# 1、系统概述

**1.1、系统简介**

简要概述系统的基本情况和背景，还包括概要设计中术语的介绍等。

本系统是一个小巧轻量、集发布和讨论于一体的论坛平台；基于北邮人论坛的使用和周围同学的反馈背景，让用户体验更加简约流畅。

**1.2、系统目标**

概述概要设计要实现的目标，包括功能目标、性能目标等。

功能目标：

1、论坛用户可以注册和登录，并能够发布文章和动态、查看用户消息

2、论坛用户可以对论坛上的文章和动态进行评论、点赞和收藏

3、管理端可以对用户发送系统消息和禁用账户

4、管理端可以对论坛上的帖子和评论进行审核、排序和删除

性能目标：

1、精度：用户上传的文字和图片信息能够被系统正确识别、处理和输出

2、时间特性：用户能够快速访问和使用系统，系统能在短时间内快速响应用户的请求

3、灵活性：系统能够适应不同用户和不同场景下的使用需求、技术和环境的变化

4、故障处理：对于网络、数据库故障和安全漏洞能够及时发现和维护

**1.3、系统运行环境**

包括对硬件平台、操作系统、数据库系统、编程平台、网络协议等的描述。

硬件平台：Legion Y7000P 2019、Dell G15 5511、MacBook Air(M1)

操作系统：Windows、Mac OS、Linux

前端框架：Vue.js + Vue Route + Naïve UI + Element Plus + Less + Axios

后端框架：FastAPI+SQLAlchemy

数据库：Sqlite

编程平台：VSCode

网络协议：TCP/IP

**1.4、开发环境**

列举进行系统分析、程序设计和程序开发时要使用的工程工具和开发语言。应描述每一工具软件的名称、版本等。

工程工具：

集成开发环境：VSCode、PyCharm

版本控制工具：Git

自动化构建工具：Vite

调试工具：Volar、Unicorn、Chrome浏览器

截止到项目发布时的最新版

开发语言：

Vue：3.2.47

Python：3.9.12

# 2、总体结构设计

**2.1、软件结构**

按照不同功能进行整体结构层次的划分，并使各层功能相对独立。以图形方式给出软件系统的子系统（或软件包）划分，模块划分，子系统间、模块间关系等，并用接口来描述各模块之间的调用关系，给出各模块之间的松散耦合关系。

**2.2、设计思想**

结合上图阐述软件的基本设计思想和理念。

# 3、模块设计

在此以列表形式说明各模块的名称、功能、接口等信息。

**3.1、模块1**

详细描述各功能模块的功能、接口及内部实现。

用户注册、登录、注销模块

3.1.1、功能描述

该模块的主要功能是允许用户注册一个新的账户，使用账户登录论坛并注销账户。用户注册时需要提供用户名、密码和电子邮箱等信息，并进行验证以确保其唯一性和合法性。用户登录时需要输入用户名和密码进行身份验证。注销账户时会清除与该账户相关的所有信息并退出登录状态。

