

**毕业设计**

**题 目 基于QAnything的智能问答系统设计与实现**

**英文题目：Design and Implementation of Intelligent Question-Answering System Based on QAnything**

**学生姓名**： **徐阳阳 申请学位门类： 工学**

**学 号： 2021213737**

**专 业：** **软件工程G**

**学 院**： **软件学院**

**指导教师：** **王红玲** **职称：** **副教授**

**二0二五年 四 月三 十 日**

作 者 声 明

本人以信誉郑重声明：所呈交的学位毕业设计（论文），是本人在指导教师指导下由本人独立撰写完成的，没有剽窃、抄袭、造假等违反道德、学术规范和其他侵权行为。文中引用他人的文献、数据、图件、资料均已明确标注出，不包含他人成果及为获得东华理工大学或其他教育机构的学位或证书而使用过的材料。对本设计（论文）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本毕业设计（论文）引起的法律结果完全由本人承担。

本毕业设计（论文）成果归东华理工大学所有。

特此声明。

毕业设计（论文）作者（签字）：

签字日期： 年 月 日

本人声明：该学位论文是本人指导学生完成的研究成果，已经审阅过论文的全部内容，并能够保证题目、关键词、摘要部分中英文内容的一致性和准确性。

学位论文指导教师签名：

年 月 日

摘 要

近年来，随着自然语言处理（NLP）和深度学习技术的突破性进展，知识库问答系统已成为智能信息检索领域的研究热点。然而，现有系统普遍存在两个关键局限：一是高度依赖云端服务，无法满足数据敏感场景下的离线使用需求；二是文件格式兼容性有限，难以处理多样化的知识载体。这些问题严重制约了知识管理系统在现实场景中的应用效果。

本课题基于 QAnything 的架构思想，设计并实现了一个 本地化部署的智能问答系统，支持多格式文档（PDF、Word、Excel、图片等）的解析与问答。系统采用 Vue.js 构建前端交互界面，NestJS 实现后端服务，结合 MySQL 进行知识存储与管理，并集成大语言模型（LLM）实现自然语言问答功能。

本研究的实践价值体现在三个方面：首先，为各类组织构建安全可靠的私有知识库提供解决方案；其次，通过智能化的知识提取和检索技术，显著提升组织内部的知识共享效率；最后，其ChatBot分享模式，可以实现多个用户使用同一套模型配置进行问答，利于信息共享。研究成果将推动知识管理技术向更更智能、更易用的方向发展。

关键词：人工智能； 智能问答系统； 知识库问答； ChatBot； QAnything

**ABSTRACT**

In recent years, with the breakthrough progress of natural language processing (NLP) and deep learning technologies, knowledge base question-answering systems have become a research hotspot in the field of intelligent information retrieval. However, the existing systems generally have two key limitations: First, they are highly dependent on cloud services and cannot meet the usage requirements in data-sensitive scenarios; Second, the compatibility of file formats is limited, making it difficult to handle diverse knowledge carriers. These problems seriously restrict the application effect of the knowledge management system in real scenarios.

Based on the architectural concept of QAnything, this project has designed and implemented a locally deployed intelligent question-answering system, which supports the parsing and question-answering of multi-format documents (PDF, Word, Excel, images, etc.). The system adopts Vue.js to build the front-end interactive interface, NestJS to implement the back-end services, combines MySQL for knowledge storage and management, and integrates the large language model (LLM) to realize the natural language question-answering function.

The practical value of this research is reflected in three aspects: Firstly, it provides solutions for various organizations to build secure and reliable private knowledge bases; Secondly, through intelligent knowledge extraction and retrieval technologies, the efficiency of knowledge sharing within the organization is significantly enhanced; Finally, its ChatBot sharing mode enables multiple users to conduct question-and-answer sessions using the same set of model configurations, which is conducive to information sharing. The research results will promote the development of knowledge management technology in a more intelligent and user-friendly direction.

**Key words:** Artificial intelligence; Intelligent question-answering system; Knowledge base Q&A; ChatBot; QAnything

目 录

[第1章 绪论 8](#_Toc196692222)

[1.1 研究背景 8](#_Toc196692223)

[1.2 研究目标与内容 8](#_Toc196692224)

[第2章 相关技术综述需求分析 8](#_Toc196692225)

[2.1 QAnything架构分析 9](#_Toc196692226)

[2.1.1 文件解析模块（PDF/Word/OCR） 9](#_Toc196692227)

[2.1.2 本地向量数据库设计 9](#_Toc196692228)

[2.1.3 问答生成逻辑 9](#_Toc196692229)

[2.2 技术选型依据 9](#_Toc196692230)

[2.2.1 前端：Vue3 + TypeScript 的优势 9](#_Toc196692231)

[2.2.2 后端：NestJS的模块化特性 9](#_Toc196692232)

[2.2.3 数据库：MySQL关系模型设计 9](#_Toc196692233)

[第3章 系统需求分析 14](#_Toc196692234)

[3.1 功能性需求 14](#_Toc196692235)

