

**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，得益于自然语言处理和深度学习技术的不断发展，知识库问答系统逐渐成为智能信息检索领域中的一个重要研究方向。不过在实际应用过程中，这类系统仍存在两个突出问题，其一是对云端服务的依赖性较强，因而在涉及数据隐私或安全性的场景下难以满足需求；其二则是文档格式支持不够灵活，难以适应多样化的知识载体类型，这些问题在一定程度上限制了知识管理系统在真实业务环境中的推广和落地。

围绕上述挑战，本文基于QAnything的系统架构理念，提出并开发了一个支持本地部署的智能问答系统。该系统设计强调数据隐私与格式兼容性的双重保障，既能够在离线环境下处理敏感数据问答任务，同时也具备处理多种文档格式的能力。系统整体采用Vue.js搭建用户交互界面，后端则由NestJS提供服务支撑，并结合MySQL数据库完成知识的存储与管理；在问答引擎部分，引入大语言模型以提升自然语言理解与生成的能力。

该系统的实际价值主要体现在以下几个方面：一方面，它为各类组织在构建私有知识库过程中提供了更加稳妥的数据安全解决方案；另一方面，借助智能化的知识抽取与检索机制，能够有效提升组织内部的信息流转效率；此外，系统中集成的 ChatBot 模块支持多用户共享模型配置，并实现了数据资源的实时协同使用，进一步拓展了其在协同办公与知识协作场景中的应用潜力。

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

**ABSTRACT**

In recent years, thanks to the continuous development of natural language processing and deep learning technologies, knowledge base question-answering systems have gradually become an important research direction in the field of intelligent information retrieval. However, in practical applications, these systems still have two major problems. The first is their strong dependence on cloud services, making them difficult to meet requirements in scenarios involving data privacy or security; the second is the insufficient flexibility in supporting document formats, making it difficult to adapt to diverse types of knowledge carriers. These problems to some extent limit the promotion and implementation of knowledge management systems in real business environments.

In response to these challenges, this paper, based on the system architecture concept of QAnything, proposes and develops an intelligent question-answering system that supports local deployment. The system design emphasizes dual guarantees of data privacy and format compatibility, enabling it to handle sensitive data question-and-answer tasks in offline environments and also having the ability to process various document formats. The system as a whole is built using Vue.js for the user interface, with NestJS providing service support at the backend, and MySQL database is used for knowledge storage and management. In the question-answering engine part, a large language model is introduced to enhance the ability of natural language understanding and generation.

The practical value of this system mainly lies in the following aspects: Firstly, it provides a more reliable data security solution for various organizations when building private knowledge bases; secondly, with the intelligent knowledge extraction and retrieval mechanism, it can effectively improve the information flow efficiency within the organization; moreover, the ChatBot module integrated in the system supports shared model configuration among multiple users and realizes real-time collaborative use of data resources, further expanding its application potential in collaborative office and knowledge collaboration scenarios.

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

目 录

[摘 要 Ⅰ](#_Toc198508407)

[**ABSTRACT** Ⅱ](#_Toc198508408)

[第1章 绪论 1](#_Toc198508409)

[1.1 研究背景 1](#_Toc198508410)

[1.2 研究意义 1](#_Toc198508411)

[第2章 技术综述 3](#_Toc198508412)

[2.1 QAnything架构分析 3](#_Toc198508413)

[2.1.1 数据接入层 3](#_Toc198508414)

[2.1.2 知识加工层 4](#_Toc198508415)

[2.1.3 向量化与存储层 4](#_Toc198508416)

[2.1.4 问答引擎层 5](#_Toc198508417)

[2.1.5 应用交互层 5](#_Toc198508418)

[2.2 技术选型依据 5](#_Toc198508419)

[2.2.1 Vue3 + TypeScript 的优势 6](#_Toc198508420)

[2.2.2 NestJS的分层架构 6](#_Toc198508421)

[2.2.3 MySQL关系模型设计 7](#_Toc198508422)

[第3章 系统需求分析 9](#_Toc198508423)

[3.1 功能性需求 9](#_Toc198508424)

[3.1.1 快速开始问答 9](#_Toc198508425)

[3.1.2 知识库问答 9](#_Toc198508426)

[3.1.3 聊天Bot 10](#_Toc198508427)

[第4章 系统设计 11](#_Toc198508428)

[4.1 总体架构设计 11](#_Toc198508429)

[4.1.1 系统架构分析 11](#_Toc198508430)

[4.1.2 数据流设计 13](#_Toc198508431)

[4.2 数据库设计 13](#_Toc198508432)

[4.2.1数据库实体关系设计 14](#_Toc198508433)

[4.2.1 数据库逻辑结构设计 15](#_Toc198508434)

[第5章 系统实现 18](#_Toc198508435)

[5.1 前端实现 18](#_Toc198508436)

[5.1.1 登录注册 19](#_Toc198508437)

[5.1.2 快速开始 19](#_Toc198508437)

[5.1.3 知识库问答 24](#_Toc198508438)

[5.1.4 Bots 25](#_Toc198508439)

[5.2 后端实现 27](#_Toc198508440)

[5.2.1 分层架构的实现 27](#_Toc198508441)

[5.2.3 文件解析服务 30](#_Toc198508442)

[5.3 部署 30](#_Toc198508443)

[5.3.1 前端挂载 31](#_Toc198508444)

[5.3.2 服务端部署 32](#_Toc198508445)

[5.3.3 数据库部署 32](#_Toc198508446)

[结束语 35](#_Toc198508451)

[致 谢 35](#_Toc198508451)

[参考文献 36](#_Toc198508452)

第1章 绪论

## 1.1 研究背景

在当下社会全面推进数字化转型的背景下，企事业单位、教育体系以及政府机关等各类组织，正不断积累并沉淀数量庞大、类型多样的知识资产，涵盖内容包括技术标准、业务流程文档、教学资料乃至行政管理文件等，但在实际使用过程中，这些资源的管理方式与服务能力仍面临较为突出的瓶颈。传统依赖关键词检索的搜索技术由于受限于语义理解能力，常常难以精准响应用户需求，既存在召回信息不足的问题，也在结果匹配度上表现不佳，难以满足人们对高效、精准知识服务的期待。相比之下，融入认知智能的问答系统凭借更加贴近自然语言的对话式交互，在提升知识获取效率方面展现出显著优势，并有助于改善用户在知识使用过程中的整体体验。

人工智能领域的突破性进展使得知识驱动型问答系统[1]（KBQA，Knowledge-Based Question Answering）在信息科学界获得广泛关注。早期问答技术多采用结构化查询或预设规则，当应用于组织机构内部的非标准化文档，如PDF、Word、Excel等文档资源时表现欠佳。最新研究表明，检索增强生成[1]（Retrieval-Augmented Generation，RAG）为知识库问答提供了新思路：RAG的核心思想是先通过检索从知识库中找出相关文档片段；再通过生成由大语言模型生成自然语言回答。优势在于无需对预训练语言模型进行参数微调，可直接利用现有知识库作为外部知识源；通过检索增强生成机制，模型生成的回答能够紧密锚定知识库中的事实依据，显著提升回答的事实准确性，有效减少大语言模型常见的幻觉现象。

以网易有道开源的本地知识问答系统QAnything为例，它作为当前实践中的代表性项目，旨在为用户构建一个既安全可控，又兼容多种数据格式，同时具备高效检索能力的私有化知识管理解决方案。该系统在设计上充分考虑了实际应用中的多元需求，其一大亮点在于对多模态文件的解析能力，支持包括PDF、Word、Excel、PPT等常见文档形式以及图像类数据源在内的多种输入类型；同时，系统内置的知识处理流程也展现出较高的智能化水平，借助语义感知的文本分割算法及基于BGE模型的向量编码技术，能够高效构建本地知识索引，进一步提升信息组织效率；此外，通过引入语义检索与大语言模型相结合的检索增强生成架构，系统在回答准确性与事实性方面也得到了显著增强。凭借上述能力，QAnything尤其适用于对数据安全与隐私有高要求的场景，如金融、医疗等领域，能够为组织层面的知识管理提供即插即用的技术支撑。