3.1.2、接口描述

描述与其他功能模块的每个接口的名称、输入信息、输出信息、异常处理等。

注册接口(register)：接收用户输入的账户名、密码、邮箱等信息，进行注册，注册成功返回用户信息，注册失败返回错误信息。

登录接口(login)：接收用户输入的账户名和密码，进行登录，登录成功返回用户信息，登录失败返回错误信息。

注销接口(logout)：接收用户请求，清除用户登录状态，注销成功返回成功信息，注销失败返回错误信息。

个人信息管理接口(updateProfile)：接收用户的个人信息修改请求，如修改密码、邮箱、头像等，修改成功返回成功信息，修改失败返回错误信息。

3.1.3、数据结构描述

本功能区域使用的数据结构。

User类：用于表示一个用户的信息，包括用户名、密码、邮箱、头像等。

UserDatabase类：用于存储用户信息的数据库，包括用户的基本信息和登录状态信息。

3.1.4、实现思路

给出实现该模块的基本思路，包括对模块内部结构、算法、编程方法等的初步设想。

注册功能实现：

a. 接收用户输入的账户名、密码、邮箱等信息。

b. 检查输入的信息是否合法，如账户名是否已经存在，密码是否符合要求等。

c. 如果输入的信息合法，则生成一个用户对象，将用户信息存储到用户数据库中，并返回用户信息。

d. 如果输入的信息不合法，则返回错误信息。

登录功能实现：

a. 接收用户输入的账户名和密码。

b. 验证用户输入的账户名和密码是否正确。

c. 如果账户名和密码正确，则将用户的登录状态设置为已登录，并返回用户信息。

d. 如果账户名或密码不正确，则返回错误信息。

注销功能实现：

a. 接收用户请求注销账户。

b. 清除用户登录状态。

c. 返回注销成功信息。

个人信息管理功能实现：

a. 接收用户请求修改个人信息。

b. 检查用户输入的信息是否合法，如修改后的密码是否符合要求等。

c. 如果输入的信息合法，则更新用户信息，并返回修改成功信息。

d. 如果输入的信息不合法，则返回错误信息。

**3.2、模块2**

**帖子模块**

3.2.1 功能描述

帖子模块负责帖子的发布、编辑、删除、查看、搜索、分类等功能。用户可以通过该模块发布新帖子，查看已有的帖子，对帖子进行分类、搜索帖子，也可以编辑自己的帖子并删除自己的帖子。

该模块具体实现的功能包括：

帖子发布：用户可以通过界面填写帖子的标题、内容等信息并发布。

帖子编辑：用户可以对自己发布的帖子进行编辑，修改帖子的标题、内容等信息。

帖子删除：用户可以删除自己发布的帖子。

帖子查看：用户可以查看其他用户发布的帖子，并可以对帖子进行点赞、评论等操作。

帖子分类：用户可以将自己发布的帖子进行分类，方便其他用户查找。

帖子搜索：用户可以通过关键词搜索已发布的帖子，以便快速找到自己需要的信息。

3.2.2 接口描述

发布新帖子

输入信息：帖子标题、帖子内容、帖子标签

输出信息：成功/失败信息

异常处理：如果输入信息不完整或不符合格式要求，会返回失败信息。

编辑帖子

输入信息：帖子id、帖子标题、帖子内容、帖子标签

输出信息：成功/失败信息

异常处理：如果输入信息不完整或不符合格式要求，会返回失败信息。

删除帖子

输入信息：帖子id

输出信息：成功/失败信息

异常处理：如果输入信息不完整或不符合格式要求，会返回失败信息。

查看帖子

输入信息：帖子id

输出信息：帖子详情（包括标题、内容、标签、作者、发布时间、点赞数、评论数等信息）

异常处理：如果输入信息不完整或不符合格式要求，会返回失败信息。

搜索帖子

输入信息：关键词、标签

输出信息：符合条件的帖子列表

异常处理：如果输入信息不完整或不符合格式要求，会返回失败信息。

分类帖子

输入信息：标签

输出信息：符合条件的帖子列表

异常处理：如果输入信息不完整或不符合格式要求，会返回失败信息。

3.2.3 数据结构描述

本模块使用的数据结构包括：

Post：帖子数据结构，包括帖子id、标题、内容、标签、作者、发布时间、点赞数、评论数等信息。

PostList：帖子列表数据结构，包括多个Post。

3.2.4 实现思路

该模块的实现主要包括以下几个步骤：

建立Post和PostList数据结构。

实现发布新帖子、编辑帖子、删除帖子、查看帖子、搜索帖子和分类帖子的功能。

针对不同功能，建立相应的函数和接口。

使用数据库存储帖子相关信息。

实现帖子的点赞和评论功能。

根据需要，实现其他扩展功能，如置顶帖子、帖子推荐等。

对代码进行测试和优化。

**3.3、模块3**

**评论模块**

3.3.1、功能描述：

评论模块允许用户对某个帖子进行评论。用户可以查看其他用户的评论，对评论进行回复和点赞。

3.3.2、接口描述：

接口名称：create\_comment

输入信息：用户ID、帖子ID、评论内容

输出信息：评论ID、评论时间

异常处理：如果用户ID或帖子ID不存在，则返回错误信息

接口名称：get\_comments\_by\_post

输入信息：帖子ID

输出信息：该帖子的所有评论

异常处理：如果帖子ID不存在，则返回错误信息

接口名称：get\_comments\_by\_user

输入信息：用户ID

输出信息：该用户的所有评论

异常处理：如果用户ID不存在，则返回错误信息

接口名称：delete\_comment

输入信息：评论ID

输出信息：删除成功信息

异常处理：如果评论ID不存在，则返回错误信息

接口名称：update\_comment

输入信息：评论ID、新的评论内容

输出信息：更新成功信息

异常处理：如果评论ID不存在，则返回错误信息

以上是评论模块的接口描述，create\_comment用于创建评论，get\_comments\_by\_post和get\_comments\_by\_user用于获取帖子或用户的评论，delete\_comment用于删除评论，update\_comment用于更新评论内容。所有接口都包含输入信息、输出信息和异常处理。