[3.1.1 多用户协作需求（Bot共享/权限管理/会话隔离） 14](#_Toc196692236)

[3.1.2 知识库管理需求（文件上传/解析/更新） 14](#_Toc196692237)

[3.1.3 问答性能需求（响应时间/并发支持） 14](#_Toc196692238)

[3.2 非功能性需求 14](#_Toc196692239)

[3.2.1 安全性（JWT鉴权/数据加密） 14](#_Toc196692240)

[3.2.2 可扩展性（模块化设计） 14](#_Toc196692241)

[3.2.3 离线部署兼容性 14](#_Toc196692242)

[第4章 系统设计 14](#_Toc196692243)

[4.1 总体架构设计 14](#_Toc196692244)

[4.1.1 系统架构图（前端/后端/数据库/LLM交互） 14](#_Toc196692245)

[4.1.2 数据流设计（用户请求→检索→生成→返回） 14](#_Toc196692246)

[4.2 核心模块设计 14](#_Toc196692247)

[4.2.1 用户与权限模块（RBAC模型设计） 14](#_Toc196692248)

[4.2.2 Bot管理模块（知识库配置/版本控制） 14](#_Toc196692249)

[4.2.3 问答引擎模块（检索增强生成流程优化） 15](#_Toc196692250)

[4.2.3 会话管理模块（MySQL表关系设计） 15](#_Toc196692251)

[4.3 数据库设计 15](#_Toc196692252)

[4.3.1 ER图与表结构（users/bots/messages等） 15](#_Toc196692253)

[4.3.2 索引优化策略 15](#_Toc196692254)

[第5章 系统实现 15](#_Toc196692255)

[5.1 前端实现 15](#_Toc196692256)

[5.1.1 多Bot切换界面（Vue3 + Pinia状态管理） 15](#_Toc196692257)

[5.1.2 实时消息展示（WebSocket/SSE） 15](#_Toc196692258)

[5.2 后端实现 15](#_Toc196692259)

[5.2.1 NestJS分层架构（Controller-Service-Repository） 15](#_Toc196692260)

[5.2.2 文件解析服务（PDF/Word/OCR集成） 15](#_Toc196692261)

[5.2.3 权限控制（@Roles装饰器实现） 15](#_Toc196692262)

[第6章 系统测试与分析 15](#_Toc196692263)

[6.1 功能测试 15](#_Toc196692264)

[6.1.1 Bot共享测试（多用户同时访问） 15](#_Toc196692265)

[6.1.2 权限控制测试（管理员vs普通用户） 16](#_Toc196692266)

[6.1.3 文件解析覆盖率测试 16](#_Toc196692267)

[第7章 总结与展望 16](#_Toc196692268)

[7.1 研究成果总结 16](#_Toc196692269)

[7.1.1 实现的功能清单 16](#_Toc196692270)

[7.1.2 创新点（多用户协作/本地化优化） 16](#_Toc196692271)

[7.2 不足与改进方向 16](#_Toc196692272)

[7.2.1 当前局限性（如大文件处理效率） 16](#_Toc196692273)

[7.2.2 未来计划（多Bot联动/边缘计算支持） 16](#_Toc196692274)

[参考文献： 16](#_Toc196692275)

第1章 绪论

1.1 研究背景

在数字化转型的浪潮下，各类组织机构（包括企业、学校及政府部门）积累了海量的内部知识资产，如技术文档、操作手册、培训材料、政策指南等。然而，如何高效地管理和利用这些知识资源正面临着严峻挑战。传统的基于关键词的检索方式存在检索精度低、结果相关性差等问题，难以满足用户对知识快速获取和精准理解的需求。相比之下，基于人工智能的问答系统能够提供更加自然、直观的人机交互体验，显著提升知识获取效率。

随着人工智能技术的发展，基于知识库的问答系统（KBQA, Knowledge-Based Question Answering）逐渐成为信息检索领域的研究热点。传统问答系统主要依赖结构化数据库或规则模板，但面对企业、教育机构中大量非结构化文档（如PDF、Word、Excel等），传统方法难以有效处理。近年来，检索增强生成（Retrieval-Augmented Generation, RAG）技术的兴起，为知识库问答提供了新思路：RAG 的核心思想是先通过检索（Retrieval）从知识库中找出相关文档片段；再通过生成（Generation）由大语言模型（LLM）生成自然语言回答。优势在于无需微调模型，直接利用现有知识库；生成结果更具事实性，减少幻觉（Hallucination）。QAnything 网易有道开源的本地知识库问答系统，其设计目标是为用户提供多格式支持、高效检索的解决方案。其核心特点包括多模态文件解析，支持 PDF、Word、Excel、图片（OCR）、PPT 等多种格式。结合文本分割、向量化技术，构建本地知识索引。

1.2 研究目标与内容

本课题基于 QAnything 的架构思想，设计并实现了一个 本地化部署的智能问答系统，支持多格式文档（PDF、Word、Excel、图片等）的解析与问答。系统采用 Vue.js 构建前端交互界面，NestJS 实现后端服务，结合 MySQL 进行知识存储与管理，并集成大语言模型（LLM）实现自然语言问答功能。

