****

**标题**

**《认识实习大作业》**

**小组成员：**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 组长 标注 \* |
| 1 | **郭岩峰** | 221310213 | \* |

**指导老师：甘杨兰**

**选题：一个基于gptapi集成开发的AI论坛——RecomBlog**

[一、项目说明 3](#_Toc184641133)

[1.1 项目简述 3](#_Toc184641134)

[1.2 工作任务书（statement of work,SOW） 4](#_Toc184641135)

[二、项目任务分解 6](#_Toc184641136)

[2.1 项目介绍 6](#_Toc184641137)

[2.2 按照项目功能进行项目结构分解 7](#_Toc184641138)

[2.3 按照软件生命周期进行项目结构分解 7](#_Toc184641139)

[三、项目需求分析 8](#_Toc184641140)

[3.1 业务逻辑 8](#_Toc184641141)

[3.2 信息需求 8](#_Toc184641142)

[3.3 处理需求 8](#_Toc184641143)

[四、概要设计 9](#_Toc184641144)

[4.1 项目类图 9](#_Toc184641145)

[4.2 项目用例图 10](#_Toc184641146)

[4.3 概要设计说明书 11](#_Toc184641147)

[五、详细设计 13](#_Toc184641178)

[5.1 项目流程图（分用户端和管理端） 14](#_Toc184641179)

[5.2 项目交互图 15](#_Toc184641180)

[5.3 详细设计说明书 16](#_Toc184641181)

[六、数据库设计 18](#_Toc184641182)

[6.1 概念设计 18](#_Toc184641183)

[6.2 逻辑设计 19](#_Toc184641184)

[6.3 外部模型 19](#_Toc184641185)

[6.4 物理模型 20](#_Toc184641186)

[七、编码实现 21](#_Toc184641187)

[7.1 项目代码目录树 21](#_Toc184641188)

[7.2 关键代码功能阐述（关于使用api集成的详细过程） 26](#_Toc184641189)

[7.3 项目运行截图以及说明 28](#_Toc184641190)

[八、测试报告 37](#_Toc184641191)

[8.1 针对项目各个功能进行测试 37](#_Toc184641192)

[8.2 测试结果以及总结 39](#_Toc184641193)

[九、项目计划 40](#_Toc184641194)

[9.1 团队管理计划 40](#_Toc184641195)

[9.2 沟通管理计划 42](#_Toc184641198)

[9.3 质量管理计划 44](#_Toc184641202)

[十、项目总结 46](#_Toc184641207)

## 一、项目说明

### 1.1 项目简述

本项目是选择“1.2 AI娱乐赛道”开发的《RecomBlog》，一个基于AI技术的论坛系统。随着信息时代的迅速发展，人们对个性化服务的需求日益增加，尤其是在生活娱乐领域。该公司意识到，传统的论坛系统已经无法满足用户对精准内容推荐的期望，因此希望通过引入智能推荐功能，提升用户体验，增强平台的吸引力。

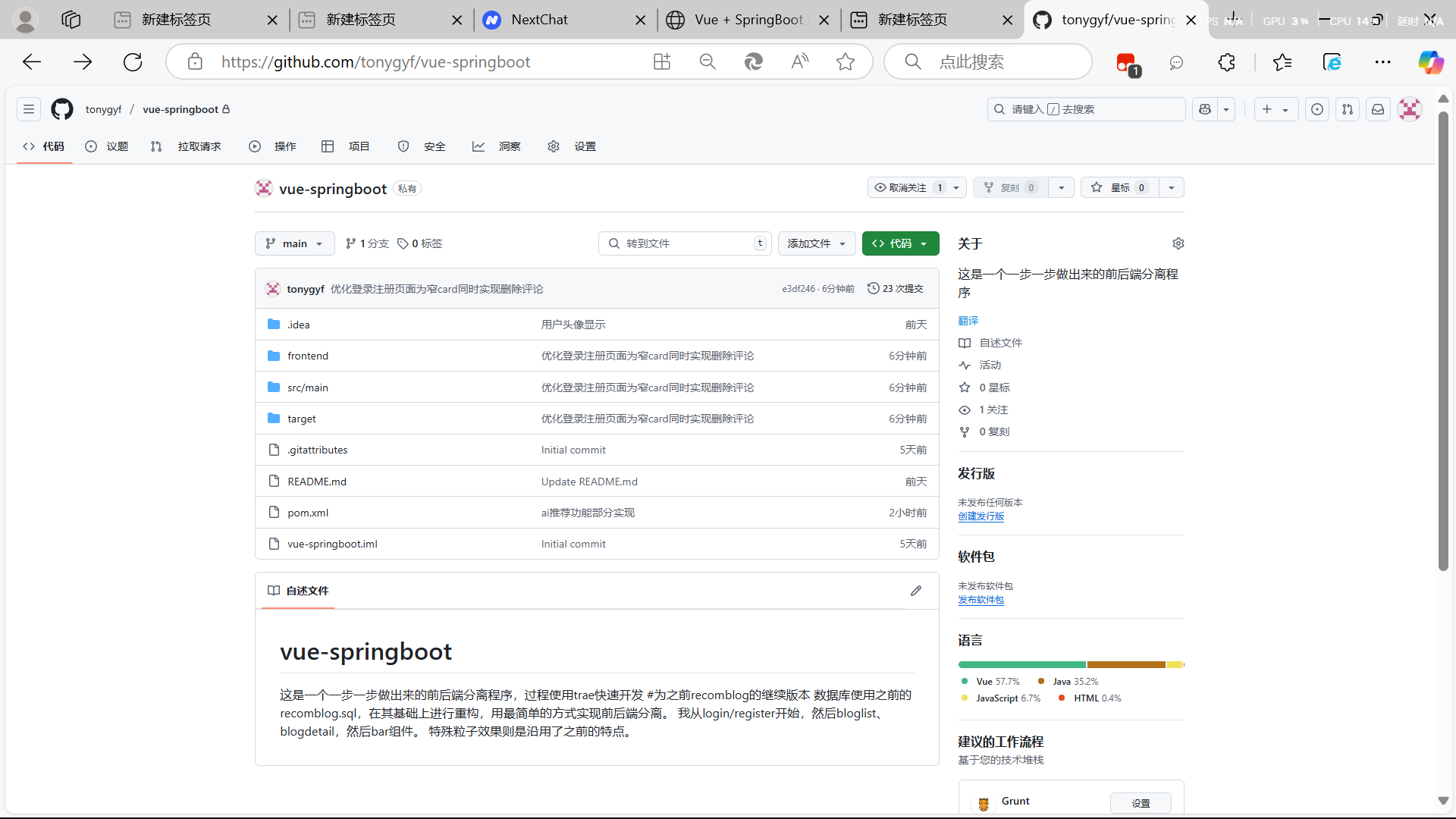
为了实现这一目标，我开发《RecomBlog》。该论坛系统不仅提供基本的讨论和交流功能，还融入了基于gptapi集成开发的AI推荐模块。通过分析用户的兴趣偏好和行为数据，系统能够智能推送相关的电影、音乐和其他内容，帮助用户发现更符合其喜好的信息。

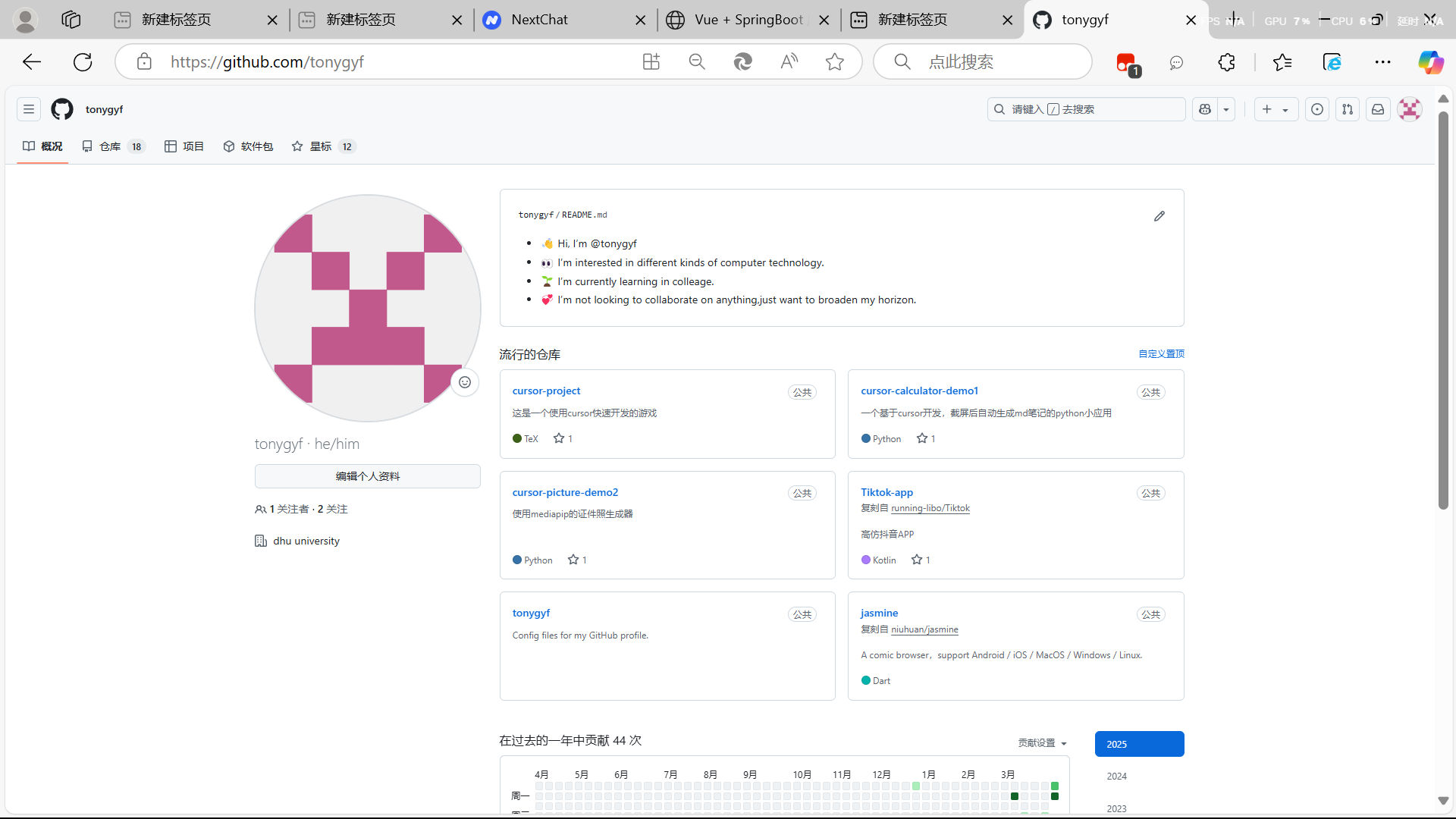
在《RecomBlog》中，用户可以轻松创建个人账户，参与讨论，并根据自己的兴趣维护个性化的内容偏好。此外，用户还可以通过系统提交反馈，帮助优化推荐算法。论坛的管理团队则利用系统提供的数据分析工具，监控用户活动，评估内容质量，并不断调整推荐策略，以提升用户满意度。

该系统的设计允许用户随时浏览和参与各类话题讨论，并根据推荐内容，快速找到感兴趣的文章和资源。通过这种方式，用户不仅能够享受更丰富的内容体验，还能与其他用户进行更深入的互动，形成良好的社区氛围。

总之，《RecomBlog》的开发旨在为用户提供一个智能化、个性化的论坛平台，提升用户参与度和满意度，同时为其他类似的生活娱乐平台或社区提供借鉴和参考，帮助它们更好地理解用户需求，提供定制化的服务，以适应信息时代的快速变化。

