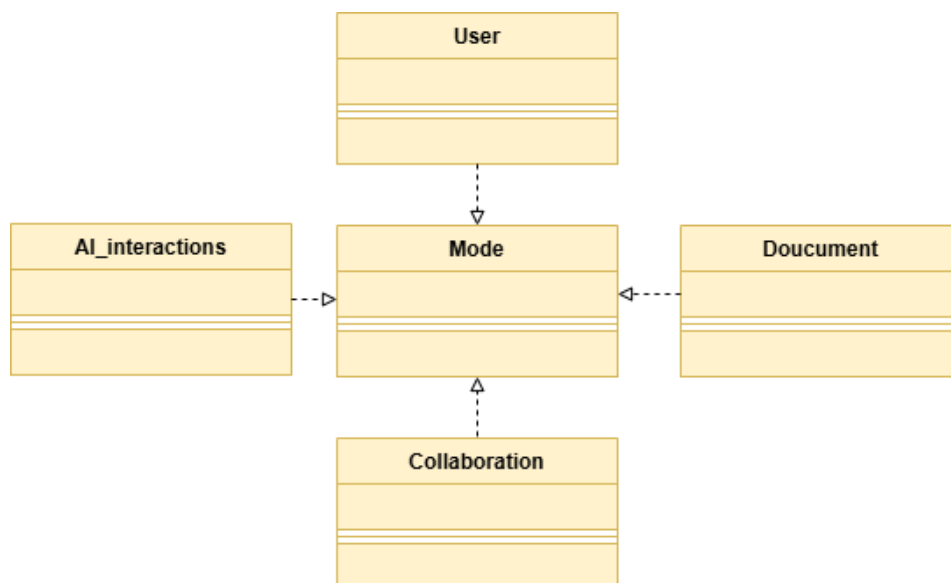
2.5 文档编辑页面



第五部分 单元模块设计

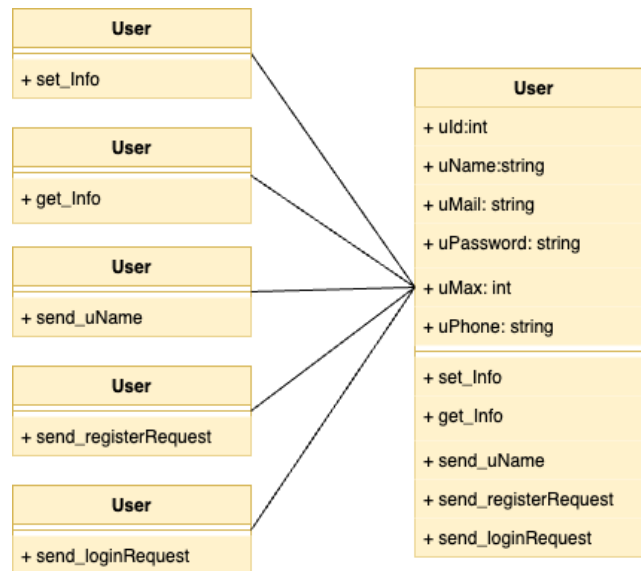
一、模型层设计

1. 类图设计



2. 类的详细描述

2.1 User 设计

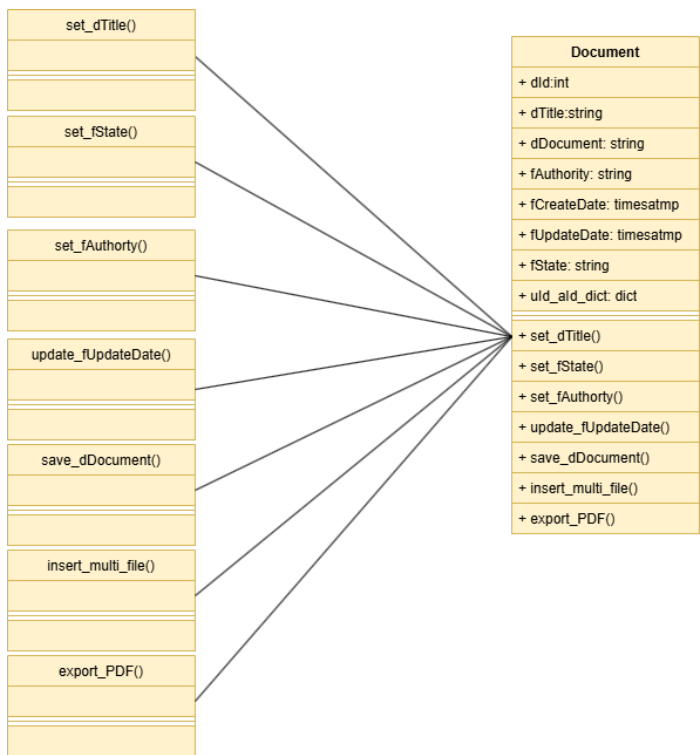


详细描述:

<pre>set_Info(uName:string, uPassword:string, **kwargs)</pre> <p>设置用户属性</p>
<pre>get_Info()->userInfo:dict</pre> <p>获取用户属性</p>
<pre>send_uName(uName:string)->response:int</pre> <p>单独发送用户名称</p>
<pre>send_registerRequest(uName:string, uPassword:string, uPhone, **kwargs)->response:int</pre> <p>发送注册请求</p>
<pre>send_loginRequest(uName:string, uPassword:string)->response:int</pre> <p>发送登陆请求</p>



2.2 Document 设计



详细描述:

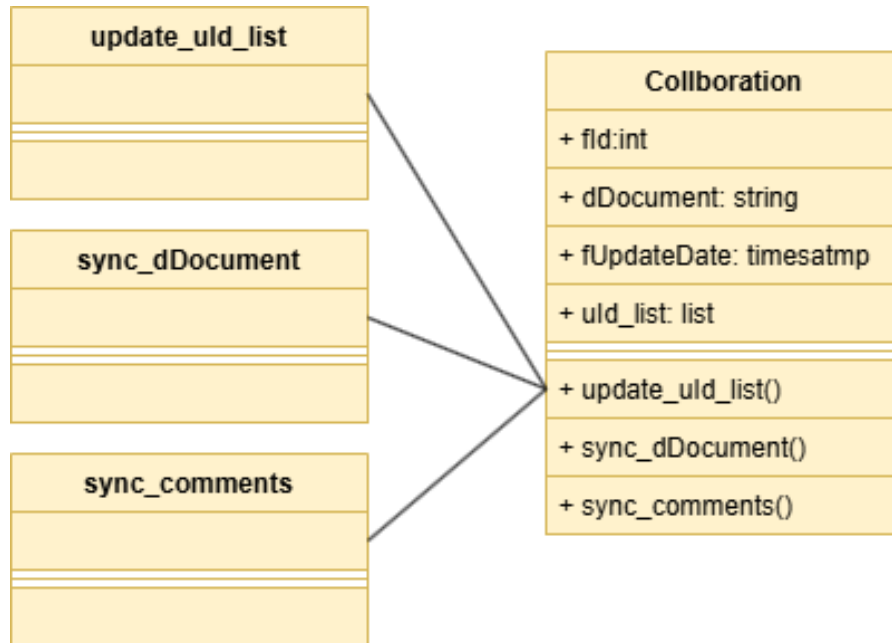
set_dTitle(dTitle)
设置或修改文档标题
set_dfState(fState)
设置文档状态（私人、协作、回收站）
set_fAuthorty(fState)
设置或修改文档编辑状态（可编辑、只读）
update_fUpdateDate(fUpdateDate)
更新文档最后保存时间
save_dDocument(dDocument)
保存文档内容
insert_multi_file(dDocument)
向文档中插入多媒体文件



export_PDF(dDocument)

将文档转化为 PDF 格式输出

2.3 Collobration 设计



详细描述:

update_uId_list(uId_list)