本研究的实践价值体现在三个方面：首先，为各类组织构建安全可靠的私有知识库提供解决方案；其次，通过智能化的知识提取和检索技术，显著提升组织内部的知识共享效率；最后，其ChatBot分享模式，可以实现多个用户使用同一套模型配置进行问答，利于信息共享。研究成果将推动知识管理技术向更更智能、更易用的方向发展。

第2章 技术综述

这一章将从QAnything架构、技术选型依据着手进行分析，QAnything 的整体架构采用五层结构。该架构逻辑清晰，职责明确，每一层均支持插件化替换，便于系统根据业务需求进行快速迭代。而技术选型，则是根据开源社区实时排名、易用性等等依据进行挑选，而并非会啥就用啥。

2.1 QAnything架构分析

QAnything 这款多模态问答系统整体架构采用“数据接入 → 知识加工 → 向量化存储 → 检索增强生成 → 应用交互”的五层结构，该架构逻辑清晰，职责明确，每一层均支持插件化替换，便于系统根据业务需求进行快速迭代。它的核心功能是将非结构化知识转化为可交互的问答形式。将这五个阶段划分为功能的话就是：用户上传文件、服务解析文件、数据库存储数据、系统信息进行检索、模型回答。

2.1.1 数据接入层

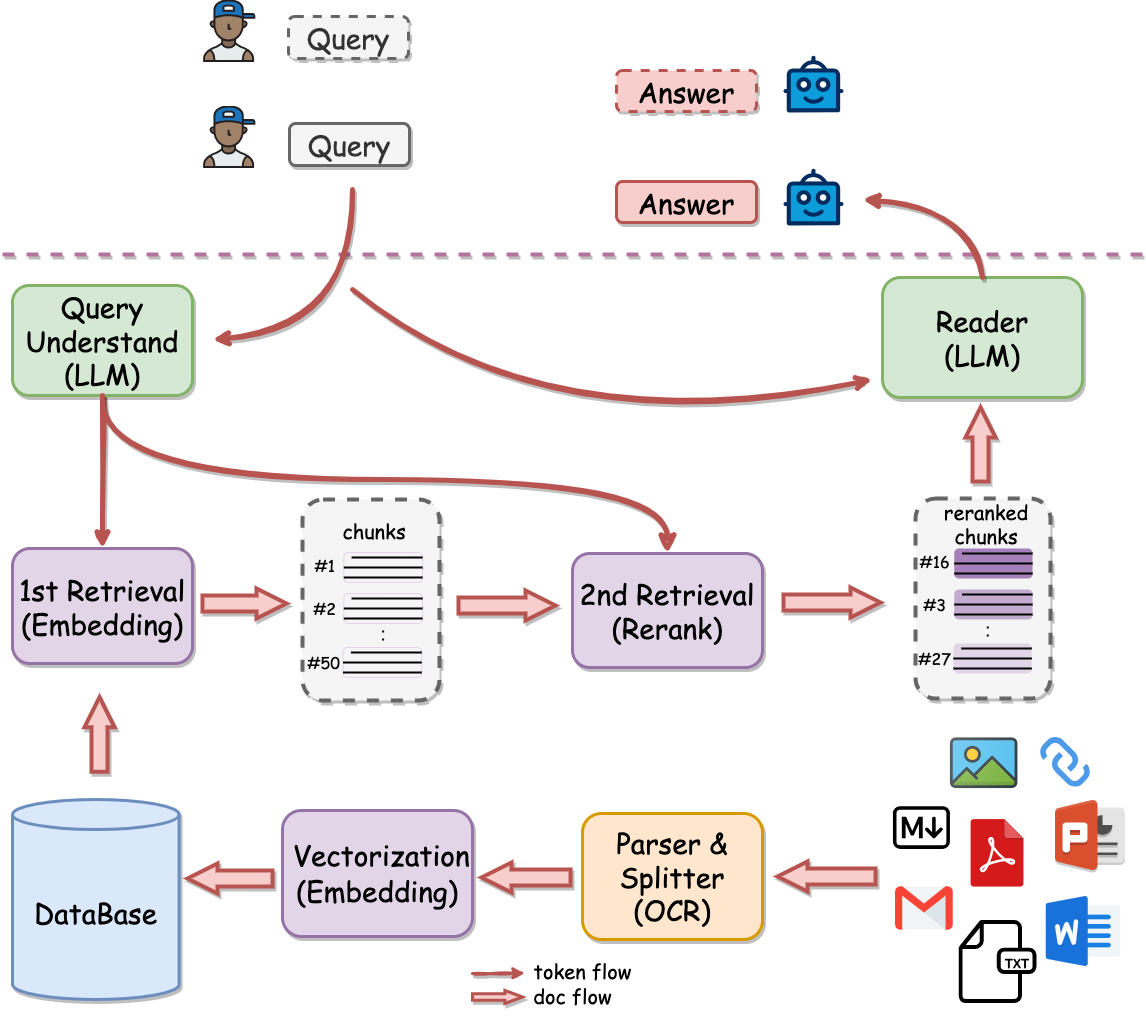
在 QAnything 系统中，数据接入层处于整个架构的最前端，承担着对多源异构数据进行采集、识别、提取和规范化的关键任务。由于面向知识问答系统的数据来源极为丰富，包括文本文档、图片、网页、语音、表格甚至混合多模态内容，如何高效、准确地处理这些非结构化数据，是系统能否顺利构建知识库的第一道门槛。