本次项目使用git进行版本管理并发布在github上，下面是我的仓库和账号：





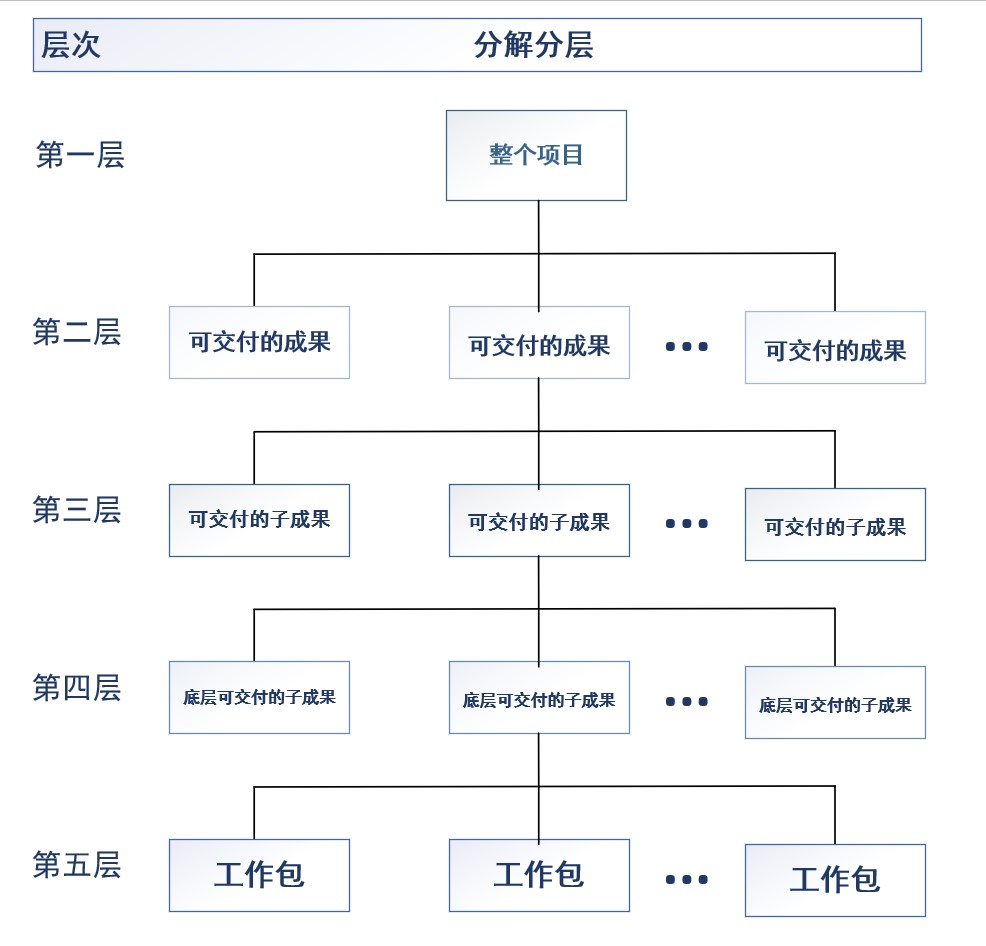
### 1.2 工作任务书（statement of work,SOW）

|  |
| --- |
| **项目计划书**   1. **项目概述**   **本项目致力于构建RecomBlog，一个基于 Java 的博客论坛项目，并在其中实现 AI 推荐系统，以根据用户偏好推荐电影和音乐。**   1. **项目功能分解**   **本项目RecomBlog针对使用人群为普通用户和管理员，并且以此分为两个子系统：管理端子系统和客户端子系统。**  **客户端拥有 注册登录、博客管理（查看、发布、编辑、删除、点踩、评论）、偏好设置、推荐展示等功能模块。**  **管理端拥有 推荐系统（收集用户偏好并实现推荐算法）、用户管理、博客管理等功能模块。**  **Ps：注意 客户端和管理端关于博客管理这个功能模块拥有不同的权限，管理端在**  **查看、发布、编辑、删除、点踩等功能中不受任何限制，但是客户端在编辑、删除**  **上只限于自己的博客。**  **权限管理**   * **客户端用户：**   + **可以查看所有博客**   + **可以发布自己的博客**   + **可以编辑和删除自己的博客**   + **可以点踩和评论任何博客** * **管理员用户：**   + **可以查看所有博客**   + **可以发布、编辑和删除所有博客**   + **可以点踩和评论任何博客**   + **可以管理用户和推荐系统**      1. **项目实现**   **本项目使用的开发框架为springboot**   1. **系统功能描述**   本项目分为客户端和管理端，客户端主要功能是：  博客浏览、点赞、评论、发布、个人信息查看、电影推荐、通过api集成的gpt通话。  管理端的功能提供点赞、博客、评论、用户管理。  **简略的关系图大致如下：** |

## 二、项目任务分解

### 2.1 项目介绍

通常来说，WBS的分层分解如下：



RecomBlog 是一个针对博客内容推荐和用户互动的平台，旨在提高用户体验和内容发现效率。系统面向管理员、博客作者和普通用户三类用户，并划分为客户端和管理端两个子系统，提供以下十项功能（前五项为客户端子系统，第六项二者皆有）：

（1）博客浏览发布

（2）博客评论点赞

（3）集成gpt通话

（4）个人信息查看

（5）ai电影推荐

（6）登录注册

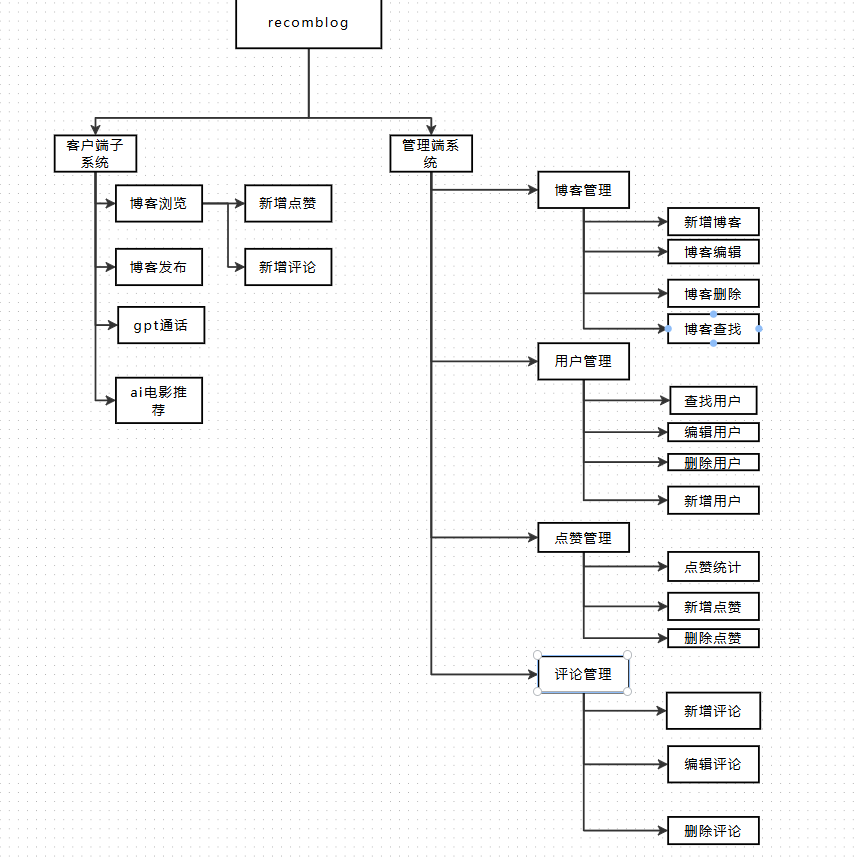
（7）评论管理

（8）博客管理

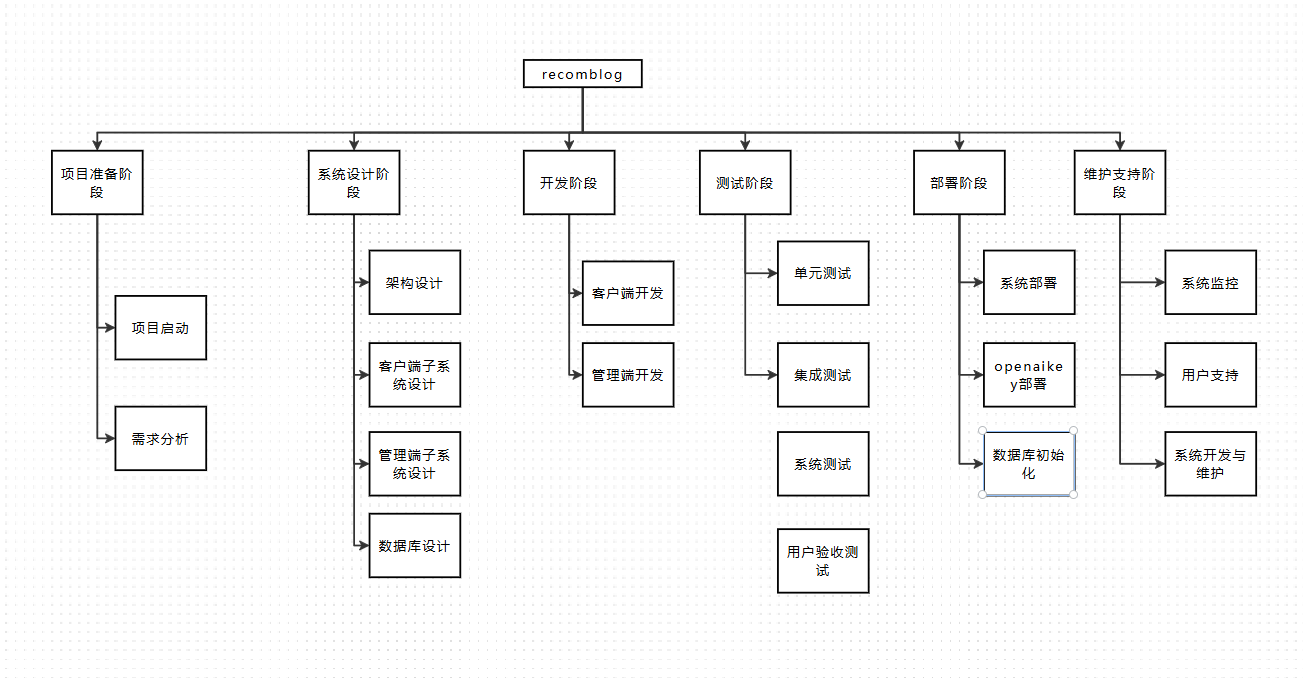
（9）点赞管理

（10）用户管理

### 2.2 按照项目功能进行项目结构分解



### 2.3 按照软件生命周期进行项目结构分解



## 三、项目需求分析

### 3.1 业务逻辑

本系统分按照users表属性role为两个角色，管理员（admin）和用户（client），设计各个角色在系统中交互的动作，功能概述如下：

### 3.2 信息需求

|  |
| --- |
| 1. **用户信息**    * 存储和管理用户的基本信息，包括用户名、密码、邮箱和角色。    * 记录用户的创建时间和更新时间。 2. **博客信息**    * 存储用户发布的博客文章，包括标题、内容和状态（是否已删除）。    * 记录博客的创建时间和更新时间。 3. **评论信息**    * 存储用户对博客的评论，包括评论内容和评论时间。    * 记录每条评论所属的博客和评论者的用户信息。 4. **推荐信息**    * gpt给用户电影的推荐，电影的关联信息。 5. **点赞信息**    * 存储用户对博客的点赞记录，包括点赞的时间。 |

### 3.3 处理需求

|  |
| --- |
| 1. **用户管理**    * 注册新用户，验证用户名和邮箱的唯一性。    * 更新用户信息（如密码、邮箱等）。    * 删除用户账户（逻辑删除，标记为已删除）。 2. **博客管理**    * 创建、更新和删除博客文章。    * 查询特定用户的所有博客。    * 查询所有博客及其相关信息（如评论和点赞）。 3. **评论管理**    * 添加、更新和删除评论。    * 查询特定博客的所有评论。    * 查询特定用户的所有评论。 4. **点赞管理**    * 添加和删除用户的点赞记录。    * 查询特定博客的所有点赞记录。 5. **gpt管理**    * 相应gpt的apikey和接口地址 |

## 四、概要设计

**概要设计（High-Level Design, HLD）** 是系统设计的第一步，主要描述系统的整体架构和主要模块。它注重“**是什么**”，以高层次的视角定义系统的功能和模块结构。

**用途：**

1. 帮助开发团队、客户或其他利益相关者了解系统的大致功能和实现方式。
2. 明确系统的主要模块和它们之间的关系。
3. 指导后续的详细设计和开发工作。

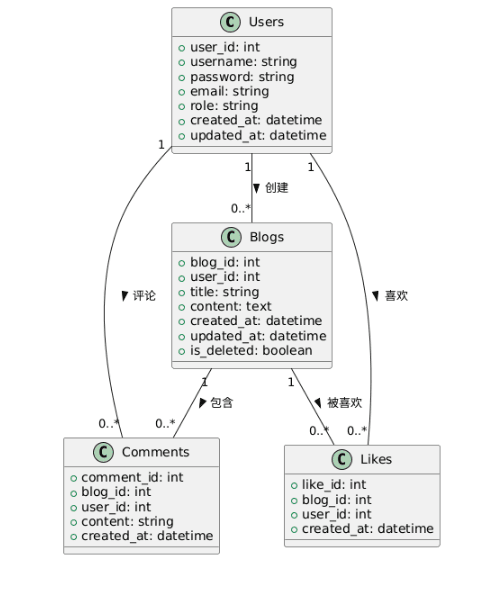
**特点：**

* 描述模块的职责和整体架构。
* 不深入具体实现细节。
* 关注模块之间的接口和交互。

### 4.1 项目类图

类图的作用是：

* **可视化**：通过图形化的方式展示系统的类及其关系，使得系统结构更易于理解。
* **文档化**：作为系统设计文档的一部分，类图记录了系统的静态结构，便于后续的维护和更新。
* **沟通工具**：为开发团队、设计人员和利益相关者提供了一种有效的沟通方式，确保对系统结构的理解一致。

