

**《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](#_Toc170485137)

[一、编写目的 5](#_Toc170485138)

[二、项目背景 5](#_Toc170485139)

[三、定义 5](#_Toc170485140)

[1.技术类 5](#_Toc170485141)

[2.业务类 6](#_Toc170485142)

[四、参考资料 6](#_Toc170485143)

[第二部分 项目概述 7](#_Toc170485144)

[一、项目简介 7](#_Toc170485145)

[二、术语与缩写解释 8](#_Toc170485146)

[1.RTE 8](#_Toc170485147)

[2.LLM 9](#_Toc170485148)

[3.RAG 9](#_Toc170485149)

[4.OCR 9](#_Toc170485150)

[5.ASR 10](#_Toc170485151)

[第三部分 总体设计 10](#_Toc170485152)

[一、技术架构设计 10](#_Toc170485153)

[二、开发结构设计 11](#_Toc170485154)

[三、物理架构设计 13](#_Toc170485155)

[四、Web环境下的集成配置 13](#_Toc170485156)

[第四部分 界面设计 15](#_Toc170485157)

[一、界面设计 15](#_Toc170485158)

[1.界面框架设计 16](#_Toc170485159)

[1.1登陆与注册界面 16](#_Toc170485160)

[1.2个人信息界面 16](#_Toc170485161)

[1.3文档空间界面 17](#_Toc170485162)

[1.4文档编辑界面 17](#_Toc170485163)

[2.PC界面设计 18](#_Toc170485164)

[2.1登陆界面 18](#_Toc170485165)

[2.2注册界面 18](#_Toc170485166)

[2.3个人信息界面 19](#_Toc170485167)

[2.4文档空间界面 19](#_Toc170485168)

[2.5文档编辑页面 20](#_Toc170485169)

[第五部分 单元模块设计 20](#_Toc170485170)

[一、模型层设计 20](#_Toc170485171)

[1.类图设计 20](#_Toc170485172)

[2.类的详细设计描述 21](#_Toc170485173)

[第六部分 数据库设计 26](#_Toc170485174)

[一、数据库整体结构图 26](#_Toc170485175)

[二、用户功能 26](#_Toc170485176)

[一、系统管理（审核相关） 27](#_Toc170485177)

[二、后台功能（保存用户数据） 27](#_Toc170485178)

[第七部分 补充设计和说明 28](#_Toc170485179)

[一、编译运行环境设计 28](#_Toc170485180)

[1.数据库连接设计 28](#_Toc170485181)

[二、包路径与WEB目录结构设计 29](#_Toc170485182)

[1.vue目录结构设计 29](#_Toc170485183)

[2.Django目录结构设计 30](#_Toc170485184)

### 引言

#### 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。

#### 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。

#### 定义

##### 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。

1. **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。

1. **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。

##### 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。

#### 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》

1. 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》

1. 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》

### 项目概述

#### 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。

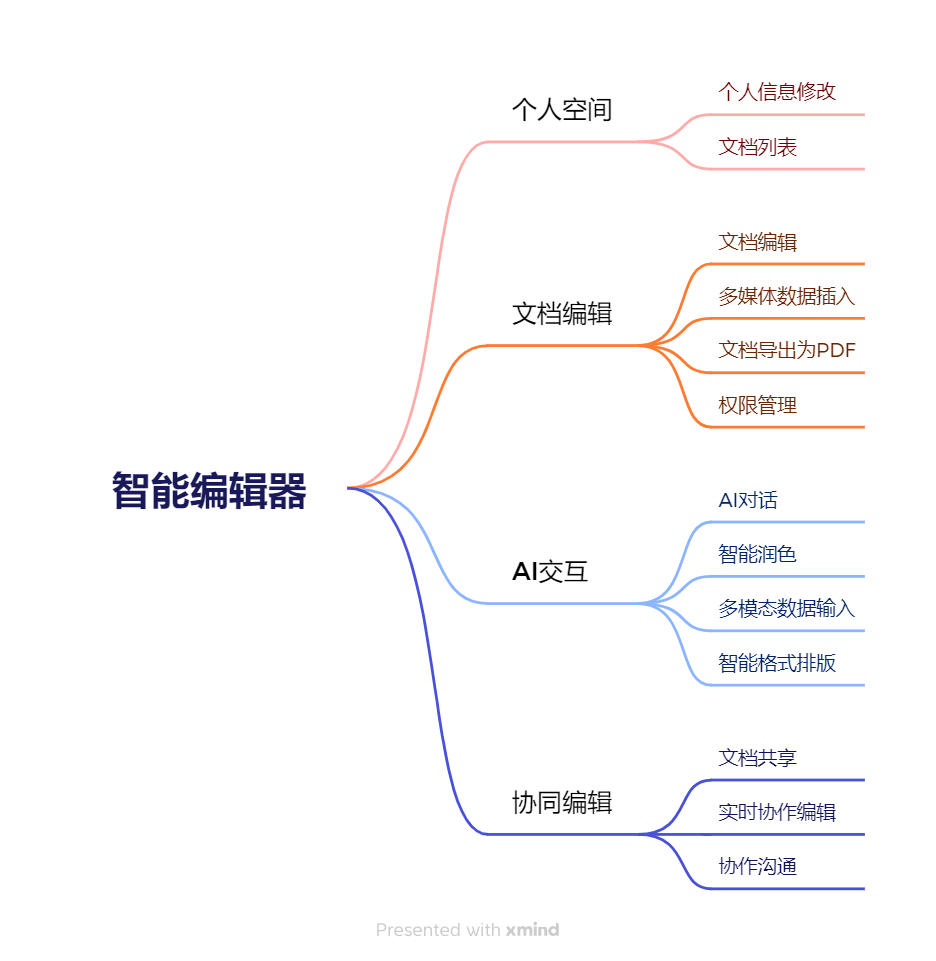
目前，尽管一些文档编辑器已经引入了AI功能，但这些功能通常仅限于基础的纠错和简单的文本处理，无法满足用户对图片、表格、代码和语音视频等多模态信息的全面处理需求。这不仅导致了大量时间的浪费，也因为功能的局限性，用户在使用时的效率低、准确性差，甚至影响了工作和学习的质量。