QAnything 的数据接入层秉承“多模态、模块化、结构化”的设计理念，结合当前主流的开源解析工具与深度学习模型，提供了丰富的文档解析能力和标准化数据封装流程，为下游的知识切分、向量化建模与问答生成提供了坚实的数据基础。

QAnything 针对结构化程度较高的传统文档类型，构建了稳定、准确的提取机制，为支持中文语境下的处理，QAnything 进行了编码自动识别（UTF-8/GBK），并引入繁简转换、符号标准化等预处理操作。当用户上传网页或提交 URL 时，系统通过 requests + BeautifulSoup 获取页面内容，并借助 readability-lxml 框架自动识别正文区域，去除网页广告、导航栏、侧边栏等干扰信息。网页元信息如标题、发布时间、作者与站点来源也会一并提取，形成完整的知识内容封装。针对扫描版 PDF、文档截图等图片格式的数据，系统集成了 PaddleOCR 模型，支持中文、英文和中英混排文档的识别，并可提取表格内容与竖排文字。图像识别过程中，支持批量图像合并处理（如多页扫描件）与图像预处理（如去噪、二值化），提升了识别质量和鲁棒性。

数据接入层承担了系统“理解原始世界”的职责。其设计理念体现了广谱适配（多格式支持）、深度抽取（结构还原）与强健架构（异步处理、错误容忍）的高度统一。该层的表现直接影响到知识切分的粒度、向量检索的准确度以及问答内容的上下文丰富性，是构建高质量问答系统不可或缺的基础组件。

2.1.2 知识加工层



图表 1

从QAnything官方给出的架构图可以看出用户

2.1.3 向量化与存储层

* 文档解析模块：
  + 文件类型适配器（PDF/Word/Excel等）
  + OCR引擎集成（如PaddleOCR）
* 向量化引擎：
  + 嵌入模型（如bge-small-chinese）
  + 量化压缩组件
* 检索系统：
  + 混合索引（倒排+向量）
  + 相似度计算（余弦/欧氏距离）

2.1.4 问答引擎层

2.1.5 应用交互层

2.2 技术选型依据

2.2.1 前端：Vue3 + TypeScript 的优势

Vue3 的 Composition API，对比 Vue2 的 Options API，代码逻辑聚合度提升 40%，便于复杂交互场景（如多文件上传、实时进度反馈）的状态管理。