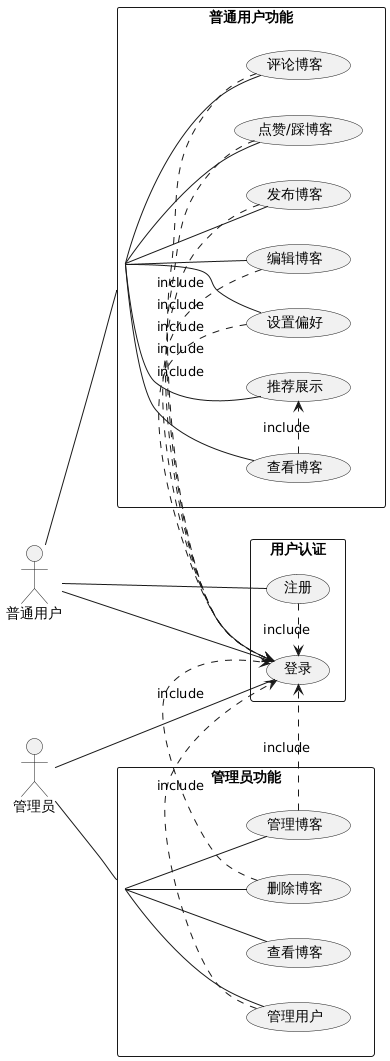


（后面新增了数据库chat\_responses，由于只是存储prompt和response的，跟其他实体没有关系因此没有画出）

### 4.2 项目用例图

用例图的作用是：

* **目的**：用例图用于描述系统的功能需求，展示用户（或其他系统）与系统之间的交互。
* **内容**：它帮助识别系统的主要用例（功能）和用户角色（参与者），并明确系统的边界。
* **作用**：为后续的详细设计提供基础，确保开发团队和利益相关者对系统功能的理解一致。



### 4.3 概要设计说明书

|  |
| --- |
| 概要设计说明书1. 系统概述RecomBlog 是一个博客管理系统，支持普通用户和管理员使用。系统分为两个子系统：客户端子系统：普通用户使用，支持博客管理、用户偏好设置、推荐展示等功能。管理端子系统：管理员使用，支持全局管理，包括用户管理、推荐系统管理及博客管理。2. 系统功能模块系统主要功能模块如下：用户管理：注册、登录功能。普通用户和管理员拥有不同的权限。博客、评论、点赞管理：普通用户：发布、查看、编辑、删除自己的博客；评论以及取消自己的评论、点赞以及取消自己的点赞。管理员：发布、查看、编辑、删除所有博客、评论、管理所有点赞。推荐系统：收集用户偏好并通过gpt生成个性化推荐。3. 权限管理普通用户：查看所有博客。发布、编辑、删除自己的博客。点赞、评论以及取消自己的点赞评论。Gpt通话以及使用电影推荐模块。管理员：查看所有博客。发布、编辑、删除所有博客。点赞、评论任何博客。管理用户。4. 系统设计架构系统采用分层架构，主要分为以下层次：前端展示层：包括用户界面和交互设计。业务逻辑层：处理用户请求和权限验证。数据持久层：管理数据库操作。 |

## 五、详细设计

**详细设计（Low-Level Design, LLD）** 是在概要设计的基础上，具体到系统实现的每个细节。它注重“**怎么做**”，提供明确的技术实现方案。

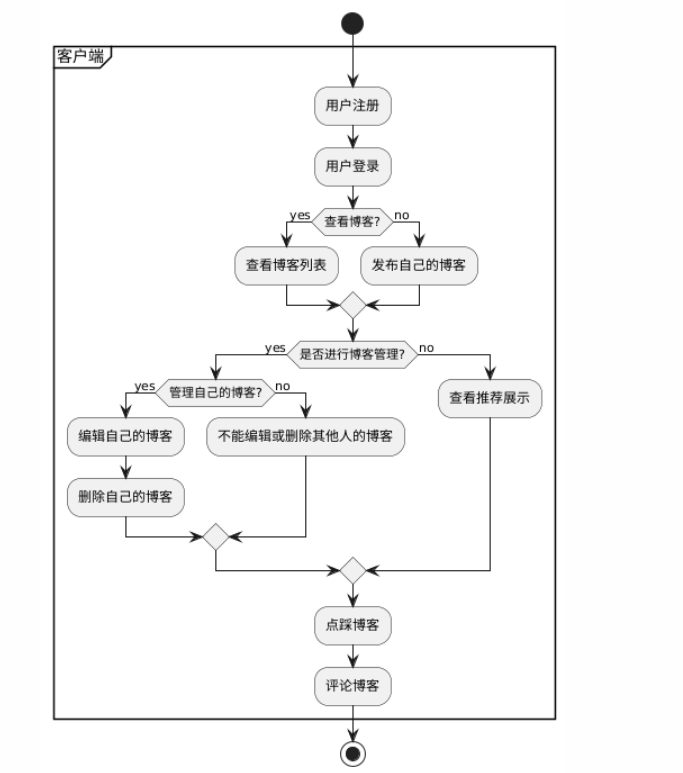
**用途：**

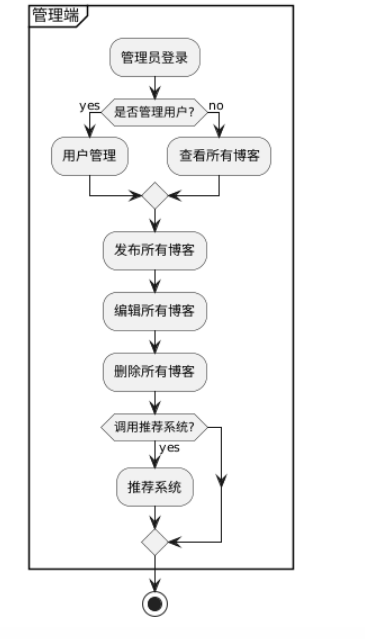
1. 为开发人员提供明确的指导，直接进行编码。
2. 确保系统实现时不会遗漏任何细节。
3. 减少开发过程中的沟通成本，提高效率。

**特点：**

* 详细说明每个模块的内部逻辑、数据库结构和具体算法。
* 涉及到类、函数、接口、数据表、流程图等技术细节。
* 针对系统功能逐一分解，确保每个细节都有方案。

### 5.1 项目流程图（分用户端和管理端）

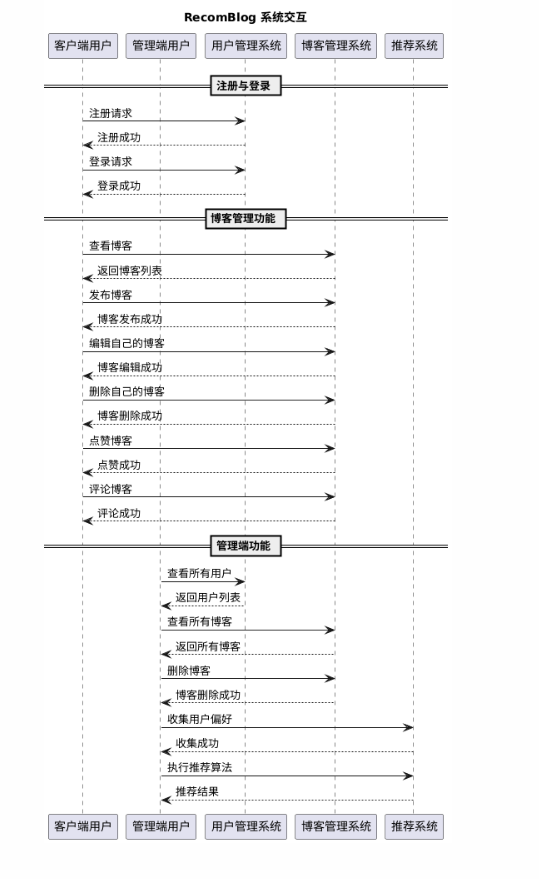




### 5.2 项目交互图

交互图的作用是：

* **可视化**：通过图形化的方式展示对象间的交互，帮助团队理解系统的动态行为。
* **沟通工具**：为开发人员和设计人员提供了一种有效的沟通方式，确保对系统功能的理解一致。
* **文档化**：作为系统设计文档的一部分，交互图记录了系统的行为和交互模式，便于后续的维护和更新。

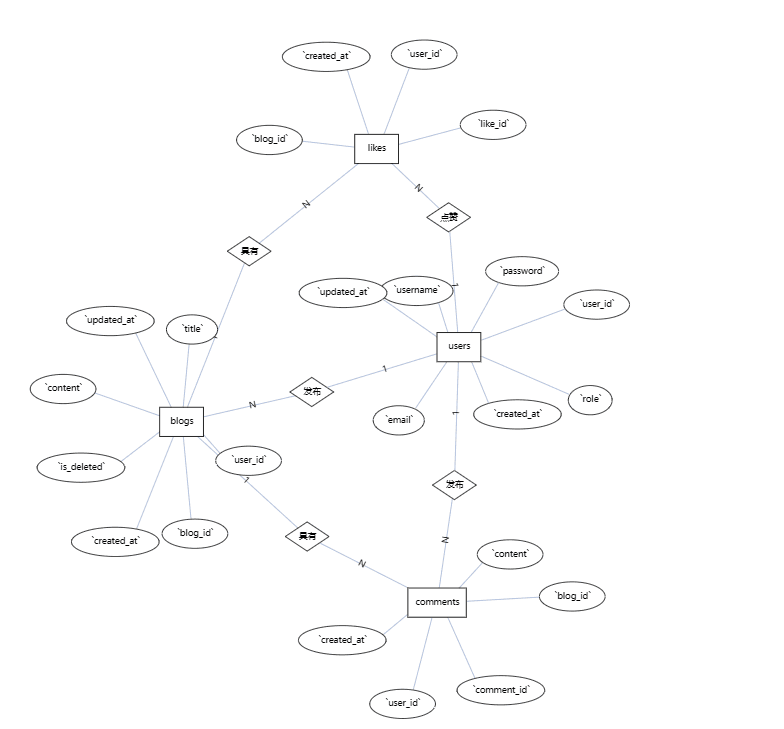


### 5.3 详细设计说明书

|  |
| --- |
| **详细设计说明书**  **1. 数据库设计**   * **用户表 (Users)**：   + user\_id (主键)，username，password，email，role（普通用户或管理员）。 * **博客表 (Blogs)**：   + blog\_id (主键)，user\_id (外键)，title，content，created\_at，updated\_at，is\_deleted。 * **评论表 (Comments)**：   + comment\_id (主键)，blog\_id (外键)，user\_id (外键)，content，created\_at。 * **点赞表 (Likes)**：   + like\_id (主键)，blog\_id (外键)，user\_id (外键)，created\_at。   **历史记录表（chat\_responses）**  Id(主键)，prompt，response，create\_at  （注明：gpt通话以及推荐模块是通过调用api实现的，没有通过数据库实现）  **2. 功能设计**  **2.1 用户管理**   * **注册**：验证用户名和邮箱唯一性，保存用户信息到数据库。 * **登录**：通过验证用户名和密码允许用户访问系统。 * **权限管理**：根据角色动态分配访问权限。   **2.2 博客管理**   * **普通用户**：   + 查看博客列表（分页显示）。   + 创建博客：输入标题和内容。   + 编辑和删除自己的博客。   + 点赞和评论任何博客。 * **管理员**：   + 管理所有博客（包括发布、编辑、删除任何博客）。   + 评论和点赞任何博客。   **2.3 偏好设置**   * 用户可以通过点击“猜你喜欢”里面对应的电影标签传给gpt相应的信息。   **2.4 推荐系统**   * 根据用户之前偏好设置。 * 后台实现推荐算法（如协同过滤）。   **3. 流程设计**   * **注册/登录流程**：   + 用户输入信息，后台验证通过后进入系统。 * **博客管理流程**：   + 用户进入博客管理模块后，选择发布、编辑、删除或查看博客。 * **推荐流程**：   + 用户查看推荐页面，后台根据用户偏好展示内容。 |

## 六、数据库设计

### 6.1 概念设计



### 6.2 外部模型

对于不同的用户（根据角色属性role分为admin（管理员）和用户（client）），前者拥有整个管理端子系统的增删查改权限，二应聘者只能在被限制的客户子系统进行有限的操作，无法登入管理页面。