更新协作团队成员

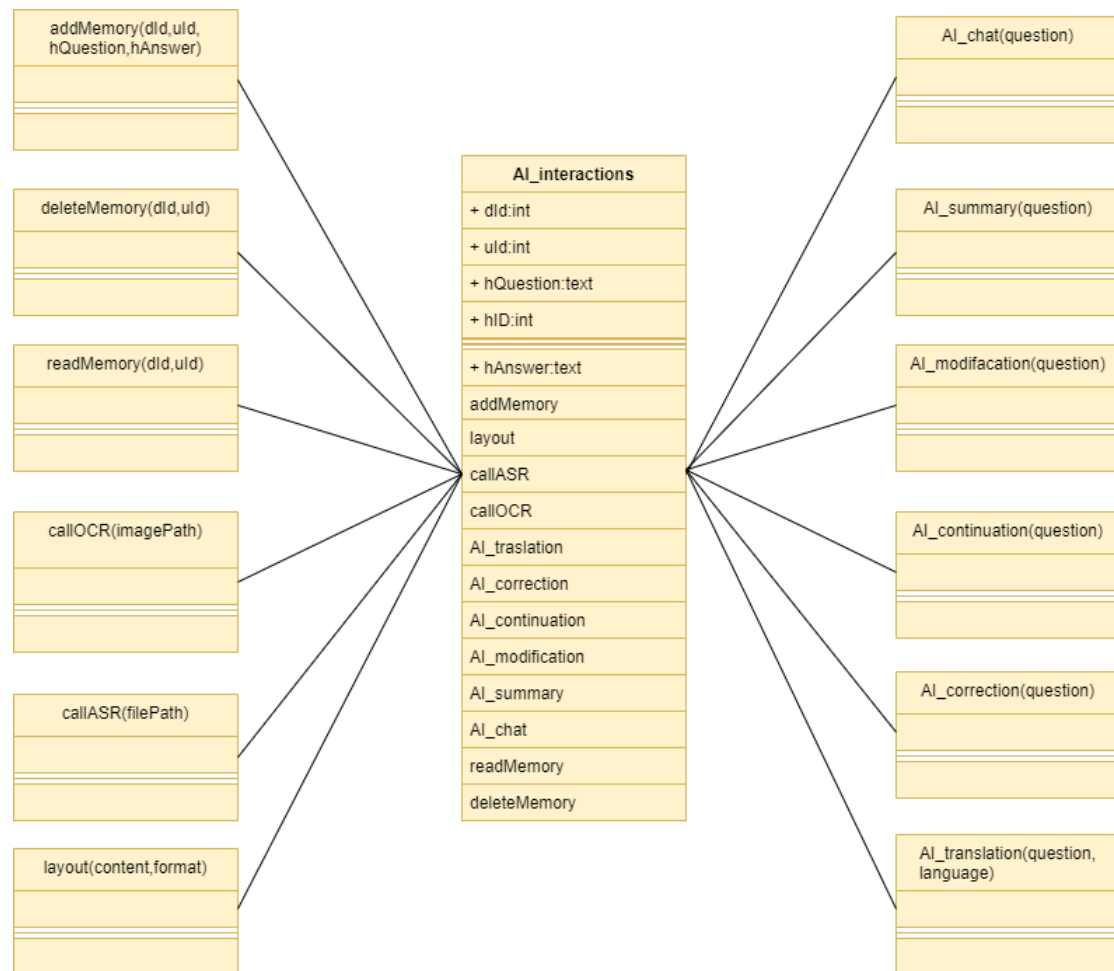
sync_dDocument(dDocument, fUpdateDate)

同步文档内容

sync_comments(dDocument, fUpdateDate)

同步评论和批注内容

2.4 AI_interactions 设计



详细描述:

<code>addMemory (dId, uId, hQuestion, hAnswer) ;</code> 保存一轮对话的问题和回答
<code>deleteMemory (dId, uId)</code> 删除 id 为 uId 的用户在 id 为 dId 的文档的对话记录
<code>readMemory (dId, uId) ;</code> 读取 id 为 uId 的用户在 id 为 dId 的文档的对话记录
<code>AI_chat (question)</code> 用户与 AI 的多轮对话
<code>AI_summary (question)</code>