Proxy 驱动的响应式系统，支持 Map/Set 等复杂数据结构的自动追踪，解决 Vue2 对数组监听受限的问题（如文档列表的动态渲染性能提升 30%）。

TypeScript 静态类型校验，开发阶段捕获 65% 的类型错误（基于微软 TypeScript 团队统计数据），避免文档解析参数传递错误等运行时问题。

与后端 API 的类型对齐，通过 OpenAPI 生成 TypeScript 类型定义，确保前后端接口一致性。

Vite 构建工具，冷启动速度 <500ms（Webpack 平均 3-5s），HMR 更新 <50ms，显著提升开发效率。

按需加载与 Tree-shaking，最终构建体积减少 35%（对比 Vue2 + Webpack），关键路径加载时间 <1s。

2.2.2 后端：NestJS的模块化特性

分层架构设计

依赖注入 (DI)：解耦业务逻辑与基础设施（如更换向量数据库仅需修改对应模块）。

领域驱动设计 (DDD)：将文档解析、向量化、检索等核心功能划分为独立领域模块。

2.2.3 数据库：MySQL关系模型设计

服务器购买

MySQL 容器化部署

docker run -d \

--name mysql-container \

-e MYSQL\_ROOT\_PASSWORD=root \

-e MYSQL\_DATABASE=mydb \

-e MYSQL\_USER=user \

-e MYSQL\_PASSWORD=password \

-p 3306:3306 \

-v mysql\_data:/var/lib/mysql \

mysql:8.0

 -d：后台运行容器

 --name mysql-container：容器名称

 -e MYSQL\_ROOT\_PASSWORD=root：设置 root 用户密码

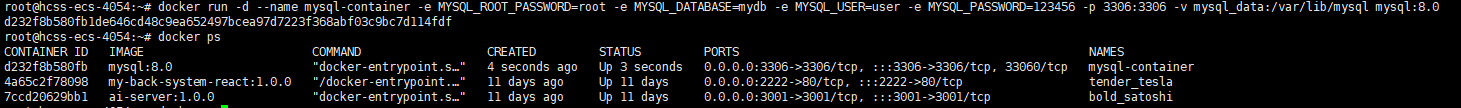
 -e MYSQL\_DATABASE=mydb：创建名为 mydb 的数据库

 -e MYSQL\_USER=user：创建用户 user

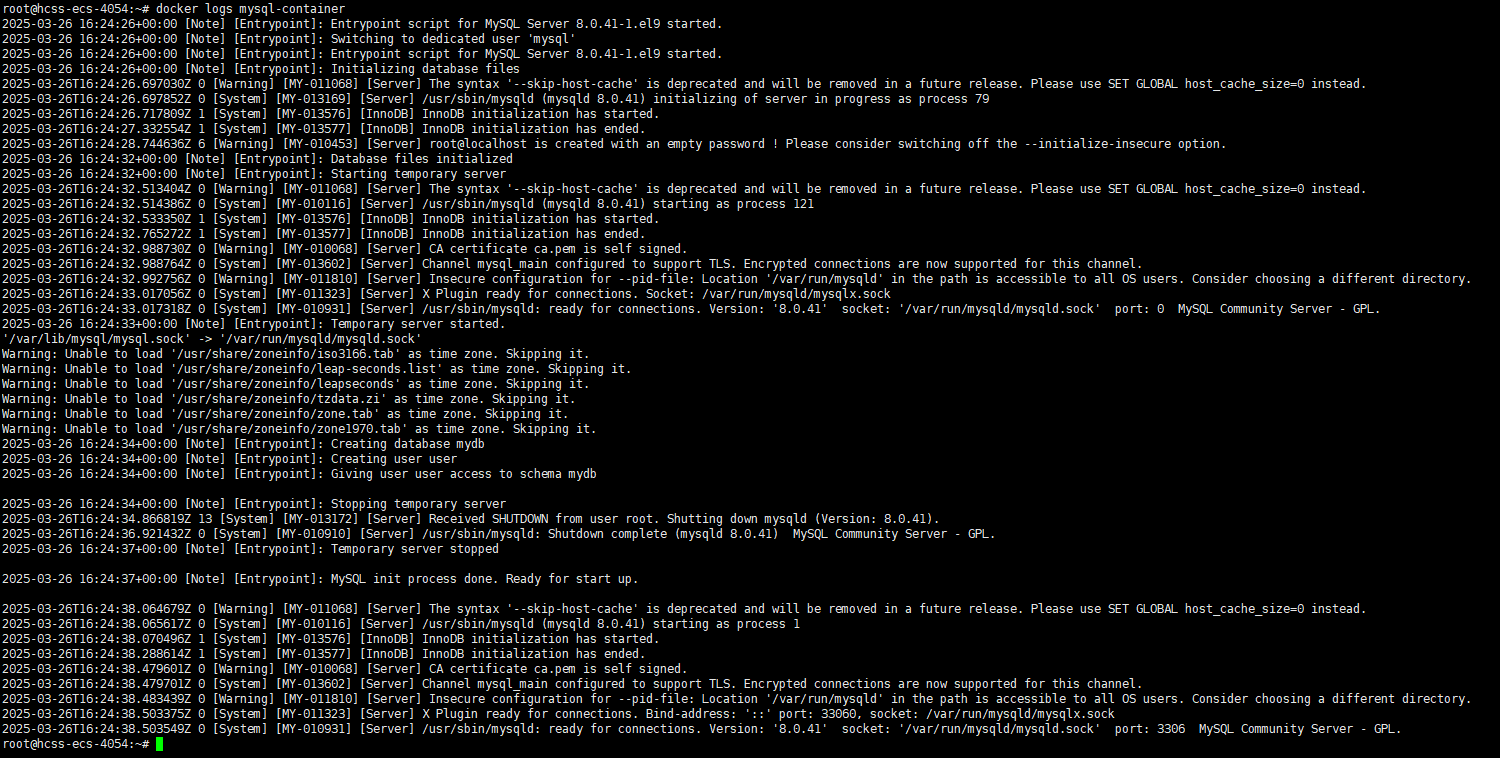
 -e MYSQL\_PASSWORD=password：用户 user 的密码

 -p 3306:3306：将 MySQL 端口映射到宿主机

 -v mysql\_data:/var/lib/mysql：数据持久化



查看日志



本机远程连接数据库

mysql -h 120.46.199.75 -P 3306 -u root -p

创建数据表

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT COMMENT '用户ID',

username VARCHAR(50) NOT NULL UNIQUE COMMENT '用户名',

password VARCHAR(255) NOT NULL COMMENT '密码（加密存储）',

role ENUM('admin', 'user') NOT NULL DEFAULT 'user' COMMENT '用户角色（admin/user）',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户表';