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：

#### 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

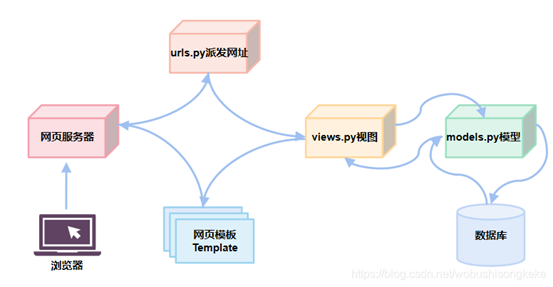
OCR识别（Optical Character Recognition，光学字符识别）是一种技术，用于将印刷或手写文本从图像、扫描件或照片中提取并转换为可编辑和搜索的数字文本。该技术使用图像处理和模式识别算法，将文本从静态的视觉形式转换为计算机可处理的数据，从而实现自动化的数据输入和文档管理。OCR识别广泛应用于文档数字化、数据录入自动化以及信息检索等领域。

##### 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。

### 总体设计

#### 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。

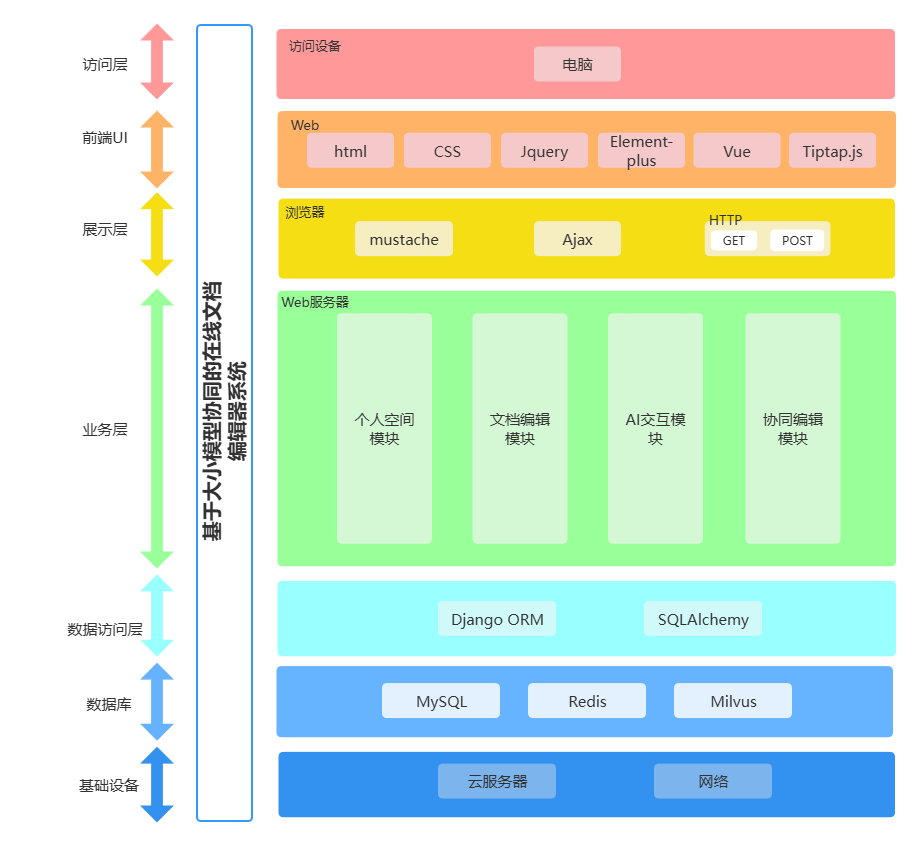
在智能在线文档编辑器系统中，我们采用Django作为后端框架，负责数据管理和业务逻辑处理，并通过Django的Model-View架构定义数据模型和处理逻辑；前端采用Vue框架，负责构建用户界面和处理用户交互。通过这种前后端分离的设计，后端通过API接口（如Django REST framework）向前端提供数据支持，前端通过Vue组件渲染和操作数据，实现高效、灵活的系统架构。

#### 开发结构设计

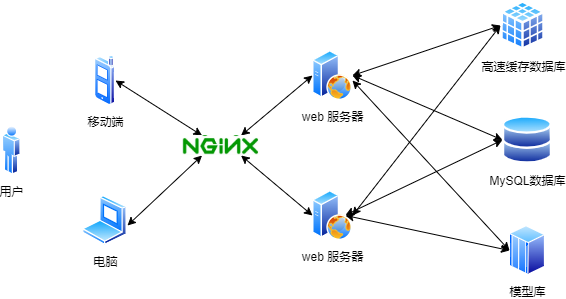
本系统采用Django作为后端框架，Vue作为前端框架，并使用Tiptap.js实现富文本编辑功能。前端通过HTML、CSS、JQuery、Element-Plus、Vue和Tiptap.js构建用户界面和交互逻辑，确保动态、高效的页面展示和丰富的编辑功能。后端Django负责数据管理和业务逻辑，通过Django ORM和SQLAlchemy进行数据的持久化操作。系统的数据存储采用MySQL、Redis和Milvus，分别处理主要数据、缓存和session管理，以及向量数据的存储和相似度搜索。