## 1.2 研究意义

本研究借鉴了QAnything的架构理念，设计并开发了一套智能问答系统，旨在推动知识管理领域的理论研究与实际应用。技术实现方面，系统前端采用Vue.js构建，确保界面响应灵活流畅，后端基于NestJS框架开发，提供高效稳定的服务支持，数据层则利用MySQL实现结构化的知识存储与管理。更为关键的是，系统创新性地融合了大语言模型，使自然语言问答功能得以实现，构建出一套完整且实用的技术方案。尤其值得一提的是，系统具备对PDF、Word、Excel、PPT等多格式文档的智能解析，这不仅突破了传统知识管理系统在文件兼容性上的限制，也提升了知识利用的广度。

从行业发展的视角来看，本研究成果有望推动知识管理系统向智能化和易用化方向的转型升级。系统将人工智能技术与传统的知识管理需求进行融合，通过自然语言交互界面降低了使用门槛，借助大语言模型强大的理解能力，系统功能得到了大幅拓展。这样的智能化升级不仅体现在技术层面，还重新定义了人机之间的知识交互模式，使知识获取从以往的被动检索转向主动对话，极大提升了知识传递的效率和用户体验。随着数字化转型的不断深入，所提出的技术方案有望在政府机构、教育科研、企业研发等领域发挥作用，为构建新一代智能知识基础设施提供了参考。

第2章 技术综述

本章主要分析了QAnything的系统架构设计及问答系统的技术选型依据。QAnything采用分层模块化的架构，各功能层通过标准化接口互相通信，确保系统具备高内聚、低耦合的特点。在技术选型过程中，研究遵循了基本评估流程，重点考虑了技术栈在开源社区的活跃程度、开发维护的便捷性以及学习曲线的陡缓。

## 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，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。QAnything系统架构图如图2-1所示。

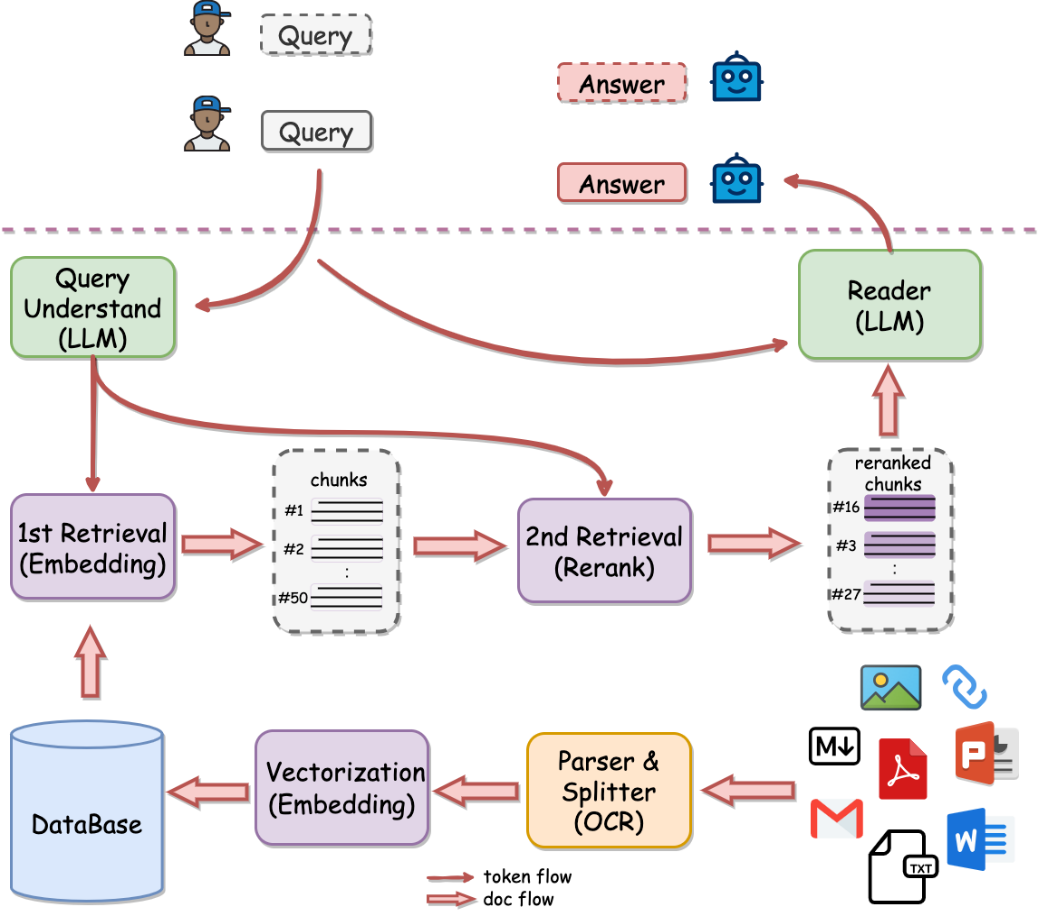


图2-1 QAnything系统架构图

### 2.1.1 数据接入层

数据接入层位于QAnything架构的最前端位置，肩负着对数据展开采集、识别、提取以及规范化等关键任务，面向知识问答系统的数据来源相当丰富，其中涉及了文本文档、图片、网页、表格甚至混合多模态内容，在此情况下，怎样可高效且准确地处理这些并非结构化的数据，这构成了构建知识库的首要任务。

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）。当用户上传网页或提交URL时，系统通过爬取页面内容，将网页元信息（如标题、发布时间、作者）进行组合成完整的知识内容。针对PDF、文档截图等图片格式的数据，系统集成了OCR模型，支持中文、英文和中英混排文档的识别。

其设计观念呈现出多格式支持跟结构还原二者相统一的特点，这一层的呈现状况会直接对知识切分的细致程度、向量检索的精确程度以及问答内容上下文中所有的丰富程度产生影响，属于构建高质量问答系统里必不可少的基础性组成部分

### 2.1.2 知识加工层

知识加工层承担着从“原始文本”向“可用知识”转化的关键角色。它的核心目标是对接入的数据进行语义理解、结构切分、内容清洗和增强，为构建高质量的语义索引与精准问答提供语义粒度更佳、上下文组织更合理的知识单元。

原始文本大多时候呈现为连贯的长文档或者复杂的结构形式，需要将其进行合理切分，形成适合语义索引的小段落，QAnything的知识切分策略对字符长度有所考量，同时也综合顾及语义完整性和段落结构方面的因素，所谓最基本的切分方法是固定长度切分，像是每500字左右设置为一个段落。此方法有简单高效的特点，适用于内容结构较为松散、篇幅相对平均的场景，如新闻文章、百科文本这类。

### 2.1.3 向量化与存储层

在知识加工层完成结构化文本的提取与切分操作之后，系统随后便进入向量化与索引阶段，此阶段的关键目标乃是把每一个知识块都转换成为在语义空间里可用于相似度检索的高维向量表示，并且要依靠打造高效的向量索引，以对后续的大语言模型基于语义的上下文感知问答能力形成支撑。

首先，系统要挑选恰当的向量化模型来对文本开展语义编码工作，QAnything一般会采用适配中文语义检索任务的bge系列模型，这个模型是在大规模中文问答与文档匹配数据里进行训练的，它可去理解句子之间的语义关系，为了提升向量的检索性能，系统在编码文本之前会引入提示词模板，像“为这个句子生成表示以用于检索：”这类方式，以此对文本实施prompt提高，促使模型生成更具可区分性的语义向量。