### 6.3 物理模型

我们选用mysql来实现对数据库的访问。根据上述逻辑模型，我们初步确定表的结构

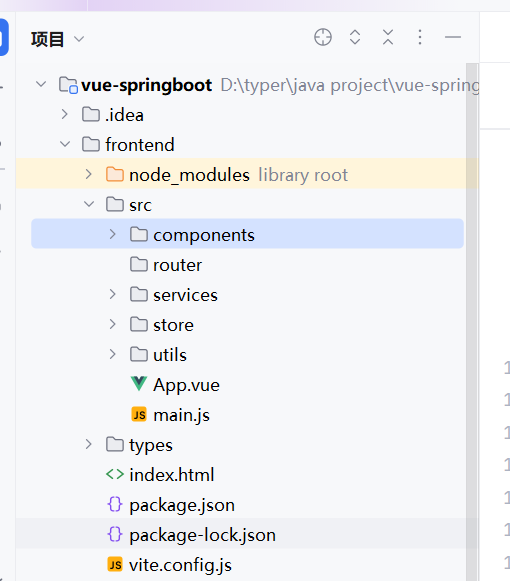
|  |
| --- |
| **1. Users 表**   * **目的**: 存储用户信息。 * **字段**:   + user\_id: 用户的唯一标识符（主键）。   + username: 用户名，唯一且不为空。   + password: 存储用户密码（应加密）。   + email: 用户邮箱，唯一且不为空。   + role: 用户角色（管理员或客户）。   + created\_at: 记录创建时间。   + updated\_at: 记录最后更新时间。   **2. Blogs 表**   * **目的**: 存储用户发布的博客文章。 * **字段**:   + blog\_id: 博客的唯一标识符（主键）。   + user\_id: 发布博客的用户（外键，关联到 Users 表）。   + title: 博客标题。   + content: 博客内容。   + created\_at: 记录创建时间。   + updated\_at: 记录最后更新时间。   + is\_deleted: 标记博客是否已删除。   **3. Comments 表**   * **目的**: 存储用户对博客的评论。 * **字段**:   + comment\_id: 评论的唯一标识符（主键）。   + blog\_id: 评论所属的博客（外键，关联到 Blogs 表）。   + user\_id: 评论的用户（外键，关联到 Users 表）。   + content: 评论内容。   + created\_at: 记录创建时间。   **4. Likes 表**   * **目的**: 存储用户对博客的点赞信息。 * **字段**:   + like\_id: 点赞的唯一标识符（主键）。   + blog\_id: 被点赞的博客（外键，关联到 Blogs 表）。   + user\_id: 点赞的用户（外键，关联到 Users 表）。   + created\_at: 记录创建时间。   + setting\_value: 设置的值。  1. **chat\_responses表**   **目的：存储prompt和response**  **字段**：Id(主键)，prompt，response，create\_at  **表关系概述**   * + 一个用户可以有多个博客（Users ↔ Blogs）。   + 一个博客可以有多个评论（Blogs ↔ Comments）。   + 一个用户可以有多个评论（Users ↔ Comments）。   + 一个博客可以有多个点赞（Blogs ↔ Likes）。 |

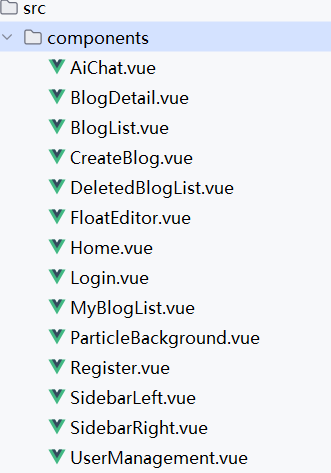
## 七、编码实现

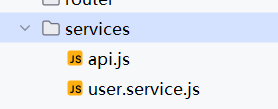
### 7.1 项目代码目录树

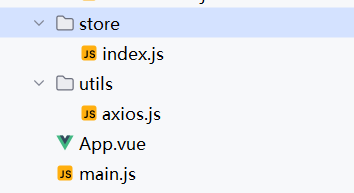
本项目基于springboot开发在代码层次结构如下:

前端：

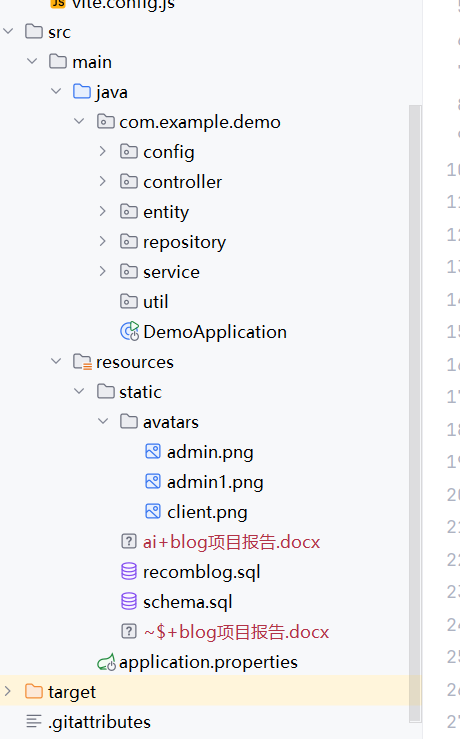


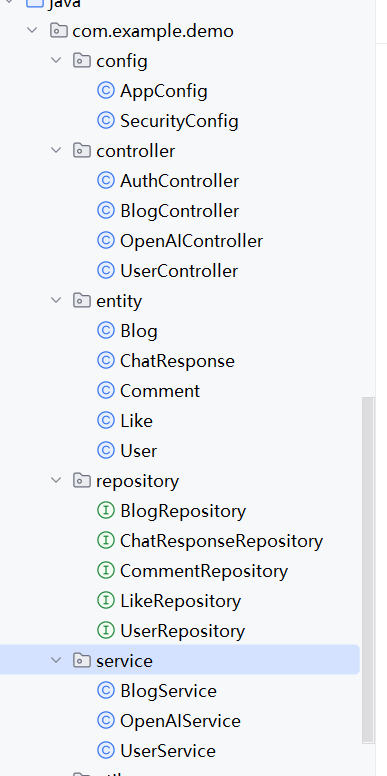






后端：

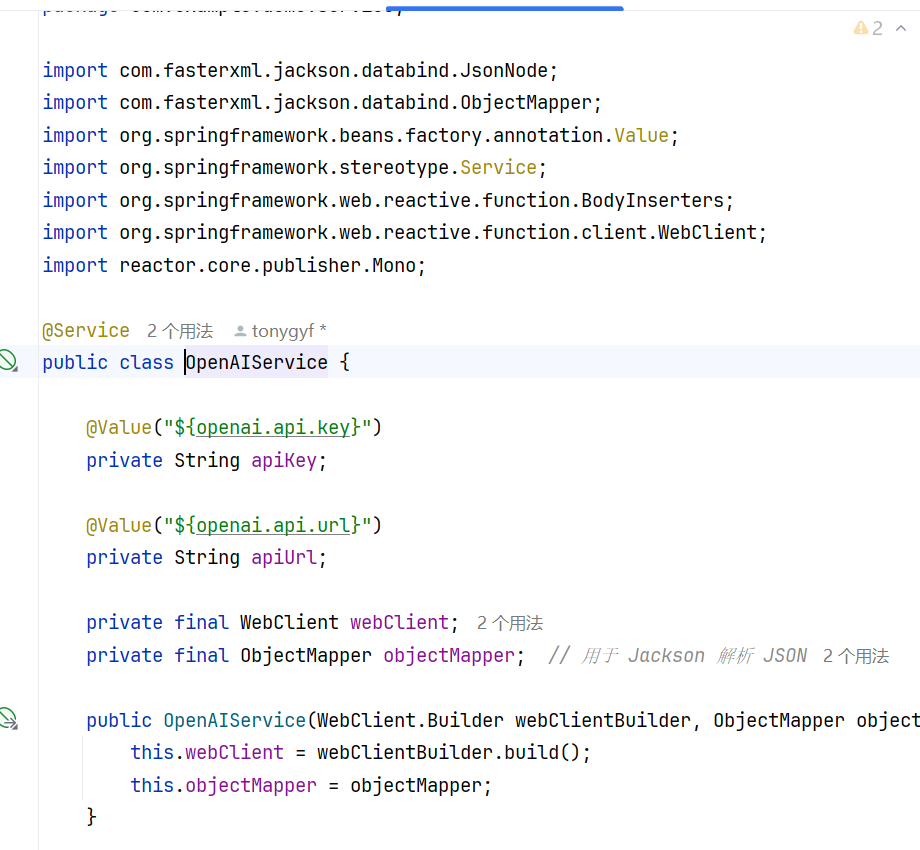




### 7.2 关键代码功能阐述（关于使用api集成的详细过程）







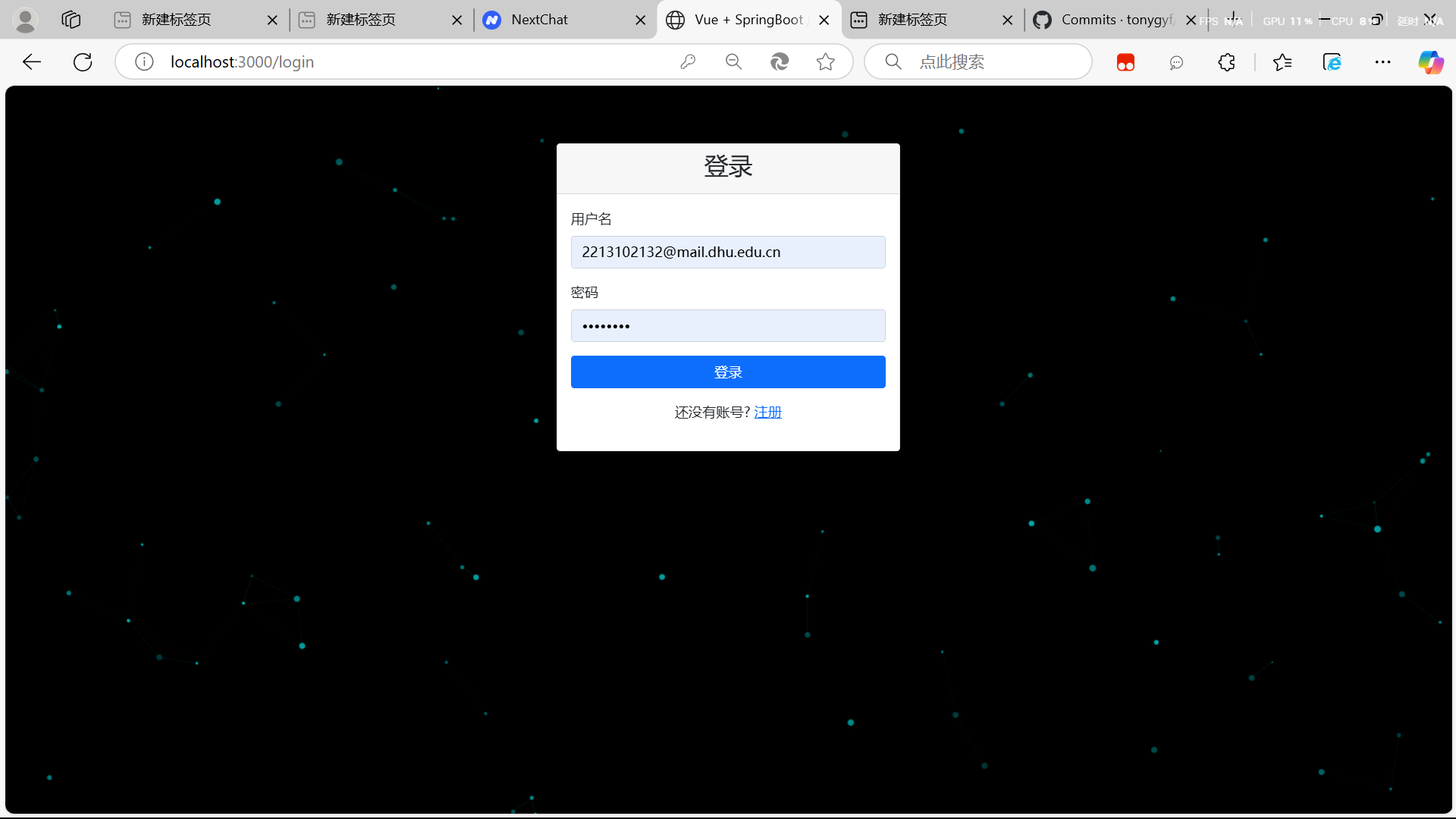


我使用webclient进行通信，前端发出请求，后端发送给接口地址，返回的json首先经过后端，后端的service进行格式化处理，仅输出content，再转发给前端。