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。

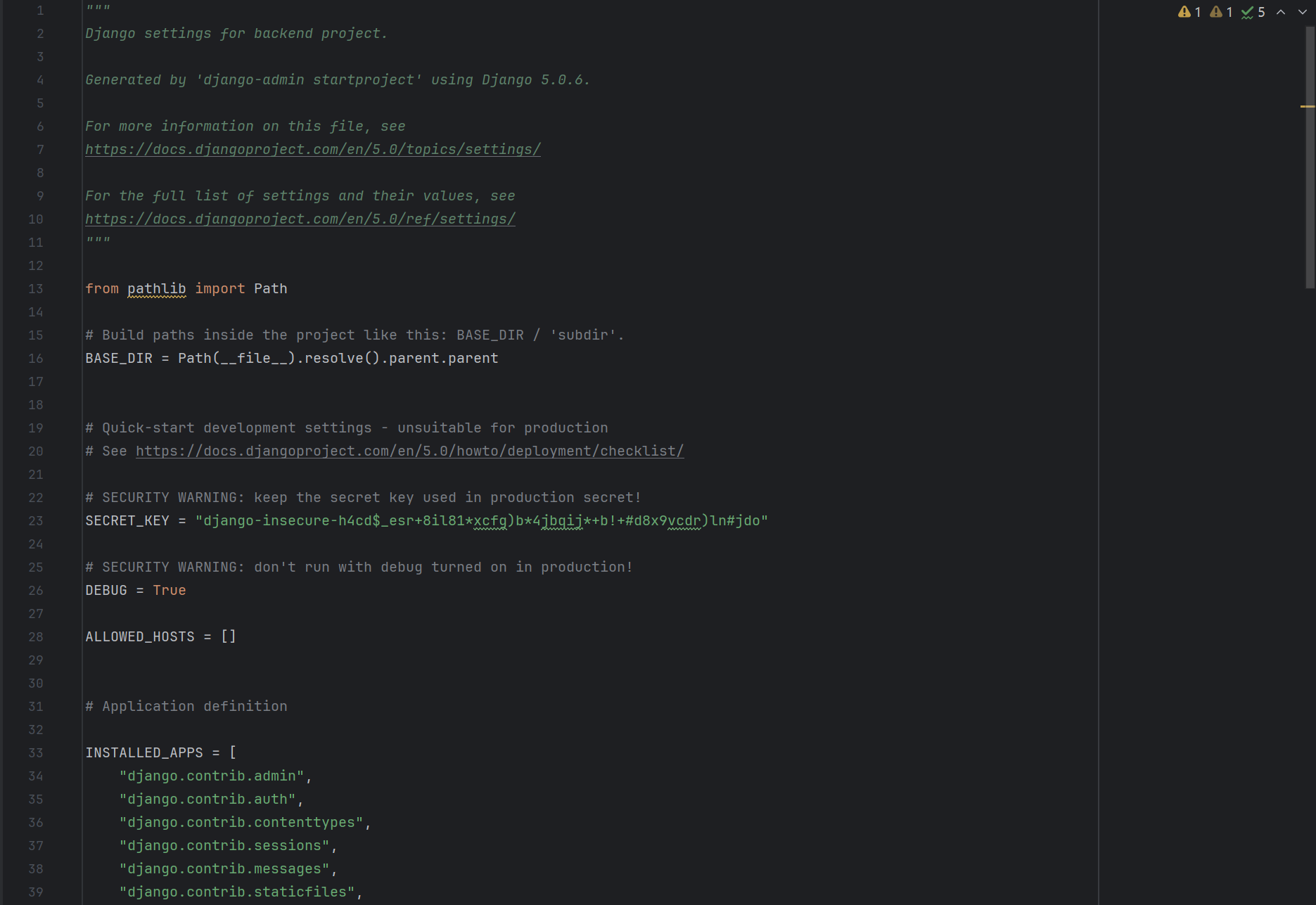
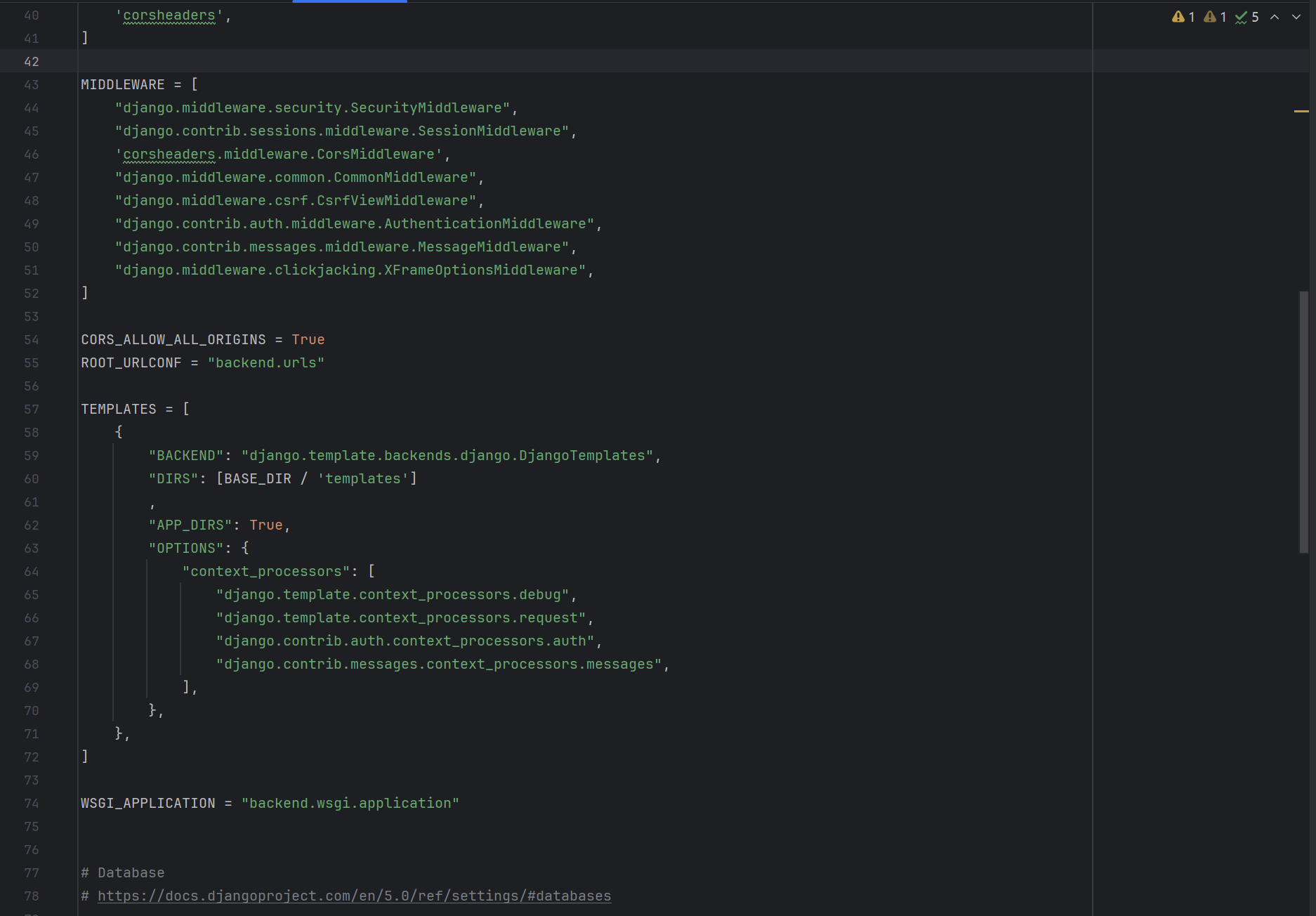
本项目基于Django+vue3+Mysql架构，进行了重新的分层，同时共享了业务代码。简化了开发、增强了与Nginx技术、Redis技术的结合。提供了一种更高效的开发模式。