考虑到文档结构信息对语义表示有意义，系统特意设计了一系列语义提高策略，一般的做法是把当前段落的标题以及上级章节标题拼接至段落前部，以此引入更为丰富的上下文线索，向量生成完毕后，紧接着的步骤是构建高效的向量索引结构，QAnything预设集成了FAISS（Facebook AI Similarity Search）向量检索库。每一个知识库都有对应的独立向量索引文件，系统可支持定时构建、增量更新以及实时持久化，保障用户在添加或者删除文档后，依旧可维持向量检索结果的时效性与一致性。

### 2.1.4 问答引擎层

问答引擎层在QAnything框架里担当着极为关键的角色，是达成用户自然语言问题与知识内容之间语义交互的核心构成部分，这一层的设计主要依据合理的Prompt工程、模型集成以及上下文关联这三个关键要点来推进。

问答引擎层的工作是构建高质量提示词模板即Prompt，用来引导语言模型于给定知识语境中生成符合预期的问题答案，QAnything针对不同任务类型像摘要、解释等设计了多样Prompt模板，包含“基于以下内容回答问题”“只回答与提供内容相关的信息”等指令性语句。这些模板为模型生成设定了行为边界，又能提升模型对原始知识块的聚焦能力，降低了幻觉现象的发生。

在模型选择这一方面，QAnything可支持多种不同类型的大语言模型，其中囊括了本地部署的模型以及API调用类模型，本地模型针对那些对隐私安全有着要求的场景比较适用，而API模型它有着强大的算力以及较快的响应速度，有一定优势，系统在架构上预留出了统一的模型调用接口，这样就可以让用户可自由地去切换或者组合模型类型了。

为提升多轮问答体验，QAnything的问答引擎还具备上下文记忆能力。通过记录用户上一轮的提问与系统回答，并将其简要摘要作为本轮输入的补充提示，系统实现了“带记忆的对话模式”。

### 2.1.5 应用交互层

应用交互层作为QAnything直接与用户交互的层面，职责是把底层知识问答能力运用图形化形式呈现给用户，并且支撑灵活的参数调控能力，这一层面的设计和用户体验密切相关，也会对知识问答系统的可用性以及可推广性产生直接影响。

前端利用响应式Web界面，经由RESTful API和后端通信，有自然语言提问、文件上传以及对话历史管理等主要功能，交互层中知识库的灵活配置界面，能让用户可视化创建和管理不同知识库，借助SSE技术达成对话内容实时流式返回，提升了用户体验。面对跨语言与多区域使用场景，应用交互层还支持界面中英文切换以推动国际化推广。

以上就是QAnything的系统分层架构，这几个层次相互协同，达成了从数据到知识，再到智能问答的全流程处理，为本课题提供了重要参考。

## 2.2 技术选型依据

为了实现系统在企业、学校内部场景下的高可用性、可维护性与良好用户体验，本系统从前后端分离架构的角度出发、基于本人在实习过程中的文档学习、使用体验，选用了Vue 3+TypeScript作为前端开发框架，UI组件库使用前端对Vue较为友好的Ant Design Vue框架，NestJS作为后端服务框架，以及MySQL作为系统核心数据存储方案。

### 2.2.1 Vue3+TypeScript的优势

在前端技术方案进行选择时，借助全面且细致的技术调研以及评估工作，最终决定采用将Vue 3和TypeScript结合起来的技术栈，Vue 3属于当下主流的前端框架，在性能体现、响应式系统以及组件化开发等方面相比于Vue 2有了较大程度的进步，它所拥有的组合式即Composition API设计可让复杂业务逻辑的组织变得更为清晰并且有可复用性，这为项目里涉及到的多功能交互场景给予了比较理想的技术支撑，囊括文件上传处理、智能问答交互、路由跳转等多个复杂功能模块的开发需求。

引入TypeScript给项目开发给予了类型方面的保障，其静态类型检查机制可有效避开常见的与类型有关的运行时错误，借助智能代码提示让开发效率得到了提升，于涉及多模块协作以及跨组件数据交互的复杂前端架构当中，TypeScript的强类型特性呈现出独特优势。随着Vue官方对TypeScript支持力度持续加大，再配合Vite等现代化构建工具的运用，整个项目的开发调试过程变得更高效且可靠。

Ant Design Vue（简称antdv）是阿里开源的专为Vue开发的UI组件库，由Vue 3和TypeScript构建而成，为企业级应用开发提供高质量的组件实现。提供统一的视觉风格和交互体验，包含60+高质量组件，覆盖表单、数据展示、导航、反馈等常见场景。全面支持Vue 3 Composition API，与TypeScript深度集成，提供完整的类型定义。采用现代前端工具链，可无缝对接Vite、Webpack等构建工具。

### 2.2.2 NestJS的分层架构

后端采用NestJS作为开发框架，首先NestJS基于Node.js平台，采用TypeScript作为开发语言，其语法特性对前端工程师较为友好；其次NestJS采用了清晰的分层架构设计，主要包括控制器层、业务服务层、数据持久层和功能模块四个核心组成部分，各层职责明确且相互独立。

控制器层处于请求的入口位置，其主要作用是接收并解析HTTP请求，整个系统依据业务领域对控制器进行了划分，其中有负责处理用户认证的UserController、承担管理对话流程任务的ChatController、针对知识库相关操作的KnowledgeController以及专门负责Bot操作的BotController等。这些控制器的职责限定于请求路由以及参数校验，而将具体的业务处理工作给业务层去完成。

业务服务层属于整个业务逻辑的关键所在，它封装了许多处理逻辑，像进行数据库调用、判断用户权限、组合知识库内容、与大模型接口开展交互以及记录对话内容等，在当前这个系统里，以ChatService为例，其中囊括了对大模型的调用流程、知识拼接逻辑以及SSE的数据格式处理，而KnowledgeService所负责的是知识库的新增、编辑以及删除等相关操作。

数据持久层主要负责与数据库的交互，通过调用Object-Relational Mapping（ORM，如TypeORM）[12]来进行实体数据的持久化管理。在NestJS中，可以直接通过@InjectRepository注解引入实体类对应的仓库，支持通过标准化的API完成数据存取操作，如查询检索、数据更新等基础操作，分别对应.find()、.save()、.update()等方法。本项目中各实体类如UserEntity、ChatRecordEntity、KnowledgeEntity等，均有独立的repository管理数据访问逻辑。

模块化设计在是NestJS的重要特性，每个模块封装相关的控制器、服务、实体等组件，形成内聚的功能集合。本项目划分了多个业务模块，如用户、Bot、知识库、聊天等，模块之间通过imports机制进行依赖注入，保证功能的独立性和模块间的解耦。NestJS分层架构图如图2-1所示。

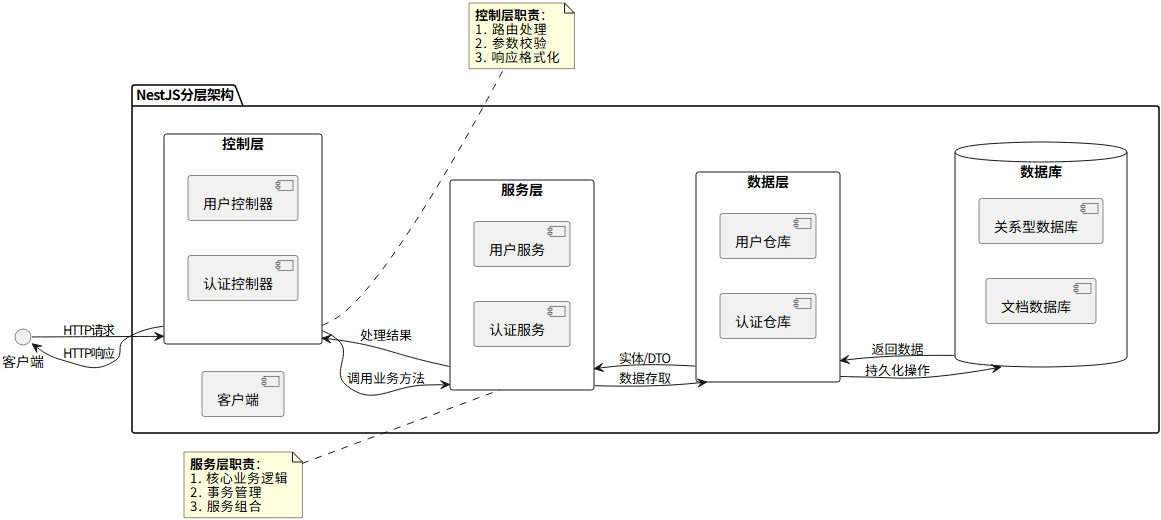


图2-1 NestJS分层架构图

### 2.2.3 MySQL关系模型设计

在数据库的选择上，选取了MySQL作为核心关系型数据库系统，这主要是鉴于其在结构化数据管理领域有的成熟度以及相应的性能呈现，该系统所要处理的数据类别包含用户信息、上传文档的元数据、问答历史记录以及Bot信息等，这些数据有着较为突出的结构性与关系性，适宜运用关系型数据库来实施管理。