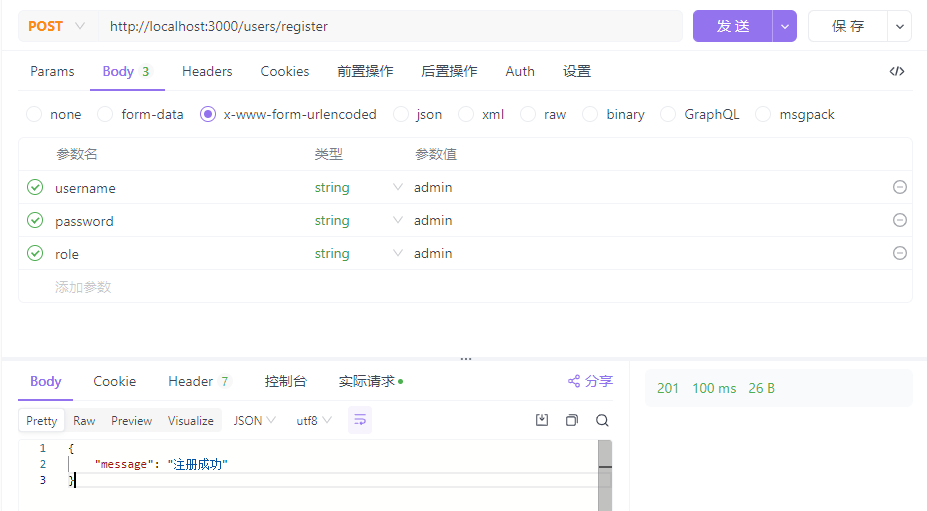
插入数据

INSERT INTO users (username, password, role) VALUES

('admin\_user', 'hashed\_password\_1', 'admin'),

('normal\_user', 'hashed\_password\_2', 'user');

测试注册接口



文件预览：

前端请求接口获取base64

@Post('getFileBase64')

**getFileBase64**(@Body() *getFileBase64*: GetFileBase64Dto) {

    return this.filesService.getFileBase64(*getFileBase64*);

  }

**getFileBase64**(*getFileBase64*: GetFileBase64Dto) {

    const { fileName } = *getFileBase64*;

    const filePath = `./uploads/${fileName}`;

    if (!existsSync(filePath)) {

      throw new BadRequestException({

        code: '404',

        message: '文件不存在',

      });

    }

    const fileExt = fileName.split('.')[1].toLowerCase();

*// 根据文件后缀判断 MIME 类型*

    const mimeType = getMimeType(fileExt);

*// 读取文件并转换为 base64*

    const fileData = readFileSync(filePath);

    const base64Data = fileData.toString('base64');

    return {

      code: 200,

      message: '获取成功',

      data: {

        fileName,

        ext: fileExt,

        base64: `data:${mimeType};base64,${base64Data}`,

      },

    };

  }

原方案，使用get请求服务端，然后将html返回给前端展示

@Get('preview/:fileName')

**previewFile**(@Param('fileName') *fileName*: string, @Res() *res*: Response) {

    return this.filesService.previewFile(*fileName*, *res*);

  }

**previewFile**(*fileName*: string, *res*: Response) {

    try {

*// const filePath = `${process.env.BASE\_URL}:${process.env.PORT}/uploads/${fileName}`;*

*// console.log(filePath);*

*// res.redirect(filePath);*

      const fileUrl = `${process.env.BASE\_URL}:${process.env.PORT}/uploads/${*fileName*}`;

      const googleDocsViewerUrl = `https://view.officeapps.live.com/op/view.aspx?src=${encodeURIComponent(fileUrl)}`;

      const htmlContent = `

        <html>

          <body style="margin: 0;">

            <iframe src="${googleDocsViewerUrl}" width="100%" height="100%" style="border: none;"></iframe>

          </body>

        </html>

      `;

*res*.setHeader('Content-Type', 'text/html');

*res*.send(htmlContent);

    } catch (error) {

      if (error instanceof NotFoundException) {

*res*.status(404).send(error.message);

      } else {

*res*.status(500).send('文件预览失败');

      }

    }

  }

现方案，服务端直接静态资源共享，前端直接访问这个src

app.use('/uploads', serveStatic(join(\_\_dirname, '..', 'uploads')));