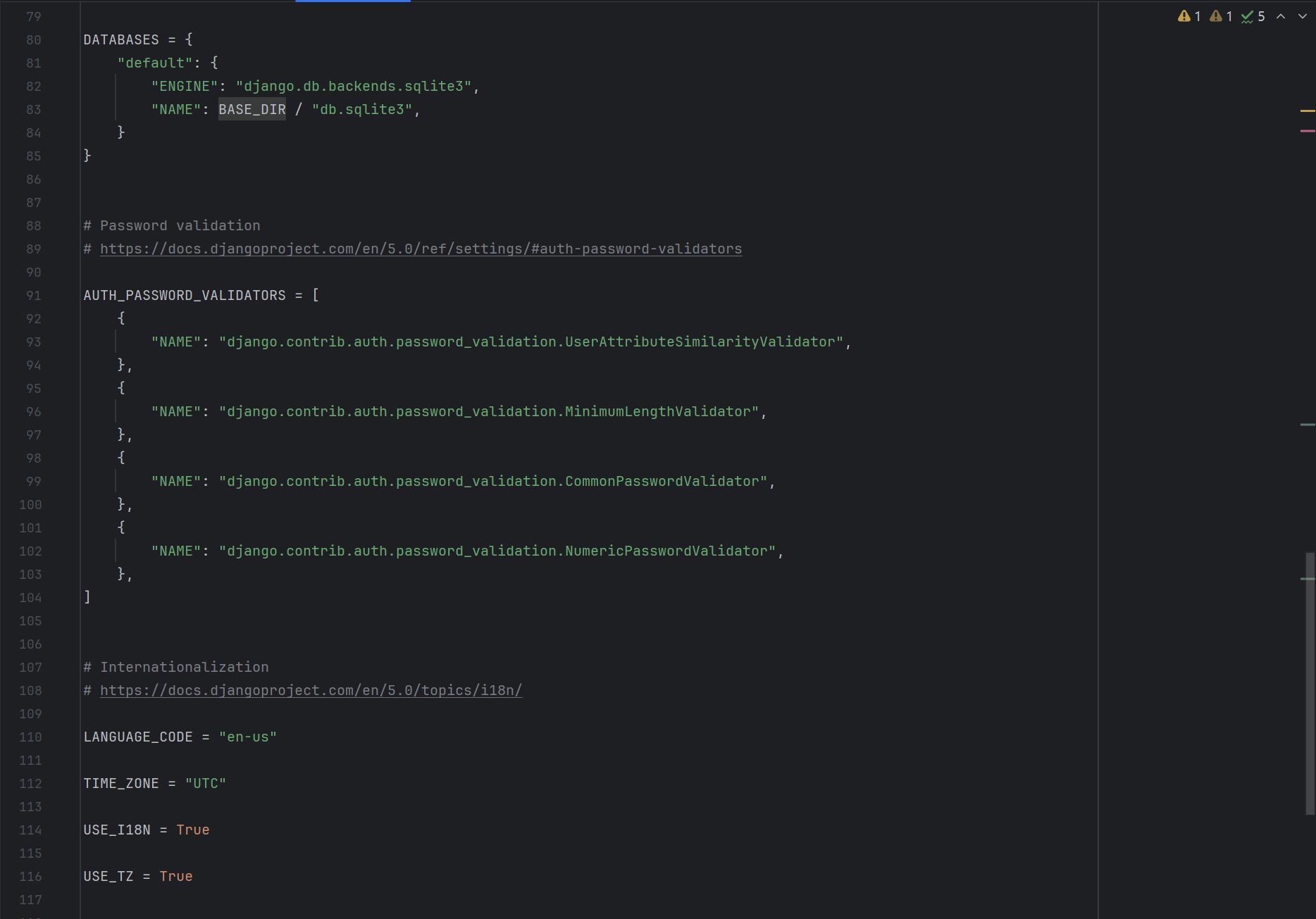
#### 物理架构设计

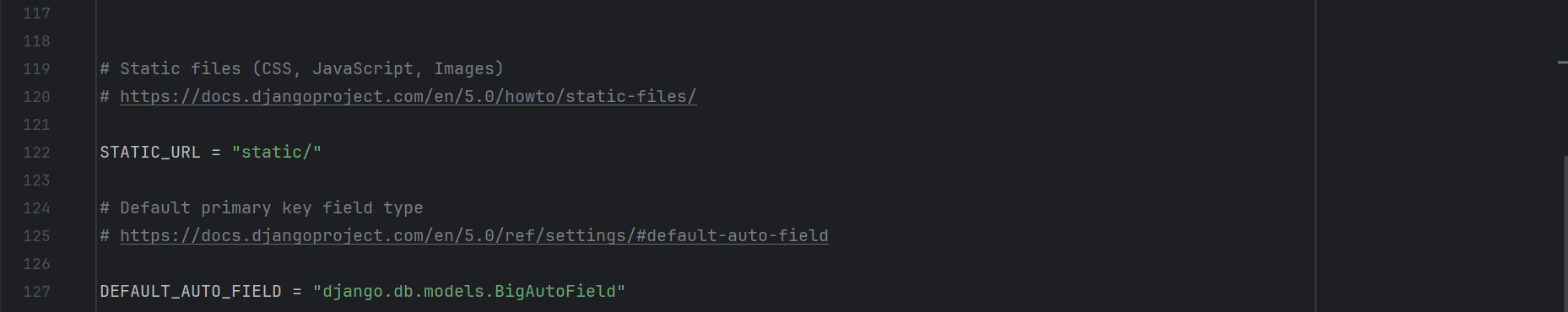


#### Web环境下的集成配置

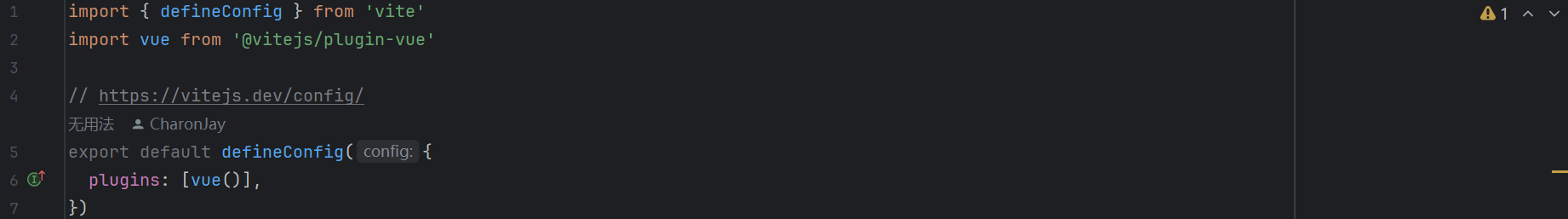
1.Django配置文件



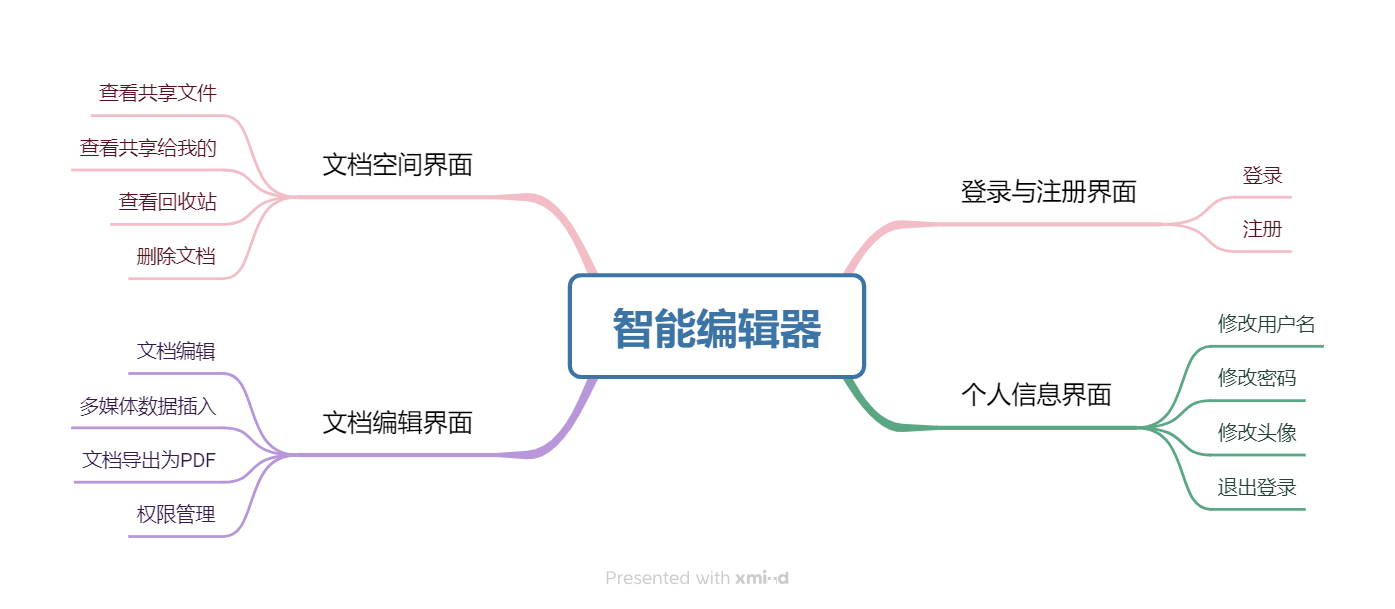


2.Vue配置文件



### 界面设计

#### 界面设计



##### 界面框架设计

#### 