在技术整合工作当中，MySQL和NestJS框架里的TypeORM模块达成了深度融合，达成了面向对象形式的数据操作模式，这样的组合维持住数据库操作的灵活性，还提升了开发效率以及代码可维护性，前端把Vue3+TypeScript与后端NestJS+MySQL的技术组合运用起来，构成了很不错的技术协同效应。前后端一致的TypeScript语言环境，让接口定义以及数据结构的类型校验变得更加严谨，切实减少了因接口变更而产生的问题，采用前后端分离架构可实现独立部署，对后期系统迭代以及性能调优有利，MySQL充当稳定的数据支撑，保障了业务核心数据有高一致性与可维护性。

综合来看，此次技术选型充分顾及系统功能实现要求，亦考量到技术生态、开发效率以及系统可扩展性等方面，于实际开发跟部署进程里，此技术栈呈现出不错的稳定性与灵活性，可达成系统在不同应用场景下的快速适配以及高质量交付诉求。

第3章 系统需求分析

## 3.1 功能性需求

为达成用户对非结构化知识的迅速访问以及有效利用，此系统架构体系要支持一整套以“文件—知识—问答—Bot”为中心的流程，系统整体功能能划分成三大核心模块，即快速开始问答、知识库问答与聊天Bot，基于这个基础，还会细分成若干关键子功能，比如创建知识库、文件上传、聊天关联、Bot创建、Bot配置与分享等，以此为用户打造低门槛且高体验的智能问答平台。

### 3.1.1 快速开始问答

首先，“快速开始问答”模块是专为两类典型用户场景设计的轻量化解决方案：一是面向初次接触系统的用户提供零门槛的入门体验；二是满足用户临时性、轻量级的文档问答需求。该功能的核心设计理念是“即传即问”的极简交互模式，用户无需提前配置模型参数，仅需上传一个或多个文档，即可迅速进入问答流程。每个用户可以在快速开始中创建多个对话，每个对话相对独立，一个对话对应关联一个知识库，此模式降低了使用门槛，尤其适用于临时性资料阅读、会议文档解读等场景。系统自动将上传文档临时存入一个知识库中，并绑定一个默认问答界面，实现即问即答的体验。

### 3.1.2 知识库问答

“知识库问答”是系统的核心功能模块，适用于持续性知识管理与长期问答需求。该模块支持用户创建任意数量的领域知识库，每个知识库具备独立的文档存储空间和问答配置体系，在文件兼容性方面，系统采用模块化解析引擎，可处理包括PDF、Word、PPT、Excel等办公文档，以及Markdown、纯文本、CSV结构化数据等共计十余种文件格式，并通过集成OCR[13]技术实现对PNG等图片格式的文字识别。在交互放慢，为了方便问答时的查阅，还支持了对各文件内容的预览。问答时，允许用户通过可视化界面自由组合多个知识库作为检索源，系统基于指定的知识库进行语义匹配与回答生成，确保问答结果既包含精确匹配又涵盖语义相关的内容。

不论快速开始模块还是知识库问答模块，用户都能操作知识库，如新增、重命名以及删除等基础操作，在知识库内部还有文件上传、删除以及预览等功能。

和“快速开始”模块相比最明显的区别在，知识库问答运用多知识库共享单一对话的机制，用户可依据自身需求自行选择参与问答的知识库范围，支持单选模式，也允许进行多选操作，如此一来，较大提升了知识检索的自由度以及可控性；另外，在问答阶段，系统提供了模型参数的配置，用户不再被预设的默认配置所局限，可按照实际需求对本地模型或者三方API的各项参数作出调整。这种有配置机制契合不同场景下的需求，提供了深度定制的可能。

为可持续不断地保障问答所有的质量，系统另外还设置了问答配置的自定义功能，比如调整上下文窗口的大小、设置温度系数以及输出token数等，依靠这些有细粒度特点的配置选项，用户可以构建出更适配特定领域需求的知识组织方案，获取更精准以及更专业的问答体验。

### 3.1.3 聊天Bot

“聊天Bot”功能主要关注问答能力的模块化封装以及模型配置的可复用性设计，借助把知识库能力以Bot的形式进行标准化输出，构建出有拟人化交互特点的AI助手体系，此功能运用多实例化架构设计，可支持用户创建任意数量的Bot实例，并且每个实例都可以灵活地绑定一个或者多个知识库，以此作为语义理解的基础。在个性化配置方面，系统给出了多维度的Bot属性定制功能，每一个Bot都可单独配置头像、名称、角色设定、系统提示词等信息，形成带有特定语气和角色定位的对话体验。

借助这种有高度可配置特性的Bot生成机制，用户可迅速搭建起个人专业化智能助手，其典型应用场景覆盖了诸多方面，像心理咨询场景的心理辅导Bot、技术开发场景的编程助手Bot、情感社交场景的恋爱顾问Bot等等，这些丰富多样的场景应用，充分契合了不同场景之下人们对于智能化交互的需求。

此外，创新的分享机制为每个Bot生成唯一访问URL与专属二维码，用户仅需复制分享链接或者用手机扫描二维码，就能快速跳转到该Bot的专属问答页面，此功能突破了地域限制，不论身处何方的用户，登录系统便可与分享的Bot实时对话交流，这种便捷分享方式降低了知识共享的技术门槛，还依靠标准化交互界面保证不同用户有一致的问答体验，为跨区域跨团队知识协作提供了高效便捷解决办法。

第4章 系统设计

## 4.1 总体架构设计

本系统在架构设计上参考了QAnything所提出的模块化问答系统思想，但结合毕设开发周期短、功能需求相对聚焦的实际情况，进行了针对性的裁剪与优化。系统整体采用分层解耦的设计思想，将架构划分为：数据接入层、数据存储层、知识加工层、问答引擎层与应用交互层四大部分。每一层职责清晰，协同工作，构建起一个简洁高效、易于部署的智能问答平台。系统架构图如图4-1所示。

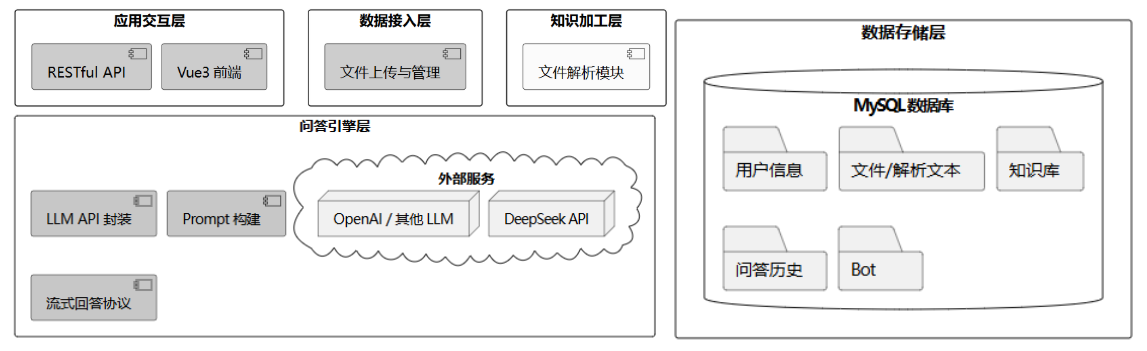
****

图4-1 系统架构图

### 4.1.1 系统架构分析

本系统采用了分层解耦的架构设计思想，将整体功能模块划分为五大核心层次：应用交互层、数据接入层、知识加工层、问答引擎层以及数据存储层，并通过对接外部大模型服务，如DeepSeek[3]和OpenAI API服务，实现了高效、稳定的问答能力。该架构不仅保障了系统的清晰性与可维护性，也为后续功能扩展和模块替换提供了良好的支撑。