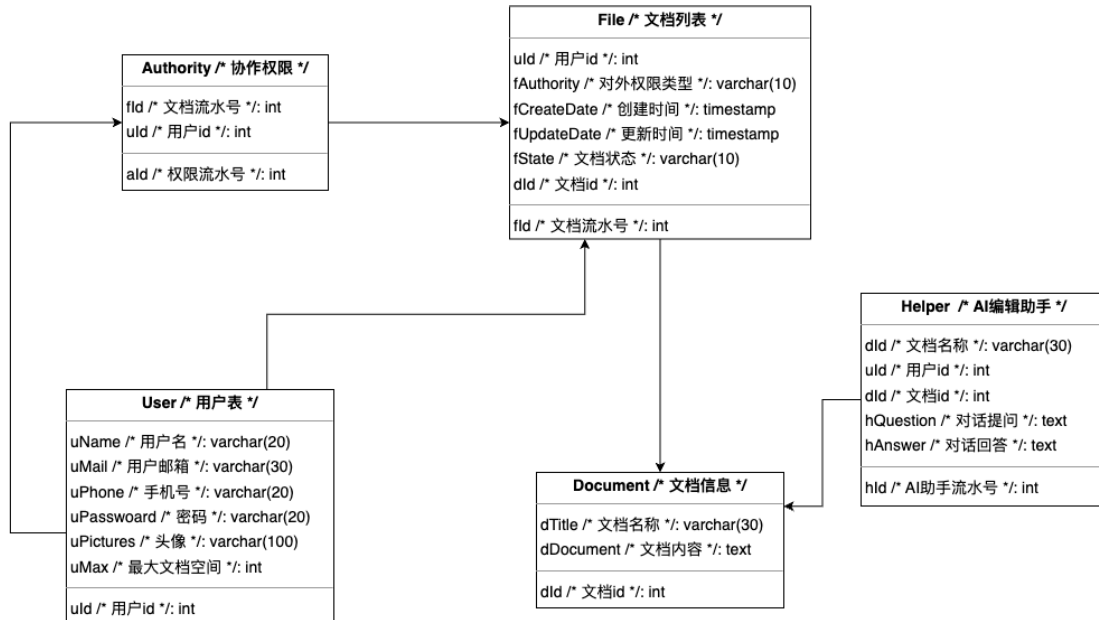


使用 AI 完成摘要功能
AI_modifacation(question) 使用 AI 完成修饰功能
AI_continuation(question) 使用 AI 完成续写功能
AI_correction(question) 使用 AI 修改病句
AI_translation(question, language) 使用 AI 完成翻译功能
callOCR(imagePath) 使用 OCR 模型提取图片、PDF 中的文字
callASR(filePath) 使用 ASR 提取视频音频中的文字信息
layout(content, format) 使用 AI 将文档内容排版为特定格式



第六部分 数据库设计

一、数据库整体结构图



二、用户功能

1. 文档信息表结构

序号	数据含义	存储名称	数据类型	宽度	精度	主键	外键	非空
1	文档 id	dId	int			TRUE	FALSE	TRUE
2	文档内容	dDocument	text			FALSE	FALSE	TRUE
3	文档名称	dTitle	varchar	30		FALSE	FALSE	TRUE

2. 协作权限表结构

序号	数据含义	存储名称	数据类型	宽度	精度	主键	外键	非空
1	权限流水号	aId	int			TRUE	FALSE	TRUE
2	用户 id	uId	int			FALSE	TRUE	TRUE
3	文档流水号	fId	int			FALSE	TRUE	TRUE



3. AI 编辑助手

序号	数据含义	存储名称	数据类型	宽度	精度	主键	外键	非空
1	AI 助手流水号	hId	int			TRUE	FALSE	TRUE
2	文本 id	dId	int			FALSE	TRUE	TRUE
3	用户 id	uId	int			FALSE	TRUE	TRUE
4	对话提问	hQuestion	text			FALSE	FALSE	TRUE
5	对话回答	hAnswer	text			FALSE	FALSE	TRUE

一、系统管理（审核相关）

1. 文档列表结构

序号	数据含义	存储名称	数据类型	宽度	精度	主键	外键	非空
1	文档流水号	fId	int			TRUE	FALSE	TRUE
2	用户 id	uId	int			FALSE	TRUE	TRUE
3	对外权限类型	fAuthority	varchar	10		FALSE	FALSE	TRUE
4	创建时间	fCreateDate	timesatmp			FALSE	FALSE	TRUE
5	更新时间	fUpdateDate	timesatmp			FALSE	FALSE	TRUE
6	文档状态	fState	varchar	10		FALSE	FALSE	TRUE
7	文档 id	dId	int			FALSE	TRUE	TRUE

二、后台功能（保存用户数据）

1. 用户表结构

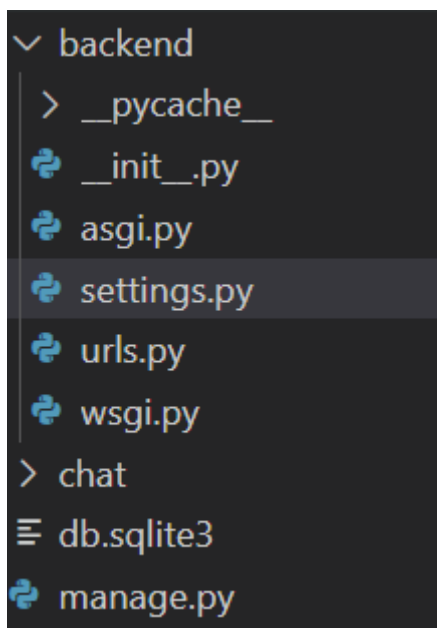


序号	数据含义	存储名称	数据类型	宽度	精度	主键	外键	非空
1	用户 id	uId	int			TRUE	TRUE	TRUE
2	用户名	uName	varchar	20		FALSE	FALSE	TRUE
3	用户邮箱	uMail	varchar	30		FALSE	FALSE	TRUE
4	手机号	uPhone	varchar	20		FALSE	FALSE	TRUE
5	密码	uPassword	varchar	20		FALSE	FALSE	TRUE
6	头像	uPictures	varchar	100		FALSE	FALSE	TRUE
7	最大文档空间	nMax	int			FALSE	FALSE	TRUE