1.1登陆与注册界面



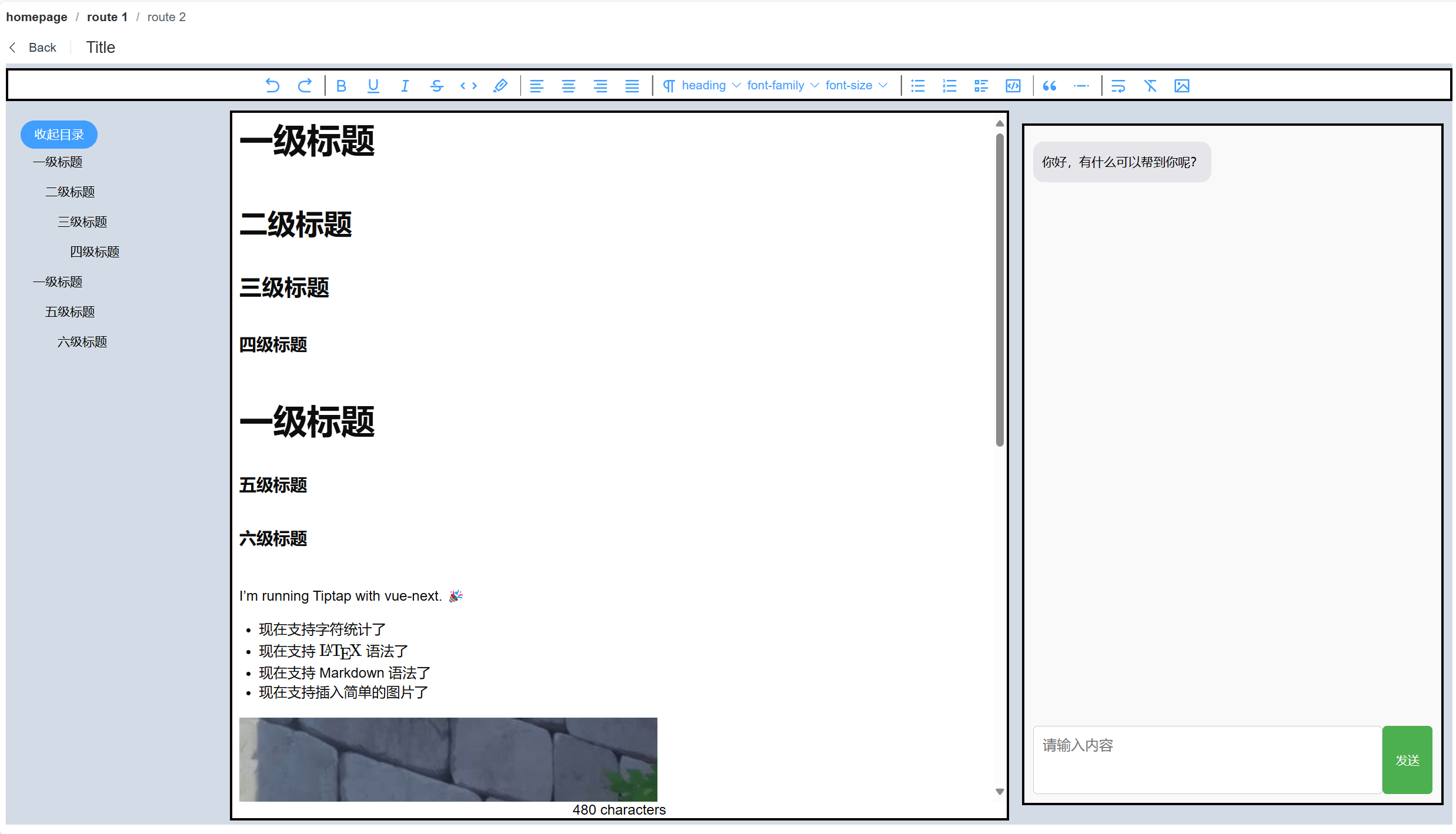
#### 1.2个人信息界面



#### 1.3文档空间界面



#### 1.4文档编辑界面



##### PC界面设计

#### 2.1登陆界面



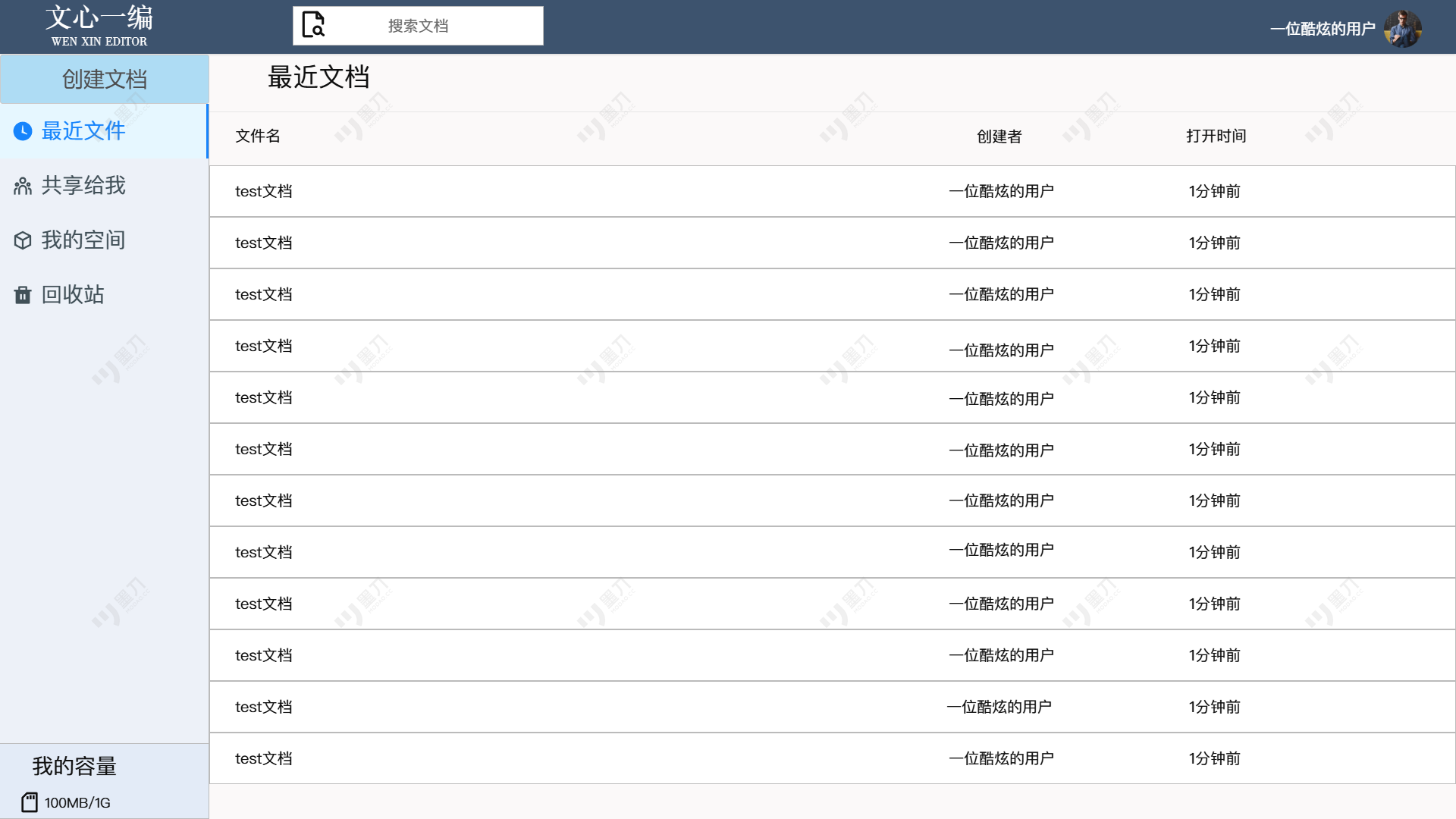
#### 2.2注册界面



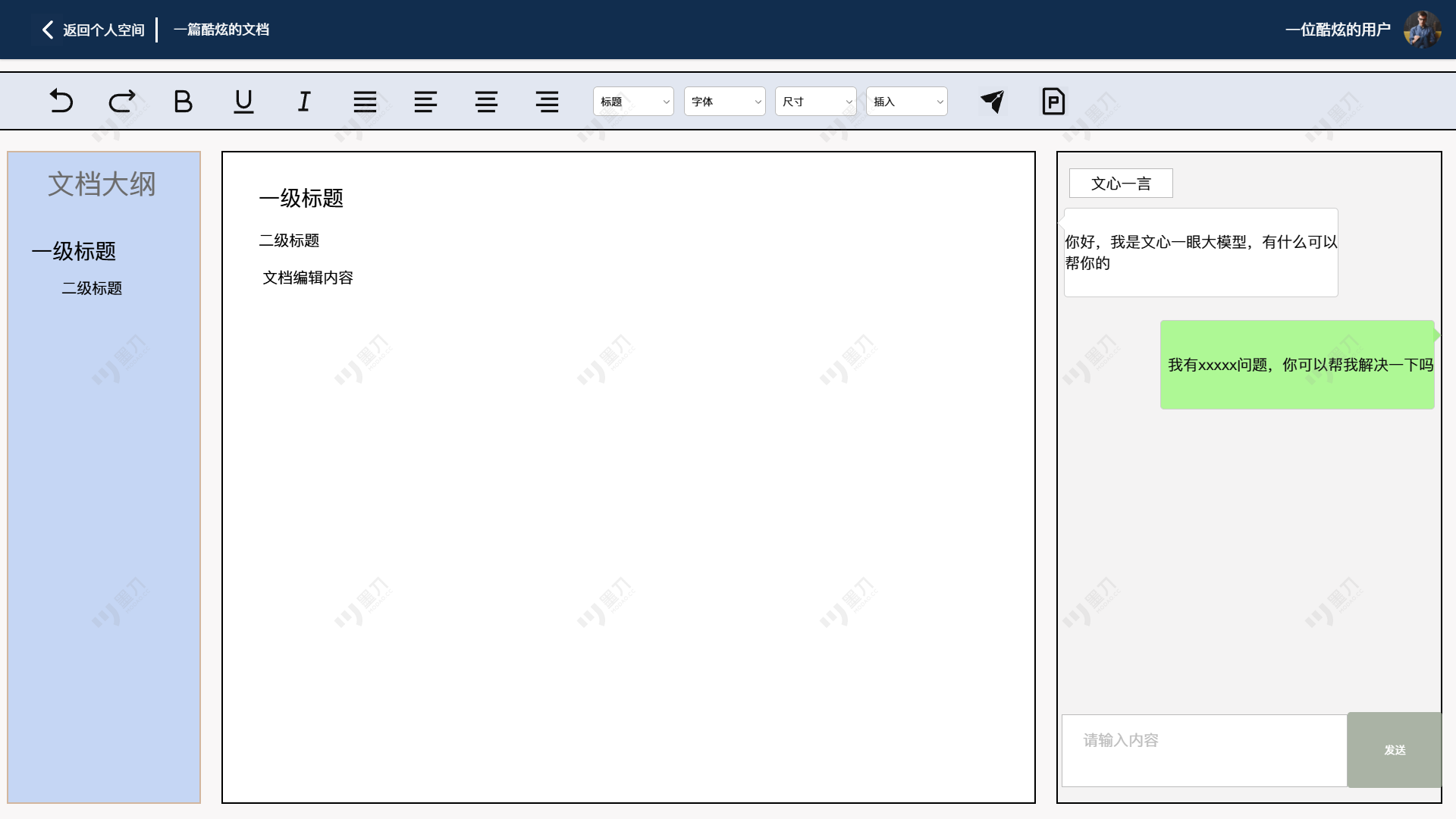