#### （1）应用交互层

应用交互层是用户与系统交互的直接入口，包含Vue3+TypeScript构建的Web前端以及采用NestJS框架开发的后端API接口服务。前端提供了用户登录、Bot创建、文件上传、知识库管理、问答对话等一系列功能页面，通过统一的UI设计提升了使用体验。后端则承担了身份鉴权、请求路由、数据分发等职责，是系统运行的中枢神经。

用户的所有行为操作都通过RESTful接口发送给后端服务，后端根据业务逻辑进一步调用数据处理模块或问答引擎模块，并将最终的处理结果返回给前端展现。前后端解耦的模式提升了开发效率与协作灵活性，便于并行迭代和独立部署。

#### （2）数据接入层

数据接入层主要负责用户上传原始数据文件的接收与初步处理，支持包括txt、md、pdf等常见文档格式。文件上传模块在用户上传文件后，先对其进行格式解析和内容提取，再统一命名和存储到系统目录，并在数据库中登记相应的元信息，如文件名、用户ID、知识库归属等。

这一层把文件上传逻辑同后续知识处理过程分离开来，以此保证了系统对于数据有着清晰的分层以及可追溯性，在模块设计方面预留了多种文件格式的处理接口，方便未来可扩展到图像、音频等多种模态的数据上。

#### （3）知识加工层

知识加工层承担了对原始文件的深入理解与组织工作，是系统“喂知识”的核心步骤。目前的设计中，该层通过文件解析模块提取纯文本内容，并进行一定程度的清洗与分段。不同于传统RAG系统使用向量化索引进行知识召回的方式，本系统将解析后的文本作为系统级提示信息（system prompt）直接传递给大语言模型，在问答过程中引导模型给出基于知识上下文的回答。

该方式降低了系统对嵌入库和向量数据库的依赖，减少了基础设施搭建的复杂性，适合轻量化部署场景。与此同时，解析结果也会被存储进数据库，支持知识内容的可视化管理与手动维护。

#### （4）问答引擎层

问答引擎层是整个系统的智能核心，负责构建模型调用所需的上下文信息，并封装对大语言模型的API调用流程。其中，Prompt构建模块根据用户输入问题、Bot配置信息、知识库内容等构造出符合预期的Prompt结构，融合知识背景提升问答准确性。

LLM API封装模块对接包括DeepSeek API、OpenAI等多个大型语言模型服务，封装了统一的请求结构与响应解析逻辑，便于切换模型提供商或实现多模型并行选择。对于用户而言，该部分的切换是透明的，只需关注最终问答效果。流式回答协议模块实现了模型响应的分段推送机制，通过SSE技术将模型的中间输出实时反馈给前端用户，提高了交互响应速度和体验流畅性。

#### （5）数据存储层

系统使用关系型数据库MySQL进行数据存储。数据库表结构包括用户信息、知识库结构、上传文件记录、解析文本内容、问答历史记录以及Bot配置信息等多个维度，实现了用户数据的持久化与管理。所有模块间的数据交互最终都会落地到数据库中，支撑系统的持续运行与审计需求。

数据库结构设计注重规范性与可扩展性，如问答历史表中记录每轮问答的时间戳、上下文内容、所属Bot与知识库等，便于后续功能如问答检索、用户分析等开发。

总体而言，本系统采用了清晰的分层结构与现代化的前后端技术栈，满足了从数据接入到智能问答的全流程需求。相较于传统QA系统，本系统更加轻量化，依赖少、部署灵活，特别适合在企业内部、教学实验、知识助手等中小规模场景落地应用。未来可在当前架构基础上进一步扩展向量化模块、知识图谱支撑、多模态输入等能力，以增强系统的鲁棒性与智能性。

### 4.1.2 数据流设计

用户首先在系统的Web前端界面中输入问题，前端将用户输入的内容以及所选上下文信息封装为标准POST请求，并发送至后端服务。

后端接收到请求后，会首根据所选择的知识库ID，从数据库中检索出该知识库中解析后的全部文本内容，并将这些内容作为上下文信息拼接进系统Prompt。

构建好完整的Prompt后，后端将“用户问题+知识库内容”一并发送至大模型。由于系统没有引入向量化检索，因此全部知识内容会被直接插入到Prompt的System角色段中，以使大模型获得尽可能全面的参考背景。

大模型处理完请求后，会生成针对用户问题的自然语言回答，并将答案返回给后端。为了提升响应速度和用户体验，后端采用SSE（Server-Sent Events）协议将答案逐段流式地推送给前端，实现内容的实时渲染展示。用户可以边看边读，避免长时间等待完整返回结果。

同时，系统也会将本次问答记录同步写入数据库，包括用户提问内容、模型生成的回答、所使用的知识库ID、时间戳、用户标识以及Bot ID等信息。这一部分数据为用户历史记录功能提供支撑，也便于后期管理端进行cr问答质量分析与内容审计。

前端接收到后端流式返回的文本后，会持续监听SSE事件流并动态拼接完整答案，最终将渲染好的结果展示在对话框中。整个过程中，用户感知到的是一个近似于ChatGPT式的流畅问答体验。数据交互图如图4-2所示。

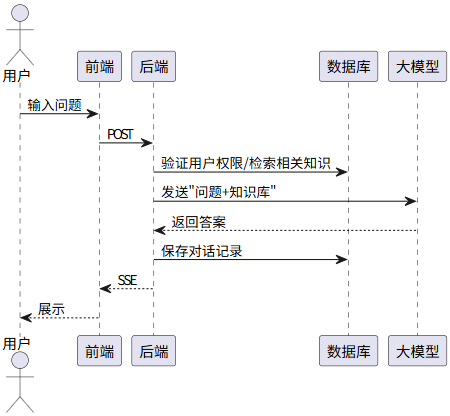


图4-2 数据交互图

## 4.2 数据库设计

系统的数据库采用MySQL进行设计与实现，整体结构围绕用户、知识库、文件管理、聊天记录与Bot配置五大核心功能展开，保证数据存储的完整性与查询的高效性。

### 4.2.1数据库实体关系设计

ER图展示了系统中六个核心实体之间的关系，如图4-3所示。包括用户User、知识库Knowledge、文件File、聊天记录ChatRecord、聊天机器人Bot。每个实体通过主键唯一标识自身，通过外键与其他实体形成逻辑关联，支持系统的权限控制、知识查询、对话存储等功能。