第七部分 补充设计和说明

一、编译运行环境设计

1. 数据库连接设计



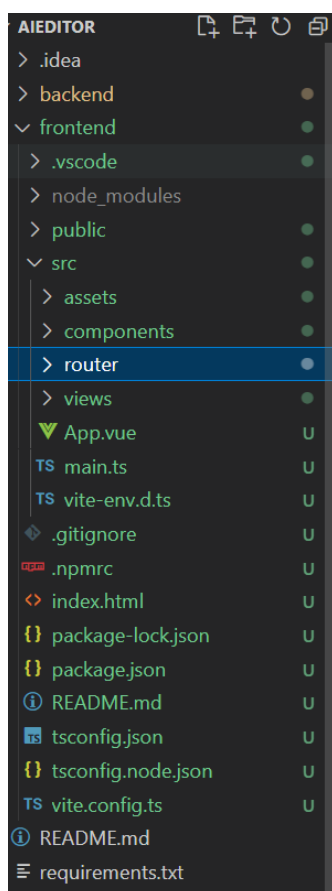
在 Django 项目中配置数据库，在 backend\setting.py 中加入如下内容：



```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'AIEditor',  
        'USER': 'eric',  
        'PASSWORD': '123123',  
        'HOST': '192.168.182.128',  
        'PORT': '3306',  
    }  
}
```

二、包路径与 WEB 目录结构设计

1. vue 目录结构设计



(1) AIEDITOR 为项目主包，frontend 为前端项目名称

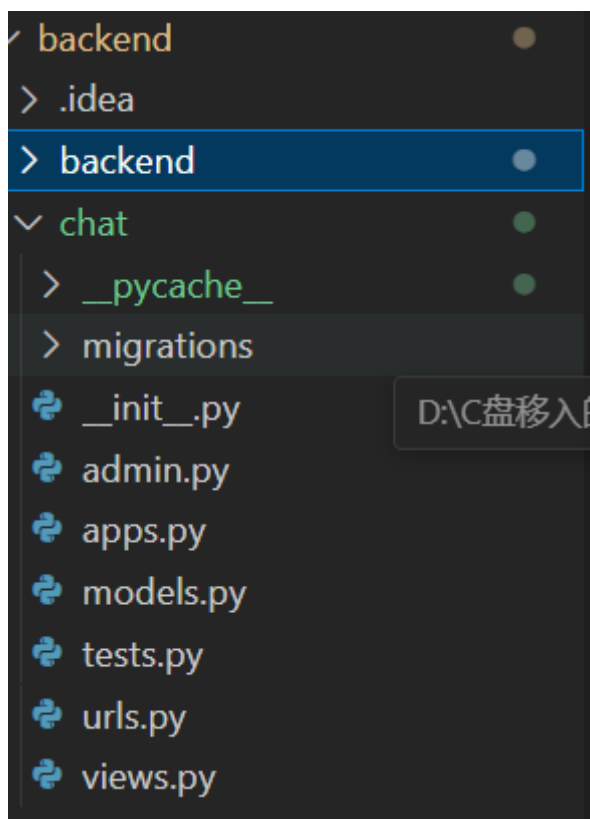
(2) 其下有 `node_modules`、`public`、`src` 文件夹

`node_modules` 这个文件夹包含了所有通过 `npm` 安装的项目依赖库和模块。`src` 文件夹是项目的主要代码库，包含所有的源代码。

(3) `src` 下有 `assets`、`components`、`router`、`views`。

`assets` 文件夹通常用于存放静态资源，例如图片、字体、样式文件等。它们可以在项目中通过相对路径或模块路径引用。`components` 文件夹包含 `Vue` 组件。组件是构建应用的基本单位，每个组件通常包括模板（HTML）、样式（CSS）、脚本（JavaScript）。组件可以是页面中的一个部分，也可以是一个完整的页面。`router` 文件夹包含路由配置文件。`Vue Router` 是 `Vue.js` 的官方路由管理器，用于在单页面应用中管理不同的视图。`router/index.js` 通常是路由的主配置文件。`views` 文件夹包含视图组件，这些组件是页面级的组件，用于定义应用的不同视图。

2. Django 目录结构设计





- (1) backend 为 Django 项目名。
- (2) 其下有项目目录 backend、应用目录 chat 等，还有 manage.py，这是一个命令行工具，通过这个脚本可以运行开发服务器、进行数据库迁移、创建应用等。
- (3) 每个 app 下，有应用的配置文件 apps.py，models.py 定义应用的数据模型、view.py 定义应用的视图函数或视图类，处理请求并返回响应。再建立 urls.py，建立 url 映射。