#### 2.3个人信息界面



#### 2.4文档空间界面



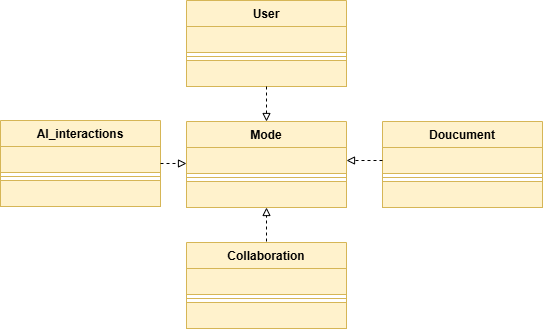
#### 2.5文档编辑页面



### 单元模块设计

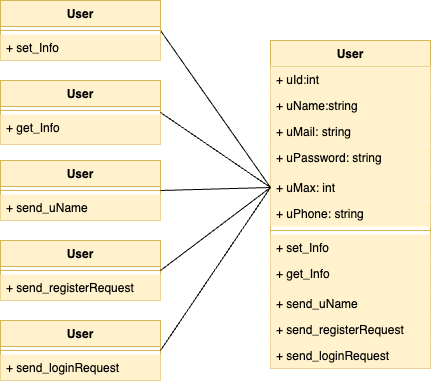
#### 模型层设计

##### 类图设计



##### 类的详细设计描述

###### User设计



**详细描述：**

|  |
| --- |
| set\_Info(uName:string, uPassword:string, \*\*kwargs)  设置用户属性 |
| get\_Info()->userInfo:dict  获取用户属性 |
| send\_uName(uName:string)->response:int  单独发送用户名称 |
| send\_registerRequest(uName:string, uPassword:string, uPhone,\*\*kwargs)->response:int  发送注册请求 |
| send\_loginRequest(uName:string, uPassword:string)->response:int  发送登陆请求 |