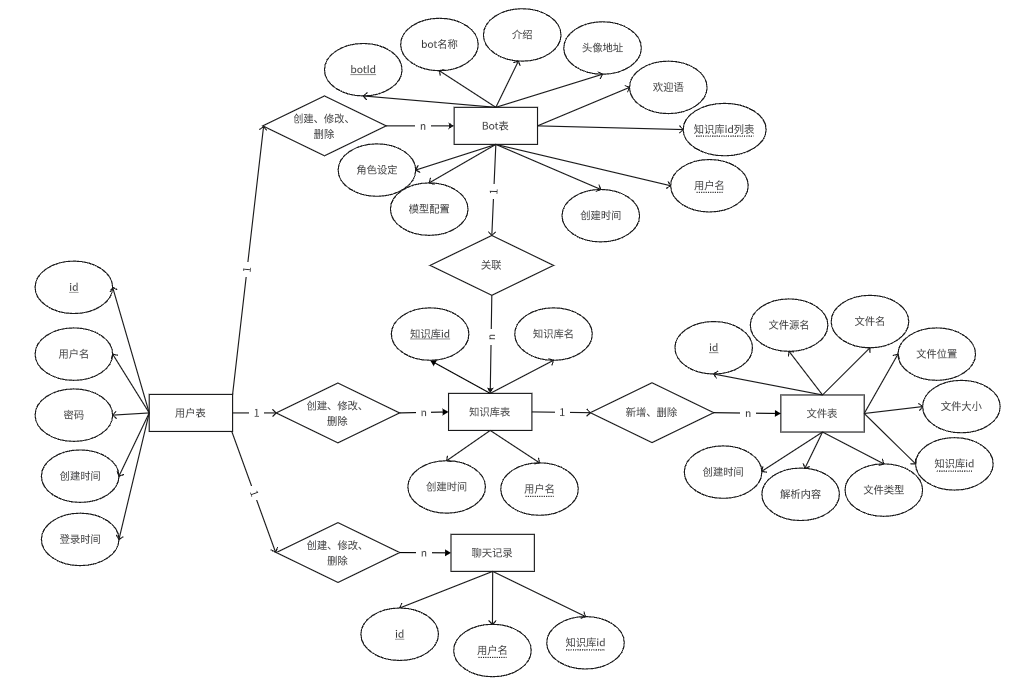


图4-3 ER图

用户User是系统的核心，每个用户通过唯一的username与多个知识库Knowledge、聊天机器人Bot和聊天记录ChatRecord关联。用户的id为自增主键，username是唯一字段，用于建立所有外键关系。

知识库Knowledge用于组织结构化或非结构化的知识信息，由用户创建。每个知识库通过kbId唯一标识，并通过username与User表建立外键联系，说明一个用户可以拥有多个知识库。

文件File属于知识库，是知识内容的具体载体。每个文件通过kbId关联到所属的知识库，支持多对一的关系，即一个知识库可以包含多个文件。文件信息包括原始文件名、实际存储名、路径、大小、类型、内容等字段。

聊天记录ChatRecord记录了用户与机器人或系统的对话历史。每条记录通过username和kbId与User和Knowledge建立外键关联。ChatRecord的messages字段以JSON字符串形式存储完整的多轮对话内容，type字段用于区分不同对话模式，如“quick”或“home”。

聊天机器人Bot由用户创建，并绑定一个或多个知识库以支持问答任务。机器人通过username与用户关联，字段kbIds是字符串数组，存储多个知识库ID，实现一个机器人对多个知识库的访问能力。机器人还包含头像、欢迎语、角色设定、聊天配置等属性，以支持自定义化的人机交互体验。

整个ER图反映出以用户为中心的设计理念，用户创建知识库、上传文件、构建机器人并产生对话记录，各模块之间通过清晰的外键关系串联，确保数据的完整性和操作的权限边界。

### 4.2.2数据库逻辑结构设计

#### 用户表

首先是用户表，每一位用户都拥有唯一的自增主键id，记录其身份凭证的是username和password字段，系统所支持的基于角色的权限控制机制，利用role字段区分不同的用户类型，默认类型为“user”，该表另外还包含createTime与updateTime字段，这两个字段用来记录用户的注册时间以及最近一次信息变更时间，以便于管理员查看系统用户活跃度。

表4-1用户表

| 字段名 | 类型 | 说明 |
| --- | --- | --- |
| id | INT | 主键，自增 |
| username | VARCHAR | 用户名 |
| password | VARCHAR | 用户密码 |
| role | VARCHAR | 用户角色 |
| createTime | VARCHAR | 创建时间 |
| updateTime | VARCHAR | 更新时间 |

#### 知识库表

知识库表用于管理知识库的基本信息。表中的主键为kbId，采用字符串类型以支持灵活生成。知识库名称保存在kbName字段，type字段用于标识知识库类型，例如“quick”表示快速问答知识库，“normal”表示标准知识库。每条知识库数据通过username字段关联到具体的用户，体现出知识库对用户的归属关系。

表4-2知识库表

| 字段名 | 类型 | 说明 |
| --- | --- | --- |
| kbId | VARCHAR | 知识库主键 |
| kbName | VARCHAR | 知识库名称 |
| type | VARCHAR | 类型 |
| createTime | VARCHAR | 创建时间 |
| username | VARCHAR | 关联用户（外键） |

#### 文件表

文件表主要是用来管理用户上传的原始文件以及解析之后的内容，其中每个文件都有自增主键id，文件的原始名称与实际存储名称，分别记录在originalName和fileName字段里面，这样做是为了方便前后端进行映射静态资源，文件的物理存储路径是filePath，而fileSize与mimeType字段可提供文件的大小以及类型方面的信息。在对文件内容进行处理的时候，系统会把解析后的纯文本内容保存在content字段，并且依靠kbId建立起与知识库的外键关联，这个表还会记录上传的时间，记录用户活跃度。

表4-3文件表

| 字段名 | 类型 | 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 |

#### 历史对话表

历史对话表用于记录用户的所有对话历史。每条记录具有唯一id，通过username关联用户，kbId用于关联知识库，支持对话与知识上下文的联动。type字段表明聊天记录的类型，例如“quick”表示快速问答模式，“home”表示主页普通对话模式。系统将每次对话以JSON数组格式序列化后存储在messages字段中。

表4-4历史对话表

| 字段名 | 类型 | 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） |

#### 聊天Bots表

Bots表为系统中的智能机器人配置表。每个Bot拥有唯一标识botId和名称botName，用户可以配置机器人的头像avatar、简介description、欢迎语welcomeMessage、角色设定roleSetting等参数。通过username字段将Bot与用户绑定，确保每个用户只能管理自己的Bot。一个Bot可以关联多个知识库，系统通过kbIds字段使用字符串数组形式存储多个知识库ID，从而支持复杂场景下的知识融合问答。此外，chatSetting字段以JSON格式存储Bot的个性化配置调优。

表4-5 Bots表

| 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 |
| description | 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

前端把主流框架Vue3用作核心开发框架，依据TypeScript语言特性构建出响应式用户界面，其主要承担系统跟用户的交互功能实现，于UI组件层面，挑选阿里巴巴开源的Ant Design Vue当作基础UI库，此组件库给予丰富的企业级UI组件以及设计规范。系统整体布局运用antdv推荐的顶部导航栏搭配左侧边栏这种设计模式，保证了导航有直观性，还提供了充裕的功能展示空间，从功能架构方面来看，系统前端主要被划分成三大核心模块，分别是快速开始问答、知识库问答以及聊天Bot管理，本文会依据这三个主要功能模块展开详尽说明，还要深入剖析每个功能点实现过程中所运用的关键前端技术。

### 5.1.1 登录注册

登录注册页面采用Flex横向布局设计，如图5-1所示，整体界面分为左右两个功能区域。左侧区域采用大尺寸Logo，右侧区域为登录注册表单，通过antdv提供的Form组件构建。

采用Form组件内置验证机制，配置了严谨的校验规则，如判空、用户名长度、密码字符类型以及注册时的二次密码校验。一切校验通过才允许提交表单。

密码校验核心代码如下：

pattern: /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,20}$/,

message: '密码必须包含大小写字母、数字和特殊字符，长度为8-20字符',

trigger: 'blur',



图5-1登录注册页

### 5.1.2 快速开始

“快速开始”功能的设计理念是让用户能够零门槛体验问答服务。用户完成注册后即可直接开始对话，无需进行复杂的模型配置或文件上传操作，其功能定位为DeepSeek-V3模型。同时，系统也提供了更丰富的功能选项。