3.3.3、数据结构描述：

Comment：表示评论的数据结构。

id：评论的唯一标识符。

content：评论的内容。

author\_id：评论的作者ID。

post\_id：评论所属的帖子ID。

timestamp：评论的发布时间戳。

Reply：表示回复的数据结构。

id：回复的唯一标识符。

content：回复的内容。

author\_id：回复的作者ID。

comment\_id：回复所属的评论ID。

timestamp：回复的发布时间戳。

3.3.4、实现思路：

评论模块需要和帖子模块进行交互，先通过帖子ID获取到对应的帖子，然后可以向评论列表中添加新的评论，或者对某个评论进行回复，这时需要指定回复的评论ID。对评论和回复进行点赞需要使用一个点赞表来存储用户对评论和回复的点赞状态。在获取评论和回复列表时，需要将对应的用户信息进行查询，以便在前端展示。在存储评论和回复时，需要进行数据合法性检查，如检查评论和回复的长度、是否为空等。

**3.4、模块4**

**关注模块**

3.4.1 功能描述

关注模块是用户可以关注其他用户、帖子、话题等内容的功能模块。用户可以在该模块中查看自己已关注的对象、取消关注、查找其他用户并关注等。

3.4.2 接口描述

用户关注用户：add\_following\_user(user\_id: str) -> bool

输入：被关注用户的用户ID。

输出：返回关注操作是否成功，成功返回True，失败返回False。

异常处理：若用户ID不存在或者已经关注过该用户，则返回False。

用户取消关注用户：remove\_following\_user(user\_id: str) -> bool

输入：被取消关注用户的用户ID。

输出：返回取消关注操作是否成功，成功返回True，失败返回False。

异常处理：若用户ID不存在或者未关注该用户，则返回False。

用户关注帖子：add\_following\_post(post\_id: str) -> bool

输入：被关注帖子的帖子ID。

输出：返回关注操作是否成功，成功返回True，失败返回False。

异常处理：若帖子ID不存在或者已经关注过该帖子，则返回False。

用户取消关注帖子：remove\_following\_post(post\_id: str) -> bool

输入：被取消关注帖子的帖子ID。

输出：返回取消关注操作是否成功，成功返回True，失败返回False。

异常处理：若帖子ID不存在或者未关注该帖子，则返回False。

用户关注话题：add\_following\_topic(topic\_id: str) -> bool

输入：被关注话题的话题ID。

输出：返回关注操作是否成功，成功返回True，失败返回False。

异常处理：若话题ID不存在或者已经关注过该话题，则返回False。

用户取消关注话题：remove\_following\_topic(topic\_id: str) -> bool

输入：被取消关注话题的话题ID。

输出：返回取消关注操作是否成功，成功返回True，失败返回False。

异常处理：若话题ID不存在或者未关注该话题，则返回False。

获取用户已关注的用户列表：get\_following\_users(user\_id: str) -> List[str]

输入：用户ID。

输出：返回用户已关注的用户ID列表。

异常处理：若用户ID不存在或未关注任何用户，则返回空列表。

获取用户已关注的帖子列表：get\_following\_posts(user\_id: str) -> List[str]

输入：用户ID。

输出：返回用户已关注的帖子ID列表。

异常处理：若用户ID不存在或未关注任何帖子，则返回空列表。

获取用户已关注的话题列表：get\_following\_topics(user\_id: str) -> List[str]