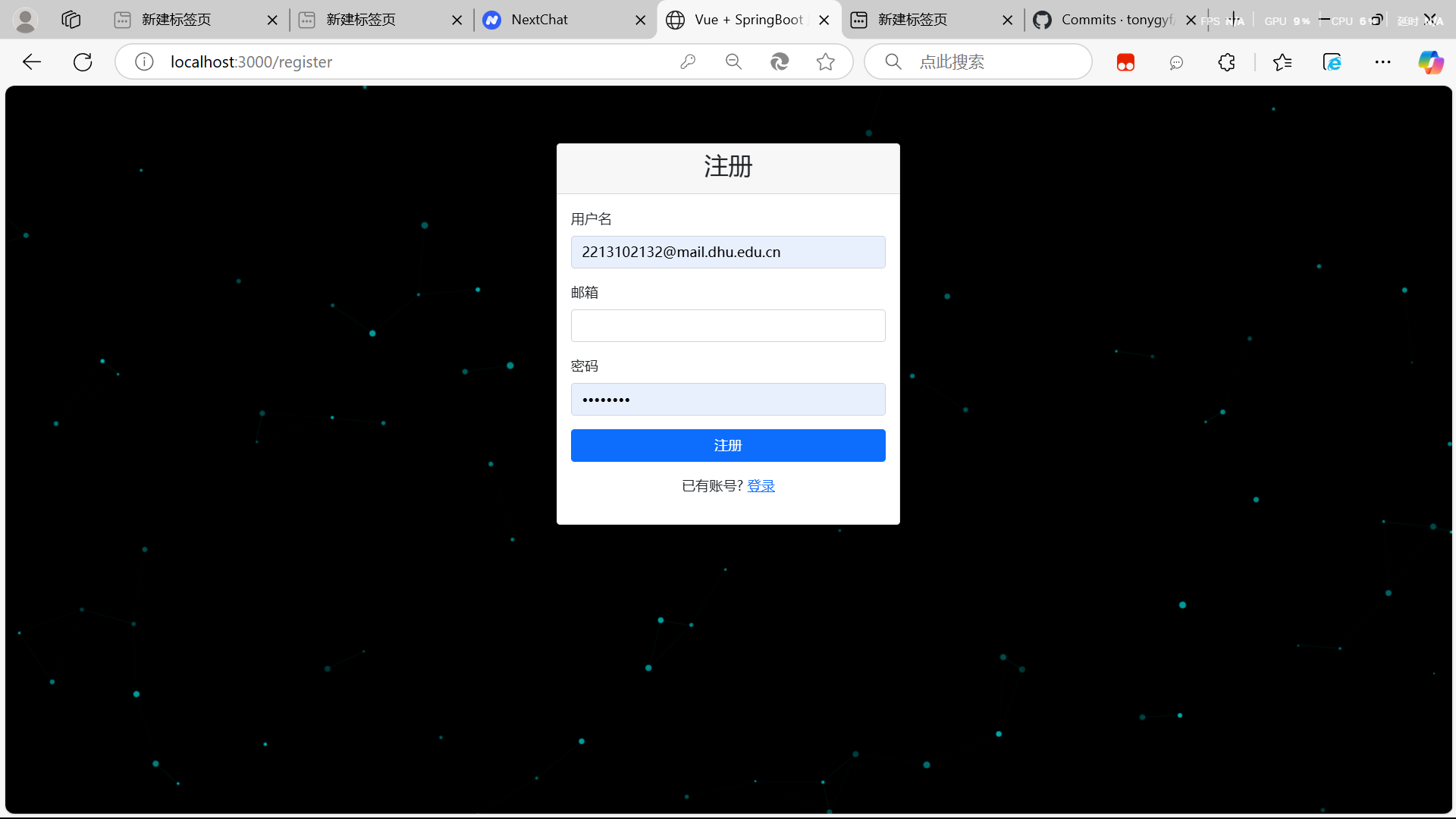
### 7.3 项目运行截图以及说明

登录注册（可以根据登录用户的role（admin、client）进行跳转到客户端或者管理端）：

登录页面：



注册页面（默认role为client，管理端才能添加或修改成admin）：



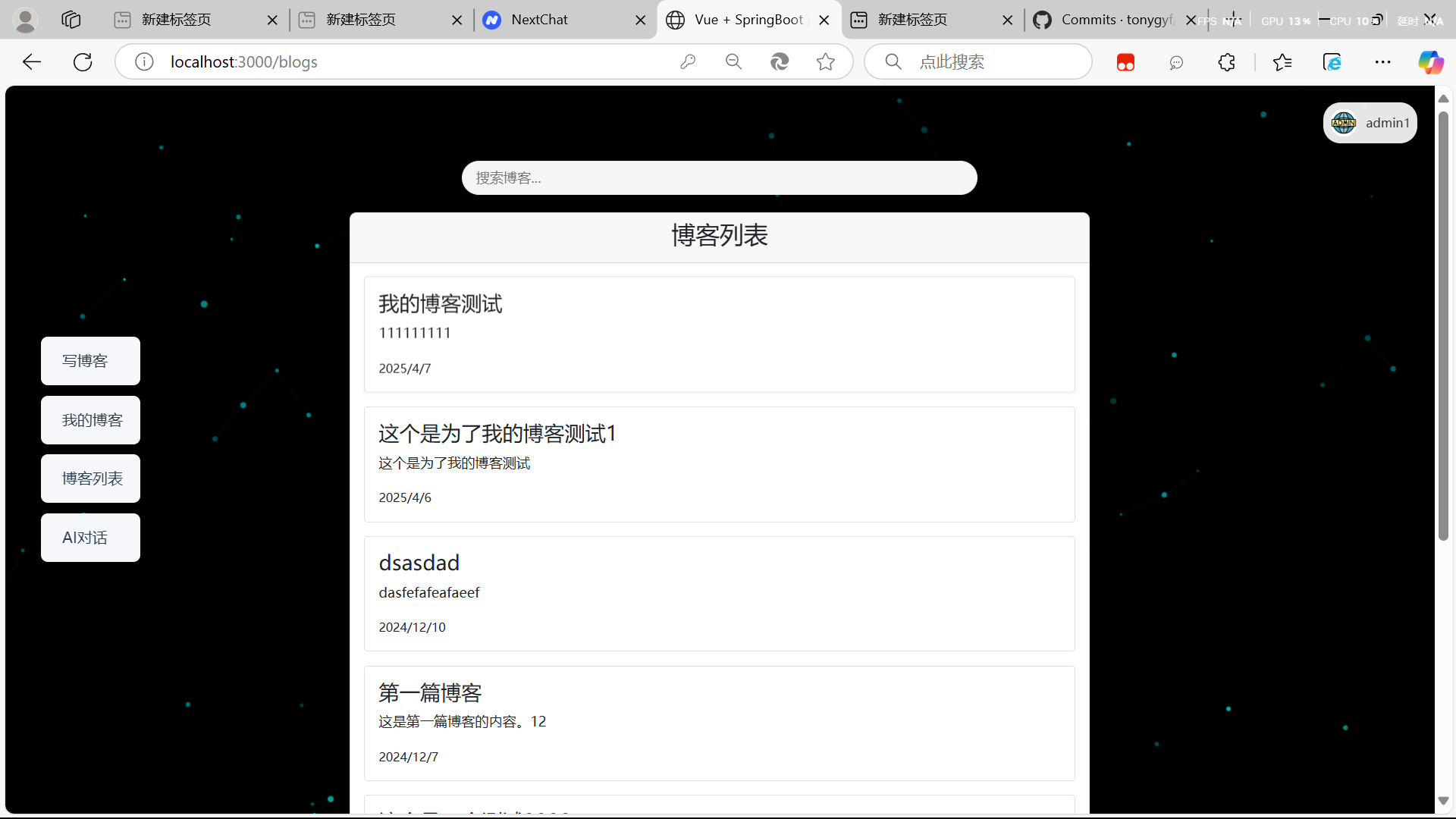




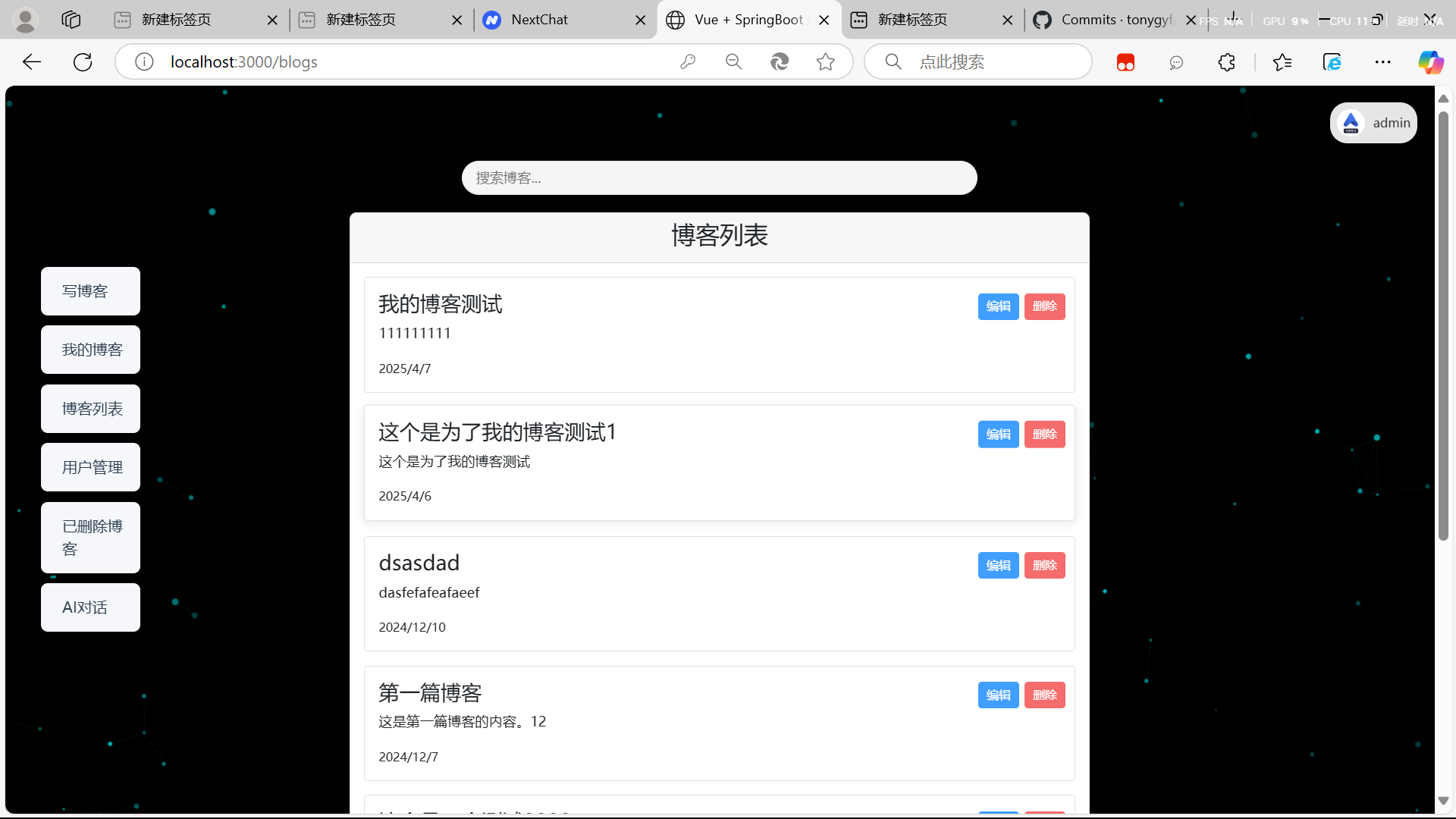
（通过app.vue,让除了注册登录页面外都有）

分别是左侧栏右上角的用户信息和背景粒子效果，而且我通过验证role，使得有不同表现。

下面是client：

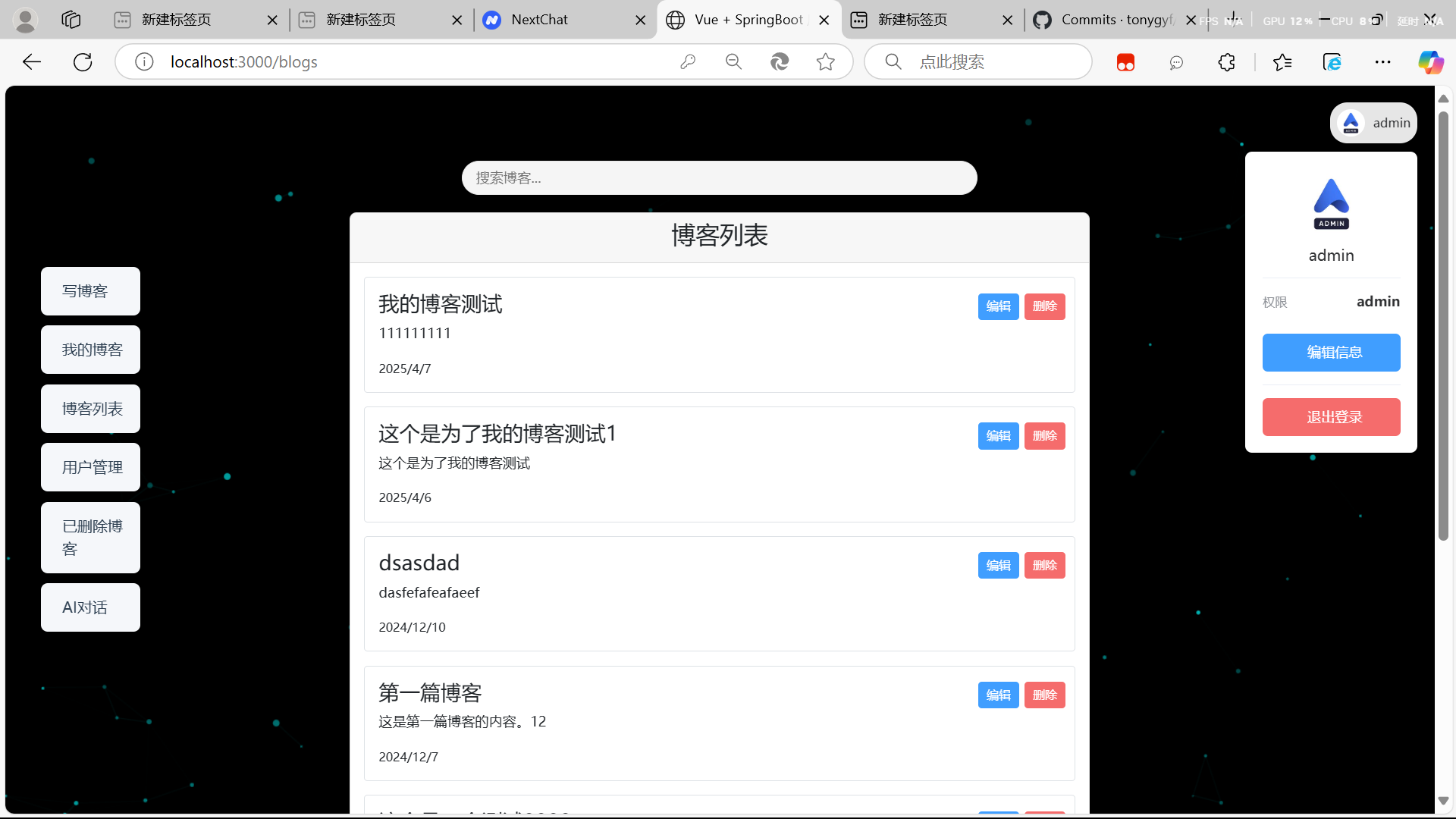


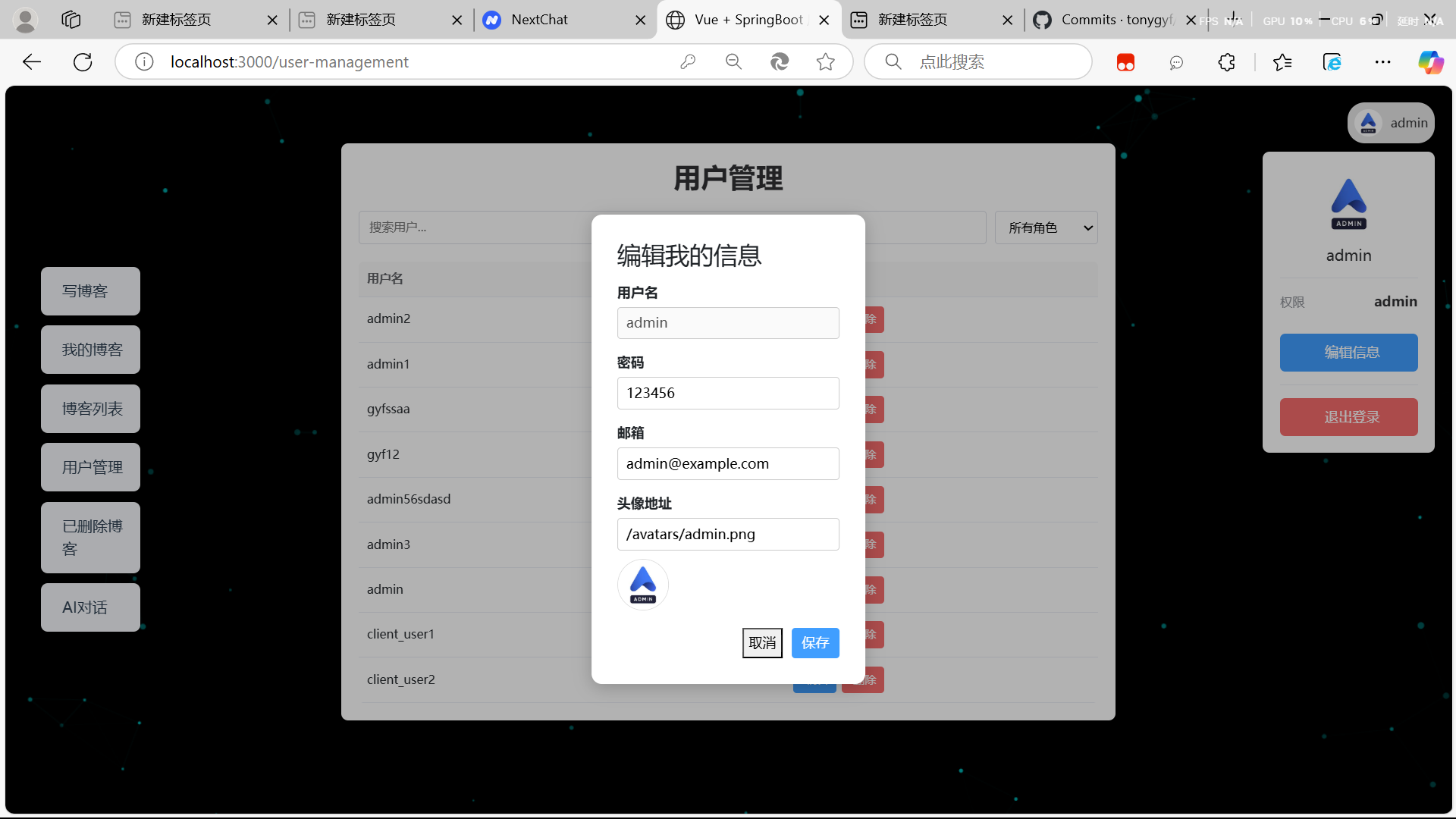
下面是admin



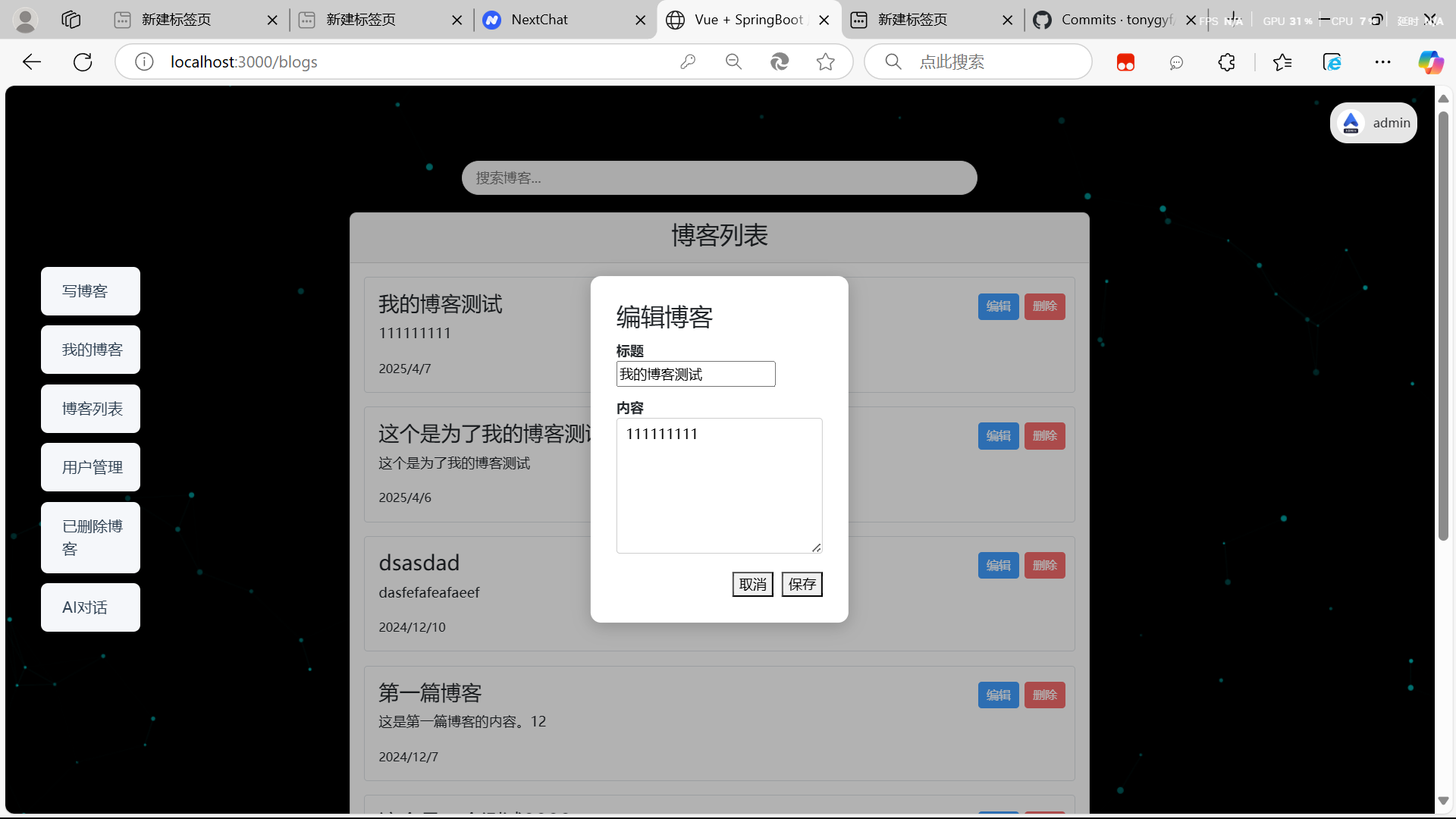
明显看到许多不同，左侧栏中admin多了用户管理和对删除博客的管理。

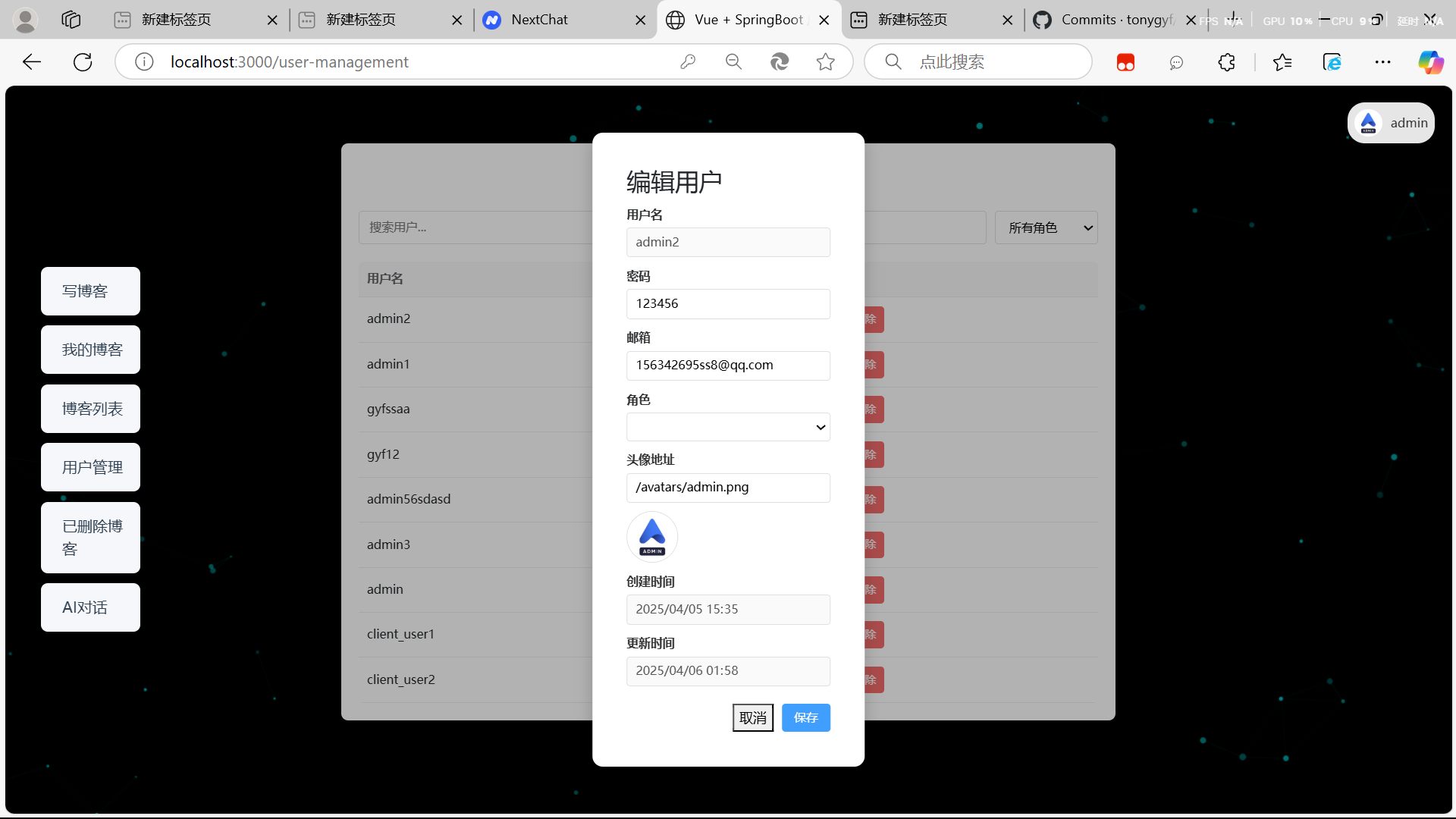
其中右侧sidebar组件用处为下：





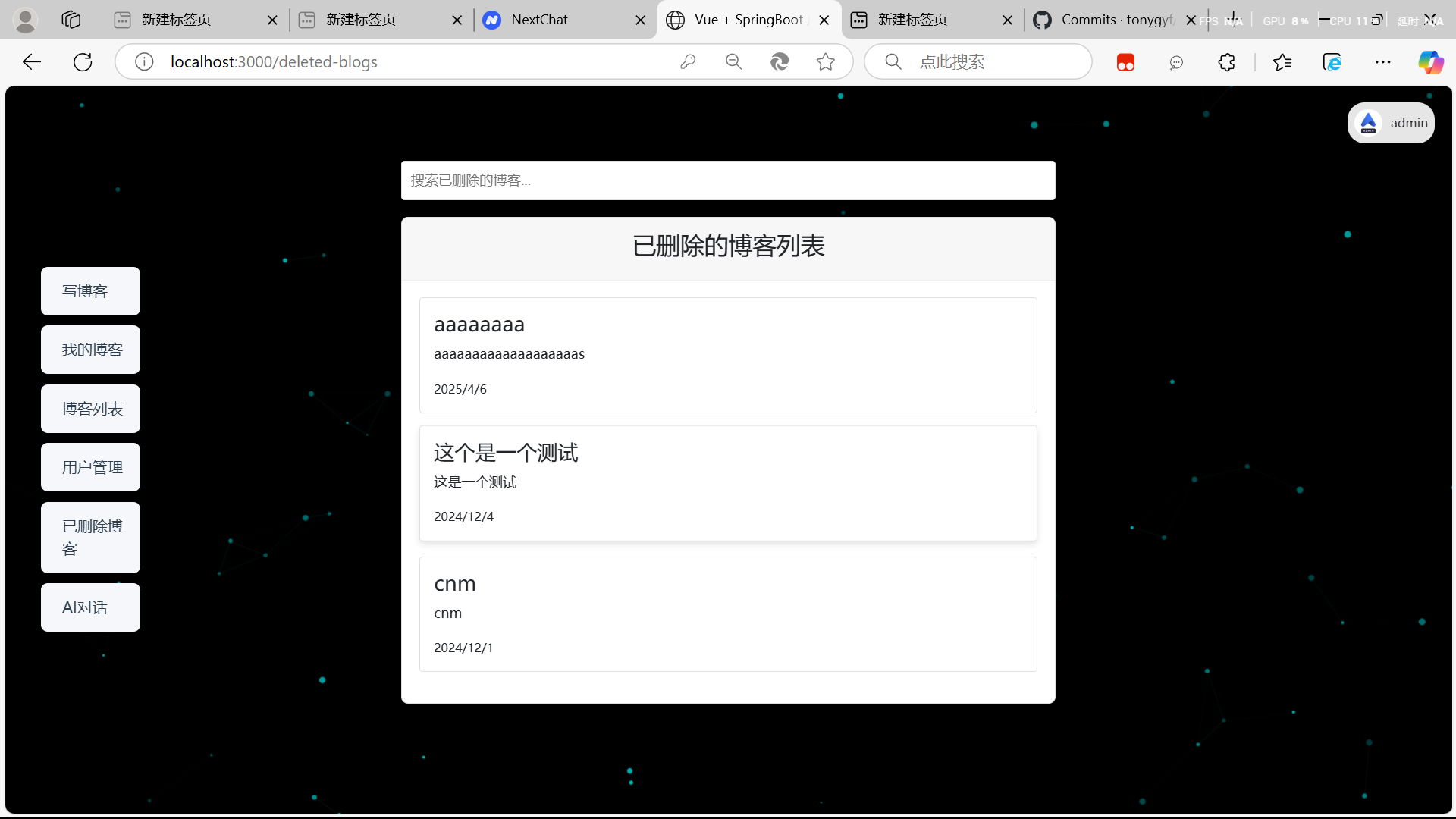
它和编辑博客，编辑用户一样都使用了浮窗





查找user和查找blog都一样风格，使用上面的搜索栏。

最重要的是下面已经删除博客这一项：



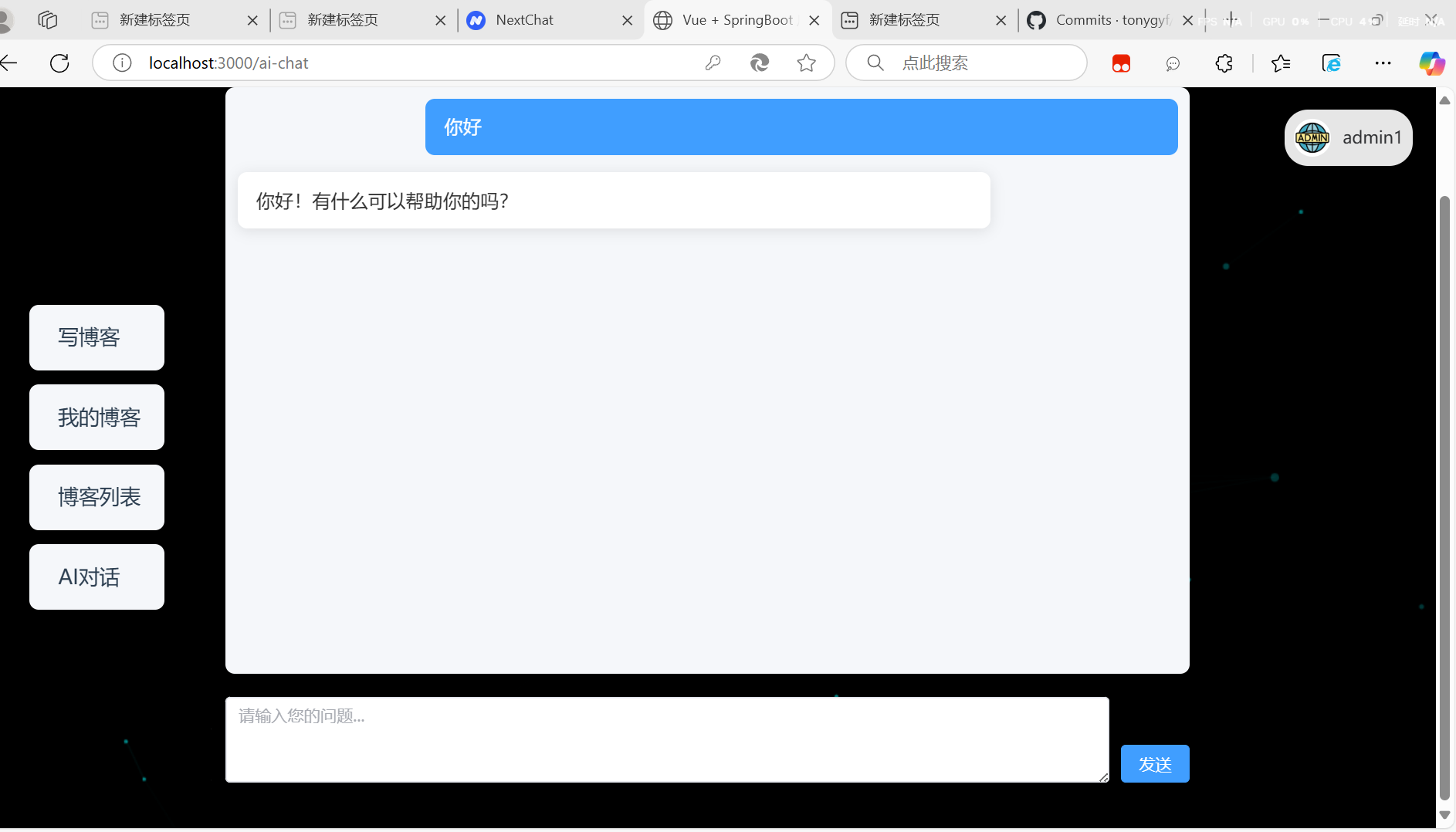
这一项非常重要，因为之前我在别的课上演示过相关的业务逻辑，发现删除一个博客，如果不删除相关点赞和评论就会发生异常，由于关系型数据库的原因。

因此数据库中给blog添加一个isdeleted的boolean值，删除的话直接将值变为true这样展示时只展示Boolean为false就可以实现删除的业务逻辑。

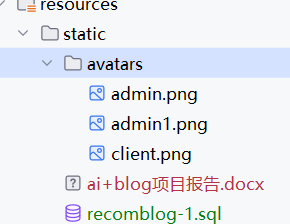