1. 开启新对话

快速开始整体布局为左右布局，左侧为对话列表操作区，右侧为问答交互区。在左侧面板中，用户可以通过“开启新对话”按钮创建新的知识库会话，并为其绑定一个知识库。每新增一个对话项，左侧列表会通过Vue的v-for指令实现动态渲染和纵向扩展，确保界面元素与数据状态保持同步。

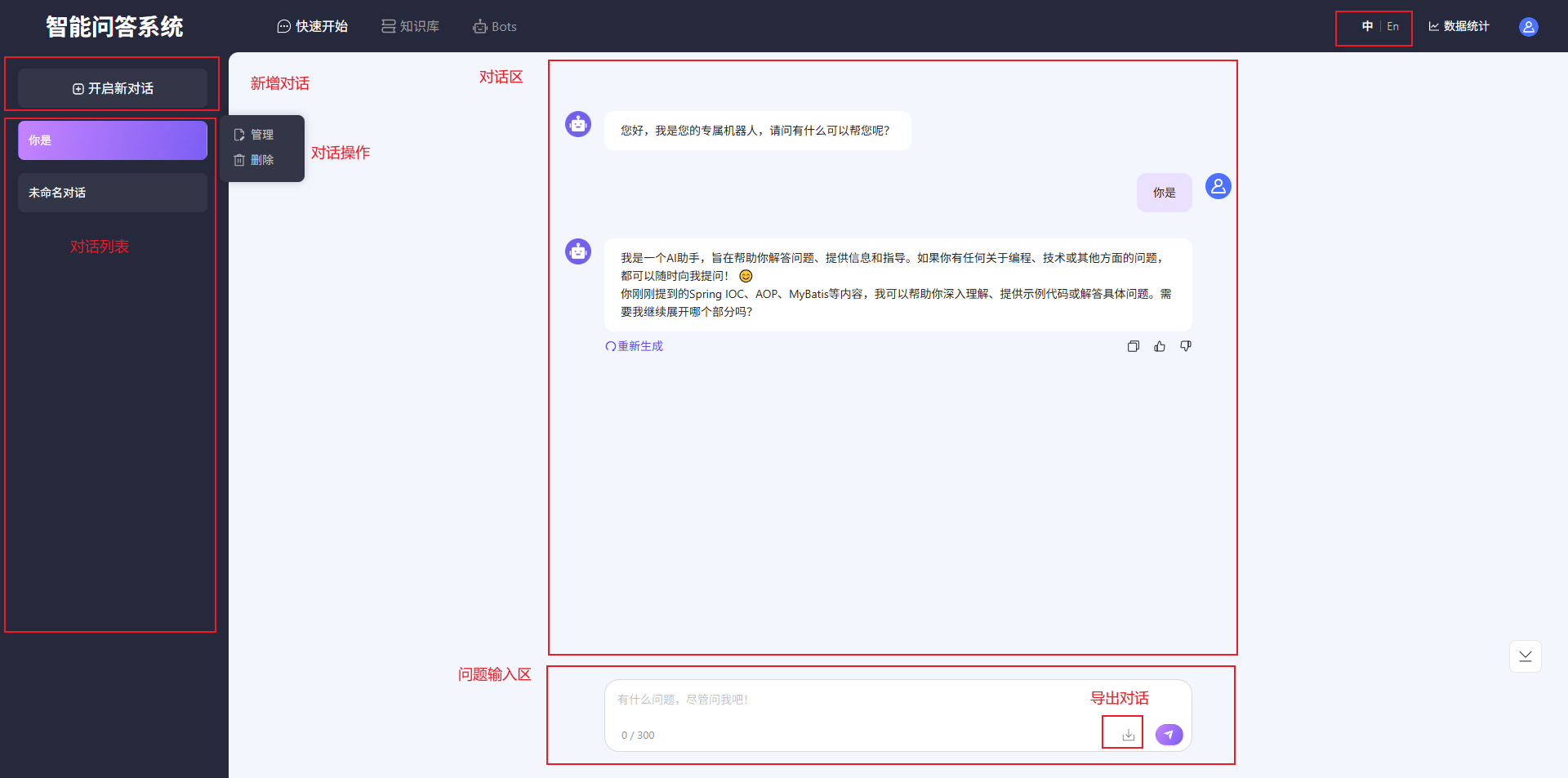
对话列表项采用交互式设计，用户左键点击任意项即可选中当前对话，此时该项会通过动态绑定class的方式呈现高亮状态，同时右侧面板会自动加载该对话的历史记录。当用户将鼠标悬停在列表项上时，会出现“管理”和“删除”两个功能按钮，分别用于进入文件管理界面和对话删除操作。快速开始界面如图5-2所示。

图5-2快速开始界面

在问题输入框中输入问题并回车，即可与模型交互，为了实现逐字输出效果，优化用户在对话过程中的实时交互体验。该方案采用SSE，结合前端动态渲染技术，模拟“打字机”效果，用户可以在回答后马上看到回复，有效减少用户等待焦虑，提升交互流畅度。

SSE基于HTTP长连接，服务端可主动向客户端推送事件流（text/event-stream）。前端通过fetchEventSource发起请求，通过AbortController（signal:ctrl.signal）实现请求中断，利用typewriter对象动态追加文本实现逐字输出的“打字机”的效果。

前端核心实现如下：

fetchEventSource(apiBase + '/chat/stream', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

Accept: 'text/event-stream, application/json',

},

body: JSON.stringify({

username,

...sendData,

}),

signal: ctrl.signal,

async onopen(e: any) {

addAnswer(q);

if (e.ok && e.headers.get('content-type') === 'text/event-stream') {

typewriter.start();

} else if (e.headers.get('content-type') === 'application/json') {

typewriter.add('Error 请检查模型是否配置正确');

}

},

onmessage(msg: IMessage) {

console.log('message', msg);

if (msg.data !== '[DONE]') {

const res: any = JSON.parse(msg.data);

if (res.content) {

typewriter.add(res.content);

scrollBottom();

}

}

},

onclose() {

typewriter.done();

ctrl.abort();

},

onerror(err: any) {

typewriter.done();

ctrl.abort();

throw err;

},

});

向目标服务器发送POST请求，设置请求头“Content-Type:application/json”声明发送 JSON数据“Accept:text/event-stream”表明客户端支持SSE，请求体携带用户信息和一些模型配置用AbortController的信号允许后续中断请求。连接建立(onopen)，首先调用addAnswer(q)在前端界面添加问题展示。检查响应，如果是SSE流(text/event-stream)，启动打字机动画，否则直接显示错误提示。消息处理(onmessage)忽略结束标记[DONE]，解析每条消息的JSON数据：当有内容(res.content)时，通过打字机逐字显示(typewriter.add())。

当服务端停止发送数据流，则代表本轮回答结束，结束打字机动画，连接关闭(onclose)。如果服务因为网络或者其他原因报错，前端接受到报错信息，在onerror回调中做响应的报错处理，中断请求，并关闭连接。

对话结束后，允许用户对模型回答做操作，如图5-3所示，像重新回答、复制回答内容等操作，复制功能实现采用@vueuse/core提供的useClipboard组合式函数实现复制功能，该函数底层调用浏览器的Clipboard API，提供了良好的兼容性和用户体验。

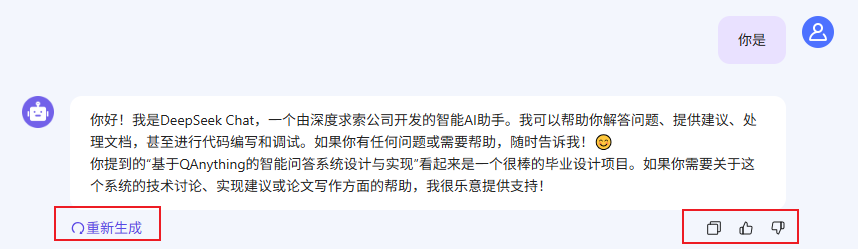


图5-3回答操作栏

在问题输入框的操作栏，允许我们将问答内容导出为图片，如图5-4所示。通过获取到的聊天dom，用html2canvas这个三方库即可将我们的dom使用canvas画出来，然后往dom树中丢一个带连接的a标签，在模拟一下点击，这一翻操作即可触发浏览器的下载。