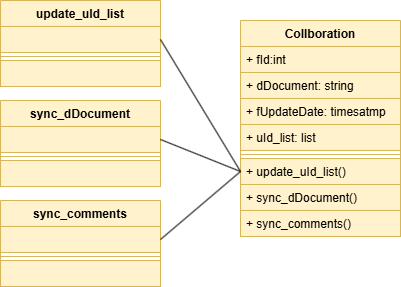
###### Document设计

###### 

**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 |

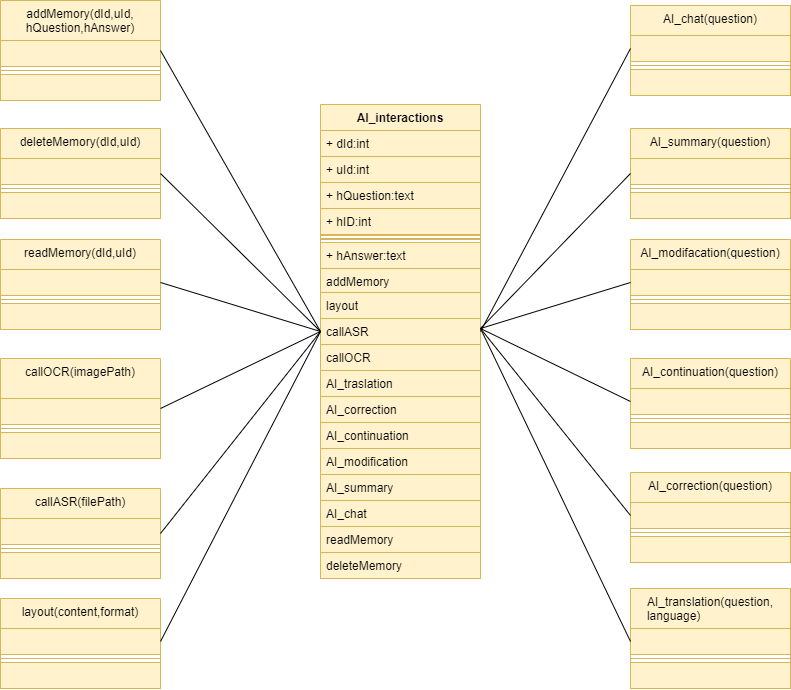
###### Collobration设计



**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 |

###### AI\_interactions设计

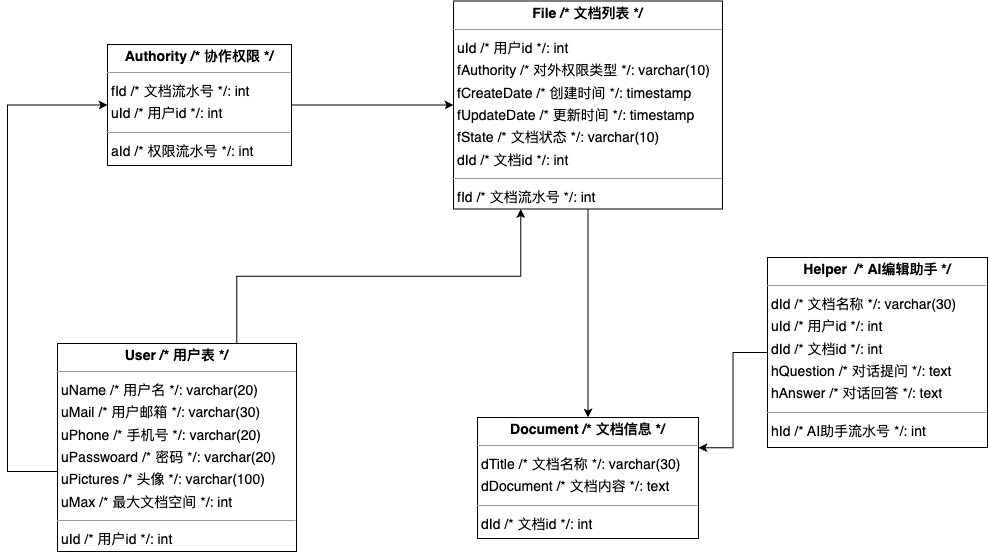


**详细描述：**

|  |
| --- |
| addMemory(dId,uId,hQuestion,hAnswer);  保存一轮对话的问题和回答 |
| deleteMemory(dId,uId)  删除id为uId的用户在id为dId的文档的对话记录 |
| readMemory(dId,uId) ;  读取id为uId的用户在id为dId的文档的对话记录 |
| AI\_chat(question)  用户与AI的多轮对话 |
| AI\_summary(question)  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 |

### 数据库设计

#### 数据库整体结构图



#### 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 |

#### 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 |

#### 后台功能（保存用户数据）

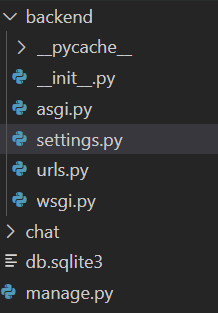
1.用户表结构

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

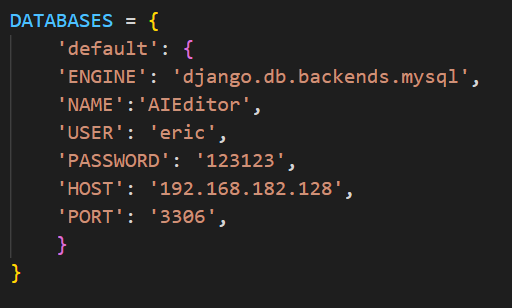
### 补充设计和说明

#### 编译运行环境设计

##### 数据库连接设计

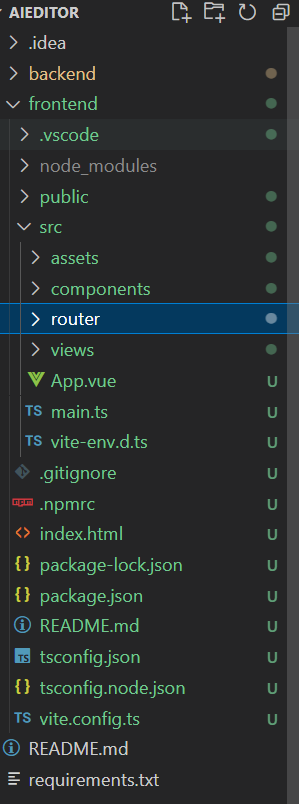


在Django项目中配置数据库，在backend\setting.py中加入如下内容：



#### 包路径与WEB目录结构设计

##### vue目录结构设计



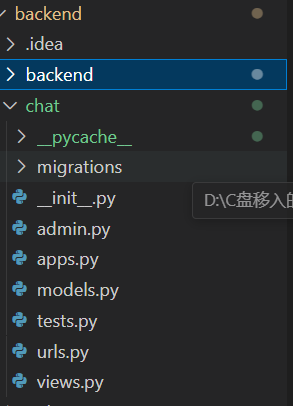
1. AIEDITOR为项目主包，frontend为前端项目名称
2. 其下有node\_modules、public、src文件夹

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。

1. 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。

##### 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。