输入：用户ID。

输出：返回用户已关注的话题ID列表。

异常处理：若用户ID不存在或未关注任何话题，则返回空列表。

查找用户并关注：find\_user\_and\_follow(user\_id: str, query: str) -> bool

输入：用户ID和查询关键字。

输出：返回是否找到目标用户并关注成功，成功返回True，失败返回False。

异常处理：若用户ID不存在或关键字不存在，则返回空列表。

3.4.3 数据结构描述：

用户关注列表：以用户ID为键，关注列表为值的字典。每个用户的关注列表存储被关注用户的ID列表。

帖子关注列表：以帖子ID为键，关注列表为值的字典。每个帖子的关注列表存储被关注用户的ID列表。

3.4.4 实现思路：

用户关注列表的实现思路：

在用户注册时，为每个新用户创建一个空的关注列表。

用户可以通过访问被关注用户的个人主页来执行关注操作。当用户关注某个用户时，将该用户的ID添加到当前用户的关注列表中。

当用户访问他人的个人主页时，通过查找该用户的关注列表，判断当前用户是否已经关注了该用户，并据此展示“关注”或“取消关注”按钮。

帖子关注列表的实现思路：

在发布帖子时，为每个新帖子创建一个空的关注列表。

当用户关注某个帖子时，将该用户的ID添加到该帖子的关注列表中。

当用户访问帖子详情页时，通过查找该帖子的关注列表，判断当前用户是否已经关注了该帖子，并据此展示“关注”或“取消关注”按钮。

关注列表的更新：

当用户执行关注或取消关注操作时，更新用户的关注列表。

当用户访问帖子详情页时，更新该帖子的关注列表。

当用户访问个人主页时，更新该用户的关注列表。

**3.5、模块5**

**管理员模块**

3.5.1 功能描述：

管理员模块主要实现了对用户、帖子、评论等内容的管理功能，包括查看、删除、封禁等操作，同时也提供了对网站统计数据的查看和管理功能。

3.5.2 接口描述：

管理员模块与其他模块的主要接口如下：

与用户模块的接口：

查看用户列表：从用户模块获取所有用户列表。

删除用户：调用用户模块的接口删除指定用户。

封禁用户：调用用户模块的接口封禁指定用户。

与帖子模块的接口：

查看帖子列表：从帖子模块获取所有帖子列表。

查看帖子详情：从帖子模块获取指定帖子的详细信息。

删除帖子：调用帖子模块的接口删除指定帖子。

与评论模块的接口：

查看评论列表：从评论模块获取所有评论列表。

查看评论详情：从评论模块获取指定评论的详细信息。

删除评论：调用评论模块的接口删除指定评论。

与数据统计模块的接口：

查看用户数量：从数据统计模块获取当前用户数量。

查看帖子数量：从数据统计模块获取当前帖子数量。

查看评论数量：从数据统计模块获取当前评论数量。

3.5.3 数据结构描述

管理员模块使用的数据结构包括：

管理员账号信息表：记录管理员的账号、密码等信息。

帖子审核表：记录待审核的帖子的信息，包括帖子ID、帖子标题、帖子内容、作者ID等。

评论审核表：记录待审核的评论的信息，包括评论ID、评论内容、作者ID、所属帖子ID等。

用户信息表：记录所有用户的信息，包括用户ID、用户名、密码等。

帖子表：记录所有已发布的帖子的信息，包括帖子ID、帖子标题、帖子内容、作者ID、发布时间等。

评论表：记录所有已发布的评论的信息，包括评论ID、评论内容、作者ID、所属帖子ID、发布时间等。

以上数据结构可以使用关系型数据库实现，例如MySQL、PostgreSQL等。

3.5.4 实现思路

管理员模块的实现思路包括：

管理员登录验证：管理员输入账号、密码后，通过查询管理员账号信息表来验证是否为合法管理员，验证成功后跳转到管理员界面。

帖子审核：管理员查看帖子审核表中的待审核帖子，对其进行审核通过或者拒绝。审核通过的帖子会被添加到帖子表中，审核拒绝的帖子会被删除。

评论审核：管理员查看评论审核表中的待审核评论，对其进行审核通过或者拒绝。审核通过的评论会被添加到评论表中，审核拒绝的评论会被删除。

用户管理：管理员可以对用户信息表进行增、删、改、查操作，例如添加新用户、禁用某个用户、修改用户密码等。

帖子管理：管理员可以对帖子表进行增、删、改、查操作，例如删除某个帖子、修改帖子内容等。

评论管理：管理员可以对评论表进行增、删、改、查操作，例如删除某个评论、修改评论内容等。

以上功能可以使用编程语言如Python、Java等实现，使用关系型数据库进行数据存储和管理。管理员模块的实现需要与其他模块进行交互，例如与用户模块交互进行用户管理、与帖子模块交互进行帖子管理等。需要定义良好的接口，确保模块之间的数据交互正确、高效。同时，需要进行异常处理，防止因为数据错误或者其他原因导致系统崩溃或数据丢失。