具体看之前的各个页面你会发现，admin比client多了许多可选，还包括博客列表的编辑删除，以及博客详情的评论的删除（经过深思熟虑，我认为点赞的管理不需要，直接每个用户自己处理）



Ai对话界面如下：



你会发现我的用户信息中有头像，是因为数据库中存储了相对路径，我将一些图片存在



并且设置spring.web.resources.static-locations=classpath:/static/还有



成功达成访问后端图片，以后等我能力成熟，我将会使用mongdb等数据库，这样会更快捷。

由于vue比较多就不一一赘叙，我的vue起名字非常通俗，应该能直接领悟。

## 八、测试报告

### 8.1 针对项目各个功能进行测试

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1. 测试目标**  确保博客应用程序的各项功能正常运行，用户体验良好，且系统稳定。  **2. 测试范围**   * 博客浏览和发布 * 评论点赞功能 * 集成 GPT 通话 * 个人信息查看 * AI 电影推荐 * 登录注册 * 评论管理 * 博客管理 * 点赞管理 * 用户管理   **3. 测试类型**   * 功能测试 * 性能测试 * 安全测试 * 用户界面测试 * 兼容性测试 * 回归测试   **4. 测试环境**   * 操作系统：Windows * 浏览器：Chrome, Firefox, Safari * 数据库：MySQL * 服务器：Tomcat * API 测试工具：Postman   **5. 测试用例**   | **功能** | **测试用例描述** | **预期结果** | | --- | --- | --- | | 博客浏览发布 | 1. 浏览博客列表  2. 发布新博客 | 1. 能够查看所有博客  2. 新博客成功发布并显示在列表中 | | 博客评论点赞 | 1. 添加评论  2. 点赞评论 | 1. 评论成功添加并显示  2. 评论点赞数增加 | | 集成 GPT 通话 | 1. 