第3章 系统需求分析

3.1 功能性需求

3.1.1 多用户协作需求

允许用户将创建的Bot共享给指定用户或用户组，支持公开/私有两种共享模式。

角色分级：管理员和普通用户，管理员可以注销任何普通用户、查看详情、快速开始、知识库问答、bot问答，普通用户不能注销用户和查看详情

不同用户的对话历史、上下文数据完全隔离，避免信息泄露。为每个用户分配独立会话ID，结合用户ID进行数据存储

3.1.2 知识库管理需求

构建可扩展的知识管理体系，支持多模态数据的高效处理与动态更新。

支持pdf、docx、pptx、xlsx、md、txt、csv、eml、jsonl、图片格式的文件上传

文件解析

3.1.3 问答性能需求

简单问题的响应时间≤1.5秒，

3.2 非功能性需求

3.2.1 安全性（/数据加密）

JWT鉴权

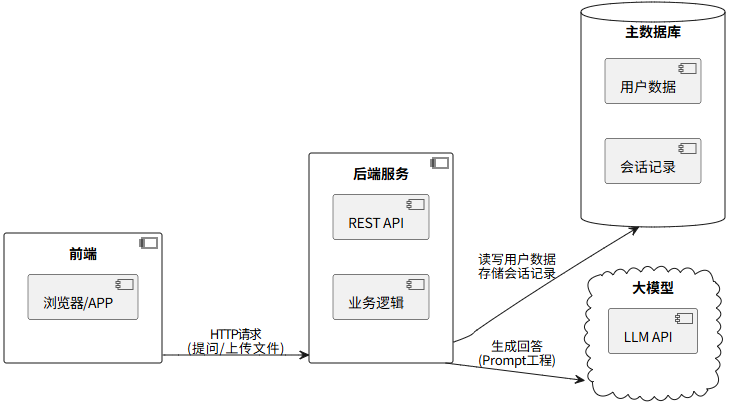
3.2.2 可扩展性（模块化设计）

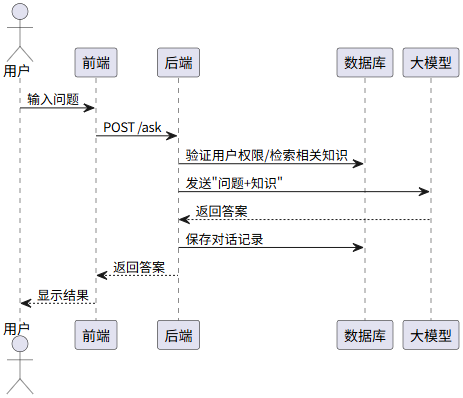
3.2.3 离线部署兼容性

第4章 系统设计

4.1 总体架构设计

4.1.1 系统架构图（前端/后端/数据库/LLM交互）



4.1.2 数据流设计

4.2 核心模块设计

4.2.1 用户与权限模块（RBAC模型设计）

4.2.2 Bot管理模块（知识库配置/版本控制）

4.2.3 问答引擎模块（检索增强生成流程优化）

4.2.3 会话管理模块（MySQL表关系设计）

4.3 数据库设计

4.3.1 ER图与表结构

### users

| **字段名** | **类型** | **是否主键** | **是否唯一** | **默认值** | **说明** |
| --- | --- | --- | --- | --- | --- |
| id | int | ✅ | ❌ | 自动生成 | 主键，自增 |
| username | varchar | ❌ | ✅ | 无 | 用户名 |
| password | varchar | ❌ | ❌ | 无 | 用户密码 |
| role | varchar | ❌ | ❌ | 'user' | 用户角色 |
| createTime | varchar | ❌ | ❌ | 无 | 创建时间 |
| updateTime | varchar | ❌ | ❌ | 无 | 更新时间 |

### knowledges

| **字段名** | **类型** | **是否主键** | **是否唯一** | **默认值** | **说明** |
| --- | --- | --- | --- | --- | --- |
| kbId | varchar | ✅ | ✅ | 无 | 知识库主键 |
| kbName | varchar | ❌ | ❌ | 无 | 知识库名称 |
| type | varchar | ❌ | ❌ | 无 | 类型："quick" / "normal" |
| createTime | varchar | ❌ | ❌ | 无 | 创建时间 |
| username | varchar | ❌ | ❌ | 无 | 关联用户（外键） |

## files

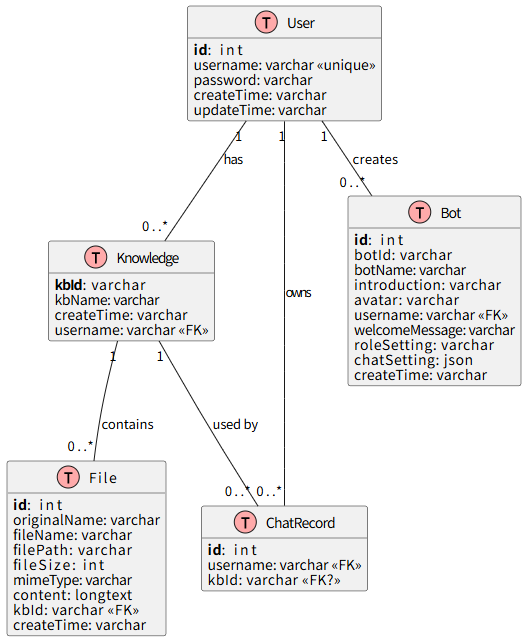
| **字段名** | **类型** | **是否主键** | **是否唯一** | **说明** |
| --- | --- | --- | --- | --- |
| id | int | ✅ | ❌ | 文件主键（自增） |
| originalName | varchar | ❌ | ❌ | 上传时的原始名称 |
| fileName | varchar | ❌ | ❌ | 实际保存的文件名 |
| filePath | varchar | ❌ | ❌ | 文件存储路径 |
| fileSize | int | ❌ | ❌ | 文件大小（字节） |
| mimeType | varchar | ❌ | ❌ | MIME 类型 |
| content | longtext | ❌ | ❌ | 文件内容 |
| kbId | varchar | ❌ | ❌ | 外键，关联知识库 |
| createTime | varchar | ❌ | ❌ | 上传时间 |