**3.6、模块6**

**通知模块**

3.1.1 功能描述

通知模块是用于发送通知消息的功能模块，可以向系统中的用户或者特定的用户组发送通知消息，以便及时传达系统的最新动态、消息或事件。

3.1.2 接口描述

该模块与其他模块的接口主要包括以下几个方面：

用户模块：通过用户模块获取用户信息，以便向指定用户或者用户组发送通知。

权限管理模块：通知模块需要使用权限管理模块的接口判断发送通知的用户是否有发送通知的权限。

消息模块：在消息模块中保存通知信息，以便用户查看。

具体的接口信息如下：

发送通知接口：send\_notification(sender\_id, receiver\_id\_list, message, timestamp)

sender\_id: 发送者的用户ID

receiver\_id\_list: 接收者的用户ID列表，可以是一个用户或者多个用户组

message: 发送的通知消息

timestamp: 发送通知的时间戳

查看通知接口：get\_notifications(user\_id, start\_time, end\_time)

user\_id: 查看通知的用户ID

start\_time: 查看通知的起始时间

end\_time: 查看通知的结束时间

删除通知接口：delete\_notification(notification\_id)

notification\_id: 要删除的通知ID

3.1.3 数据结构描述

通知模块主要使用以下数据结构：

Notification: 通知的数据结构，包含通知ID、发送者ID、接收者ID列表、通知内容、时间戳等字段。

3.1.4 实现思路

通知模块的实现思路主要包括以下几个方面：

数据库设计：使用数据库保存通知信息，包括通知ID、发送者ID、接收者ID列表、通知内容、时间戳等字段。

发送通知：在发送通知时，首先需要使用权限管理模块的接口判断发送通知的用户是否有发送通知的权限，然后将通知信息保存到数据库中，并将通知信息发送给指定的接收者。

查看通知：在查看通知时，用户可以调用get\_notifications接口获取自己的通知列表，并在界面上显示通知的内容。

删除通知：在删除通知时，用户可以调用delete\_notification接口删除自己的通知。

# 4、数据库与数据结构设计

**4.1、 数据库及数据表**

描述所使用的数据库系统,及数据库和数据表设计。如果系统不以数据库方式存储数据则可省略。

数据库系统：Sqlite

数据库和数据表设计：

用户表（users）:

id：用户ID，主键

username：用户名，唯一

email：电子邮件，唯一

password\_hash：密码哈希值

avatar\_url：头像URL

introduction：个人介绍

create\_time：注册时间

update\_time：最后一次更新时间

is\_admin：是否为管理员，布尔类型

is\_active：是否已被激活，布尔类型

帖子表（posts）

id：帖子ID，主键

title：帖子标题

content：帖子内容

create\_time：帖子创建时间

update\_time：最后一次更新时间

is\_deleted：是否已被删除，布尔类型

user\_id：发帖用户ID，外键

评论表（comments）

id：评论ID，主键

content：评论内容

create\_time：评论创建时间

update\_time：最后一次更新时间

is\_deleted：是否已被删除，布尔类型

user\_id：评论用户ID，外键

post\_id：所属帖子ID，外键

parent\_id：父评论ID，外键，可以为空

关注表（follows）

id：关注ID，主键

follower\_id：关注者ID，外键

followed\_id：被关注者ID，外键

点赞表（likes）

id：点赞ID，主键

user\_id：点赞用户ID，外键

post\_id：被点赞帖子ID，外键

comment\_id：被点赞评论ID，外键，可以为空

通知表（notifications）

id：通知ID，主键

content：通知内容

create\_time：通知创建时间

is\_read：是否已读，布尔类型

user\_id：接收通知的用户ID，外键

**4.2、 数据结构设计**

给出本系统内所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。

本系统中使用的数据结构包括以下表格：

用户表（users）：

标识符：users

数据项：

id: 用户ID，整数类型，主键，自增长

username: 用户名，字符串类型，唯一，长度为20

email: 电子邮件，字符串类型，唯一，长度为64

password\_hash: 密码哈希值，字符串类型

avatar\_url: 头像URL，字符串类型

introduction: 个人介绍，字符串类型

create\_time: 注册时间，日期时间类型

update\_time: 最后一次更新时间，日期时间类型

is\_admin: 是否为管理员，布尔类型

is\_active: 是否已被激活，布尔类型

帖子表（posts）：

标识符：posts

数据项：

id: 帖子ID，整数类型，主键，自增长

title: 帖子标题，字符串类型

content: 帖子内容，字符串类型

create\_time: 帖子创建时间，日期时间类型

update\_time: 最后一次更新时间，日期时间类型

is\_deleted: 是否已被删除，布尔类型

user\_id: 发帖用户ID，整数类型，外键，关联用户表的id字段

评论表（comments）：

标识符：comments

数据项：

id: 评论ID，整数类型，主键，自增长

content: 评论内容，字符串类型

create\_time: 评论创建时间，日期时间类型

update\_time: 最后一次更新时间，日期时间类型

is\_deleted: 是否已被删除，布尔类型

user\_id: 评论用户ID，整数类型，外键，关联用户表的id字段

post\_id: 所属帖子ID，整数类型，外键，关联帖子表的id字段

parent\_id: 父评论ID，整数类型，外键，可以为空，关联评论表的id字段

关注表（follows）：

标识符：follows

数据项：

id: 关注ID，整数类型，主键，自增长

follower\_id: 关注者ID，整数类型，外键，关联用户表的id字段

followed\_id: 被关注者ID，整数类型，外键，关联用户表的id字段

点赞表（likes）：

标识符：likes

数据项：

id: 点赞ID，整数类型，主键，自增长

user\_id: 点赞用户ID，整数类型，外键，关联用户表的id字段

post\_id: 被点赞帖子ID，整数类型，外键，关联帖子表的id字段

comment\_id: 被点赞评论ID，整数类型，外键，可以为空，关联评论表的id字段

通知表（notifications）：

标识符：notifications

数据项：

id: 通知ID，整数类型，主键，自增长

content: 通知内容，字符串类型

create\_time: 通知创建时间，日期时间类型

is\_read: 是否已读，布尔类型

user\_id: 接收通知的用户ID，整数类型，外键，关联用户表的id字段

这些表之间的关系如下：

用户表（users）和帖子表（posts）是一对多的关系，即一个用户可以创建多个帖子，但一个帖子只能由一个用户创建。

帖子表（posts）和评论表（comments）也是一对多的关系，即一个帖子可以有多个评论，但一个评论只能属于一个帖子。

评论表（comments）和自身之间存在一种多级关系，即一个评论可以有多个子评论（即回复），也可以没有子评论。

用户表（users）和关注表（follows）之间是多对多的关系，即一个用户可以关注多个其他用户，也可以被多个用户关注。

用户表（users）和点赞表（likes）之间也是多对多的关系，即一个用户可以给多个帖子或评论点赞，一个帖子或评论也可以被多个用户点赞。

用户表（users）和通知表（notifications）是一对多的关系，即一个用户可以收到多条通知，但一条通知只能发送给一个用户。

**4.3、 数据存储设计**

给出本系统内所使用的每个数据结构中的每个数据项的存储要求,访问方法、存取单位、存取的物理关系(索引、设备、存储区域)、设计考虑和保密条件。

访问方法：

以上数据结构中的数据项可以通过SQLAlchemy语句进行访问。

存取单位：

以上数据结构中的数据项可以按照单个记录（一行）或整个表格（多行）进行存取。

存取的物理关系：

以上数据结构中的数据项存储在SQLite数据库中，可以使用索引访问。

# 5、接口设计

**4.1、外部接口**

描述需要访问的外部接口的类型、控制方式。

前端和后端之间的接口采用RESTful API设计风格，使用HTTP协议进行通信。前端通过axios库向后端发送请求，后端返回JSON格式的响应。

**4.2、内部接口**

描述与其他模块及子系统的接口。

在后端中，各功能模块之间的接口采用函数调用的方式进行通信。例如，用户模块调用了权限模块的函数来检查用户是否具有特定操作的权限。同时，在后端中使用了中间件机制，例如FastAPI提供的Depends，来实现功能模块之间的通信和数据传递。

# 6、其他设计

对以上概要设计中未包含项目的说明，如并行处理、安全设计、内存管理等。

安全设计：OAuth2 实现密码哈希与 Bearer JWT 令牌验证