输入提示并发送请求  2. 接收 GPT 响应 | 1. 能够发送提示  2. 正确接收到 GPT 的响应 | | 个人信息查看 | 1. 登录用户账户  2. 查看个人信息 | 1. 登录成功  2. 显示正确的个人信息 | | AI 电影推荐 | 1. 请求电影推荐<br>2. 查看推荐列表 | 1. 成功获取电影推荐  2. 推荐列表正确显示 | | 登录注册 | 1. 注册新用户  2. 登录用户账户  3. 登录失败的情况 | 1. 注册成功  2. 登录成功  3. 提示错误信息 | | 评论管理 | 1. 查看评论列表  2. 删除评论  3. 编辑评论 | 1. 显示所有评论  2. 评论成功删除  3. 评论成功编辑 | | 博客管理 | 1. 查看博客列表  2. 编辑博客  3. 删除博客 | 1. 显示所有博客  2. 博客成功编辑  3. 博客成功删除 | | 点赞管理 | 1. 查看点赞列表  2. 取消点赞 | 1. 显示所有点赞记录  2. 点赞成功取消 | | 用户管理 | 1. 查看用户列表  2. 编辑用户信息  3. 删除用户 | 1. 显示所有用户  2. 用户信息成功编辑  3. 用户成功删除 |   **6. 测试执行**   * **执行人员**：QA 团队成员 * **执行时间**：根据项目进度安排 * **执行方式**：手动测试与自动化测试相结合   **7. 缺陷管理**   * 使用缺陷跟踪工具（如 JIRA、Bugzilla）记录和管理发现的缺陷。 * 定期评审缺陷状态，确保及时修复。   **8. 测试报告**   * 测试完成后，生成测试报告，包含测试覆盖率、缺陷统计、测试结果等信息。   **9. 风险管理**   * 针对可能的风险（如功能未按预期工作、性能问题等）制定应对措施。 |

### 8.2 测试结果以及总结

经过检测，项目功能大致实现，但是发现数据库的外键问题导致有些博客（被评论或者点赞），不可以直接删除，而是需要通过删除相关数据后再删除，这个问题应该是数据库选用设计的问题，总体来说问题不大。