### chat\_record

| **字段名** | **类型** | **是否主键** | **说明** |
| --- | --- | --- | --- |
| id | int | ✅ | 自增主键 |
| username | varchar | ❌ | 外键，关联 User.username |
| kbId | varchar | ❌ | 外键，关联 Knowledge.kbId，可为空 |
| type | varchar | ❌ | 枚举：'quick' 或 'home' |
| messages | text | ❌ | JSON 字符串（序列化的数组） |

### ****bots****

| **字段名** | **类型** | **说明** |
| --- | --- | --- |
| id | int | 主键，自增 |
| botId | varchar | 机器人唯一标识 |
| botName | varchar | 名称 |
| introduction | varchar | 简介 |
| avatar | varchar | 头像 |
| username | varchar | 外键，关联 users.username |
| kbIds | simple-array | 知识库ID数组（字符串数组） |
| welcomeMessage | varchar | 欢迎语 |
| roleSetting | varchar | 角色设置 |
| chatSetting | json | 聊天配置（结构体） |
| createTime | varchar | 创建时间 |



第5章 系统实现

5.1 前端实现

5.1.1 多Bot切换界面（Vue3 + Pinia状态管理）

5.1.2 实时消息展示（WebSocket/SSE）

5.2 后端实现

5.2.1 NestJS分层架构（Controller-Service-Repository）

5.2.2 文件解析服务（PDF/Word/OCR集成）

第6章 系统测试与分析

6.1 功能测试

6.1.1 Bot共享测试（多用户同时访问）

6.1.2 权限控制测试（管理员vs普通用户）

6.1.3 文件解析覆盖率测试

第7章 总结与展望

7.1 研究成果总结

7.1.1 实现的功能清单

7.1.2 创新点（多用户协作/本地化优化）

7.2 不足与改进方向

7.2.1 当前局限性（如大文件处理效率）

7.2.2 未来计划（多Bot联动/边缘计算支持）

致 谢

在本项目的完成过程中，我得到了许多老师、同事、朋友和家人的大力支持与无私帮助。在此，我谨向所有在项目开发与研究过程中给予我指导、帮助和鼓励的人表示最诚挚的感谢。

首先，我要特别感谢我的导师王红玲，她不仅在专业知识上给予了我极大的帮助，更在项目设计思路、系统结构架构以及技术细节方面给予了我悉心的指导。在项目遇到瓶颈和困难时，她总能耐心地提供思路，帮助我重新理清方向。

其次，我要感谢我的同事刘俊雄雄哥以及组内的同事们。最初，我在网易实习的过程中，接触到了QAnything这个项目，通过数日的学习和开发，我了解了其中使用的新型技术，于是我萌生了实现一个自己的问答系统，这样一个构想。于是便有了这个毕设以及这篇论文。在整个开发过程中，熊哥为我解答了本不属于我的工作内容，我不断地跟学习，跟大家相互协作、共同进步。他们严谨的态度、细致的代码风格以及对问题的深刻见解都给我留下了深刻印象，是他们的陪伴和支持使得本项目能够顺利推进并取得阶段性的成果。

同时，我也感谢项目过程中参考和学习过的优秀开源项目、文档及社区资源，是这些宝贵的资料为我的开发提供了理论与实践基础。特别是Vue、NestJS 等技术栈的使用，使项目的后端架构更加清晰与高效。

最后，我要感谢我的家人对我的理解与支持，是他们给予我充足的时间和良好的环境，让我能够安心地投入到项目之中。正是因为有你们的默默支持，我才能坚持走到最后。

再次衷心感谢所有帮助、鼓励和支持我的人！

参考文献

1. 梅忆寒,王琳琳,王鹏飞,等.基于多模态与检索增强生成的数据库知识问答系统[J].计算机教育,202 4,(12).
2. 杜恒峰.DeepSeek-R1惊艳全球展示中国AI发展巨大潜力[N].每日经济新闻,2025-01-27(001).
3. 穆肃,陈孝然,周德青.生成式人工智能赋能教学设计分析：需求、方法和发展[J].开放教育研究,2025,31(01).
4. 刘永东, 王文涛, 胡鹏. 本地化知识库问答系统研究与实现[J]. 软件学报, 2022, 33(6): 1428-1439.
5. [姜嘉伟.基于Langchain-LLMs框架的智能问答系统的设计与实现[D].延边大学,2024.
6. 任海玉,刘建平,王健,等.基于大语言模型的智能问答系统研究综述[J/OL].计算机工程与应用,1-24[2025-02-06].
7. Radeva I ,Popchev I ,Doukovska L , et al.Web Application for Retrieval-Augmented Generation: Implementation and Testing[J].Electronics,2024,13(7).
8. Mansurova A ,Mansurova A ,Nugumanova A .QA-RAG: Exploring LLM Reliance on External Knowledge[J].Big Data and Cognitive Computing,2024,8(9):115-115.