**结论**

通过上述测试计划，可以系统地验证博客应用程序的各项功能，确保其在上线前达到预期的质量标准。

## 九、项目计划

### 9.1 质量管理计划

软件项目质量管理计划是一个文档，旨在确保软件项目在开发和交付过程中达到预期的质量标准。它通常包括以下几个关键要素：

1. **质量目标**：明确项目的质量标准和目标，例如功能完整性、性能、可靠性等。
2. **质量保证措施**：描述将采取的预防性措施，以确保项目在开发过程中遵循质量标准。这可能包括代码审查、单元测试、集成测试等。
3. **质量控制方法**：定义用于监控和评估项目质量的工具和技术，例如缺陷跟踪、测试计划和审计。
4. **角色与责任**：明确团队成员在质量管理中的角色和责任，包括项目经理、开发人员、测试人员等。
5. **质量评审与审核**：规定定期进行的质量评审和审核活动，以确保项目在各个阶段都符合质量要求。
6. **风险管理**：识别可能影响项目质量的风险，并制定相应的应对策略。
7. **文档和报告**：指定质量管理相关文档的格式和存储方式，以及定期生成的质量报告。

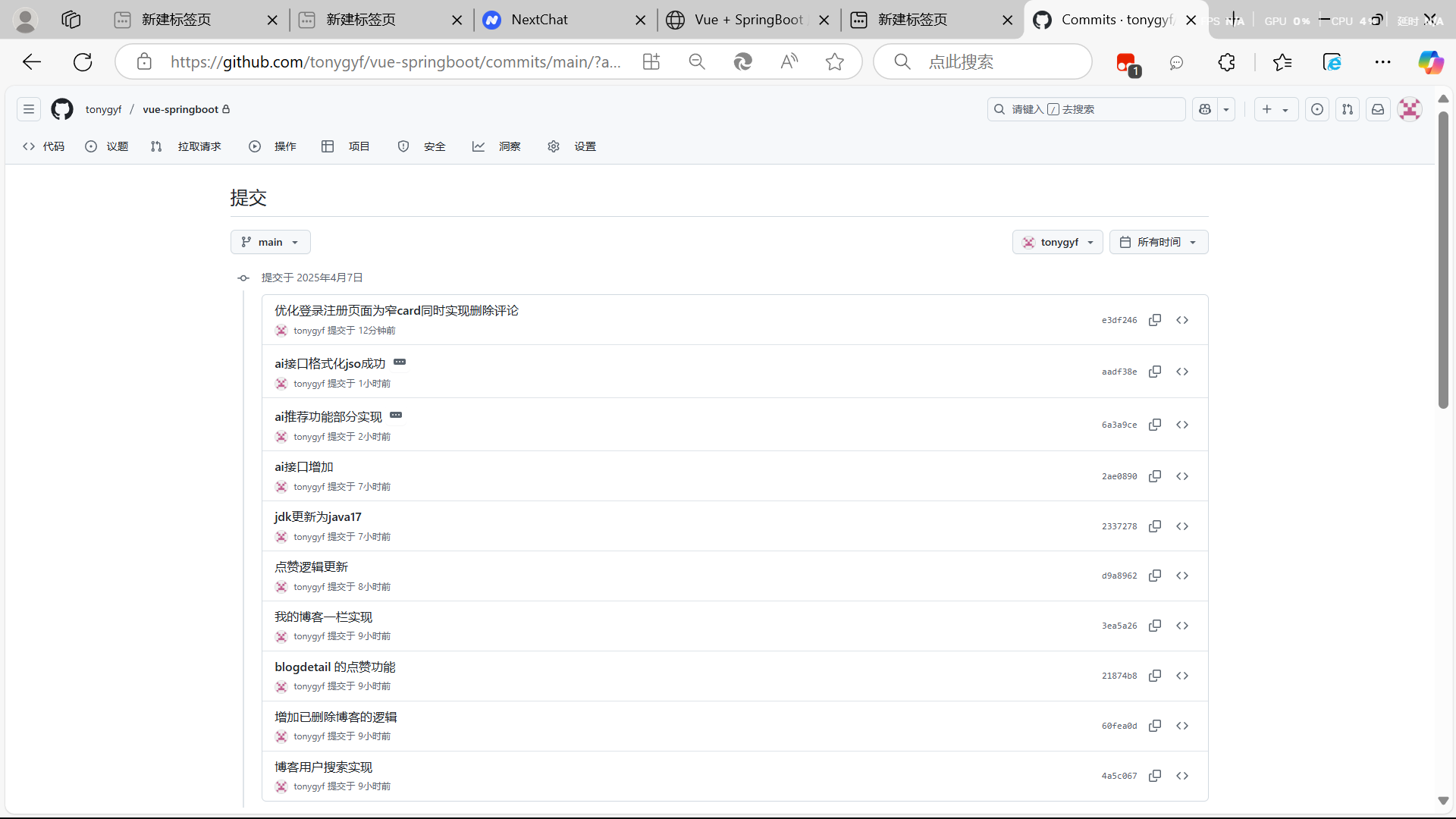
通过制定详细的质量管理计划，团队可以更有效地控制软件项目的质量，降低缺陷率，提高客户满意度。

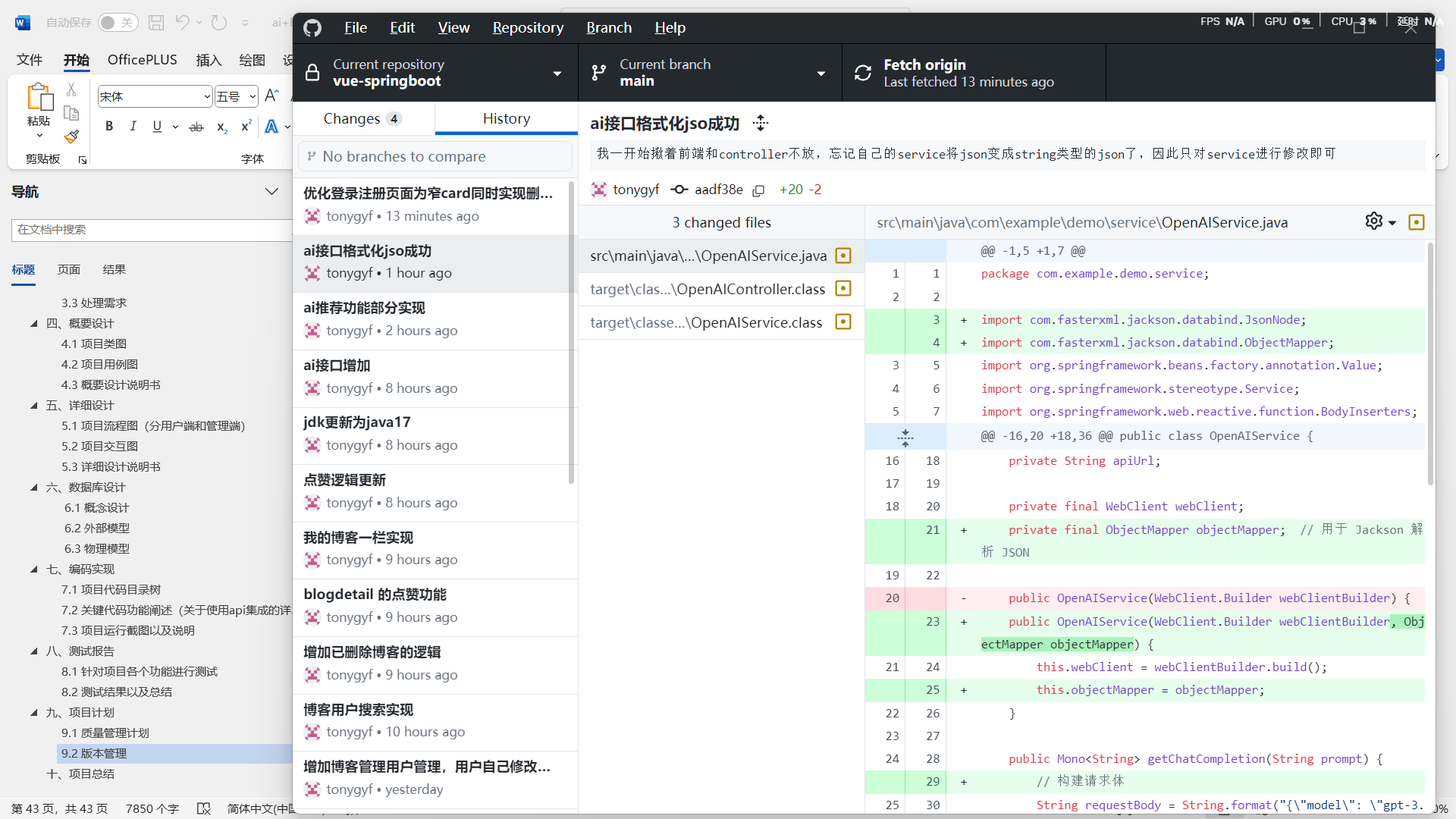
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9.3 质量管理计划 **9.3.1 质量目标**  确保软件功能符合需求规格，功能实现率达到95%。  系统的响应时间在95%的情况下不超过2秒。  用户满意度调查中，目标满意度达到80%以上。  本项目采用因果分析图    对于本项目，编制质量计划采用因果分析图，描述相关的各种原因和子原因如何产生潜在问题或影响，将影响质量问题的“人员、设备、参考资料、方法、环境”等各方面的原因进行细致的分解，方便地在质量计划中制定相应的预防措施。其次，质量计划中还必须确定有效的质量管理体系，明确质量监理人员对项目质量负责和各级质量管理人员的权限。项目中采用计划——执行——检查——措施的质量环，质量计划中将质量环明确落实到各责任单位，才能保证质量计划的有效实施。  **9.3.2 质量控制措施**  代码审查： 每个功能模块完成后进行同行评审，确保代码质量和可维护性。  自动化测试：建立全面的单元测试和集成测试覆盖，目标覆盖率达到80%。  性能测试： 在发布前进行压力测试和负载测试，确保系统在高并发情况下的稳定性。  **9.3.3 质量保障**  我们每人相应的工作进展都会在群中公示以便于其他对接的成员进行质量评估，在过程中不断讨论问题和改进措施。  我们还会将软件给身边的同学老师介绍，收集用户反馈，持续改进产品质量。 9.3.4质量保证活动  |  |  |  |  | | --- | --- | --- | --- | | ***项目*** | ***具体描述*** | ***计划*** | ***实际*** | |  | 需求检查 | 5 | 6 | | 系统总体设计检查 | 4 | 2 | | **缺陷排除率（缺陷数/KLOC）** | 详细设计复核 | 25 | 19 | | 详细设计检查 | 19 | 15 | | 代码复核 | 50 | 45 | | 代码检查 | 13 | 13 | | 编译 | 10 | 13 | |  | 单元测试 | 12 | 12 | | 系统集成 | 2 | 3 | | 系统测试 | 2 | 3 |  9.3.5产品审计 产品审计需要由质量保证人员来进行，检查项目产品是否达到了质量目标。  质量保证人员可以有选择性地审计项目生存期中创建工作产品，以验证是否符合适当  的标准，是否进行了质量检查，质量审计表如下：   |  |  |  |  | | --- | --- | --- | --- | | ***项*** | ***审计对象*** | ***审计阶段*** | ***参照标准*** | | **1** | 软件项目计划 | 计划结束 | 企业质量体系 | | **2** | 软件配置管理计划 | 计划结束 | 企业质量体系 | | **3** | 软件质量保证计划 | 计划结束 | 企业质量体系和项目规划 | | **4** | 总体设计文档 | 设计结束 | 企业质量体系和项目规划 | | **5** | 详细设计文档 | 计划结束 | 企业质量体系和项目规划 | | **6** | 数据库表和编码规范 | 计划结束 | 企业质量体系和项目规划 |  9.3.6 测试计划  1. 建立每个测试阶段的目标确定活动的进度和职责。 2. 确定工具、设备和测试库的可用性。 3. 建立用于计划和进行测试以及报告测试结果的规程和标准。   制定衡量测试成功与完成的准则。 |  |

### 9.2 版本管理

本次项目使用git 进行版本管理，大大优化了开发进程，有的时候代码写乱了，我直接一个reset –hard就能回到原来的提交。

下面是我的push记录





## 十、项目总结

本次项目很有意义，整个项目基本是我之前做组长的一个项目时的业务逻辑。但是由于之前的项目是thymeleaf的，并没有前后端分离。我也缺少这方面的经验。这学期跟着黄老师的组进行学习，掌握一点要领并使用多个ai进行辅助开发，终于将项目完成，很有意义。

我将项目重置成前后端分离的应用，前端使用vuejs，后端使用springboot，java使用17，数据库使用mysql，调用的api接口是chatanywhere的，很有意思。我也逐步掌握了前后端开发和git开发的方法，我认为这个十分有意义。