核心代码如下：

const ele = document.getElementById('chat-ul');

const canvas = await html2canvas(ele as HTMLDivElement, {

useCORS: true,

});

const imgUrl = canvas.toDataURL('image/png');

const tempLink = document.createElement('a');

tempLink.style.display = 'none';

tempLink.href = imgUrl;

tempLink.setAttribute('download', 'chat-shot.png');

if (typeof tempLink.download === 'undefined') tempLink.setAttribute('target', '\_blank');

document.body.appendChild(tempLink);

tempLink.click();

document.body.removeChild(tempLink);

window.URL.revokeObjectURL(imgUrl);

message.success('下载成功');

Promise.resolve();



图5-4问题输入操作栏

1. 知识库管理

点击“管理”会跳转至管理页即进入到该问答的知识库，使用<view-router>标签实现页面路由的切换。在文件管理页面中，用户可通过左上角的返回按钮快速切换回对话界面。页面顶部清晰展示了当前对话的标题和关联知识库ID等元信息。右侧区域提供文件上传功能。页面下方以表格形式列出该对话关联的所有文件，使用antdv的Table组件，包含文档ID、原始文件名、服务器存储名、文件大小和MIME类型等详细信息。表格最后一列提供操作按钮，支持文件的删除和预览功能。文件管理界面如图5-5所示。

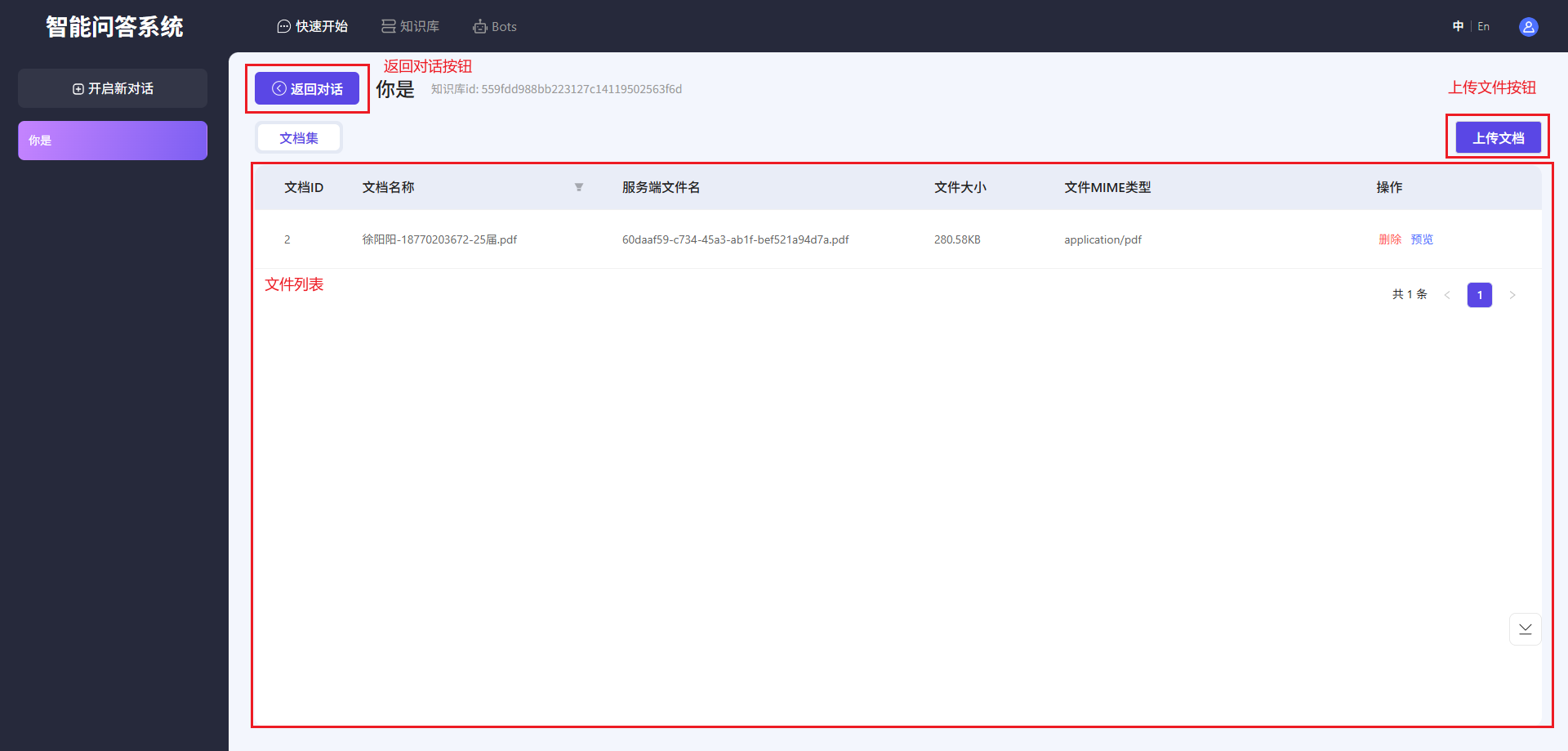


图5-5文件管理界面

点击“上传文档”按钮会调起上传文件弹窗，如图5-6所示，弹窗内容详细介绍了支持的文件格式以及大小限制：单个文档不超过30MB，图片文件不超过5MB。支持拖拽以及点击调起文件选择器以获取文件内容，目前只支持单个文件上传。



图5-6文件上传弹窗

这里使用的是type为file的input框，将文件拖入input中，或者点击即可调起系统文件选择器。选择完文件，系统会自动请求上传接口，将文件转换成formData格式，指定请求头需设置：Content-Type: multipart/form-data。

<input

type="file"

accept="acceptList.join(',')" <!-- 限制文件类型 -->

@change="upload"

@click="(e) => ((e.target as HTMLInputElement).value = '')"

/>

在对文件进行操作时，我们可以选“预览”功能，这里主要涉及两种预览方式：第一种是采用弹窗预览，第二种是跳转至新页面预览。考虑到实际使用场景中，用户需要边查看文件内容边进行提问或操作，为了提供更便捷的查看体验，避免频繁开关弹窗带来的操作中断，最终选择了新开窗口预览的方式。具体实现上，当用户点击预览按钮时，系统会打开一个新的浏览器窗口，在前端通过iframe标签嵌入文件内容，将iframe的src属性直接指向服务器对应的静态资源URL。这种方式能够很好地支持大部分文本文件（如txt、jsonl等）以及PDF的预览。

这里存在一个问题：iframe仅支持部分文件[9]，例如常见的docx、pptx、xlsx等Office文件格式都不在支持范围内。为了解决这个问题，我采用了条件判断的方案：首先通过文件名的后缀来判断文件类型，如果文件不属于docx、pptx、xlsx这三种格式，则继续使用iframe进行渲染；如果是这三种Office文件格式，则使用对应的预览库进行处理。经过调研，在npm包管理平台中，找到了这三者对应的适用于Vue项目的包。分别是“@vue-office/docx”、“@vue-office/ pptx”和“@vue-office/excel”。再借助Vue的自定义组件功能，实现了自动识别文件类型并使用对应组件进行渲染的方案。

### 5.1.3 知识库问答

知识库问答模块相比于快速开始，功能上没有很大的差异，但在配置项方面进行了显著优化和提升。整个知识库问答采用统一对话机制，支持同时管理多个知识库，用户可以手动选择提问的知识库（背景紫色代表选中），并且支持多知识库联合查询，针对模块化的数据又增加一定耦合。

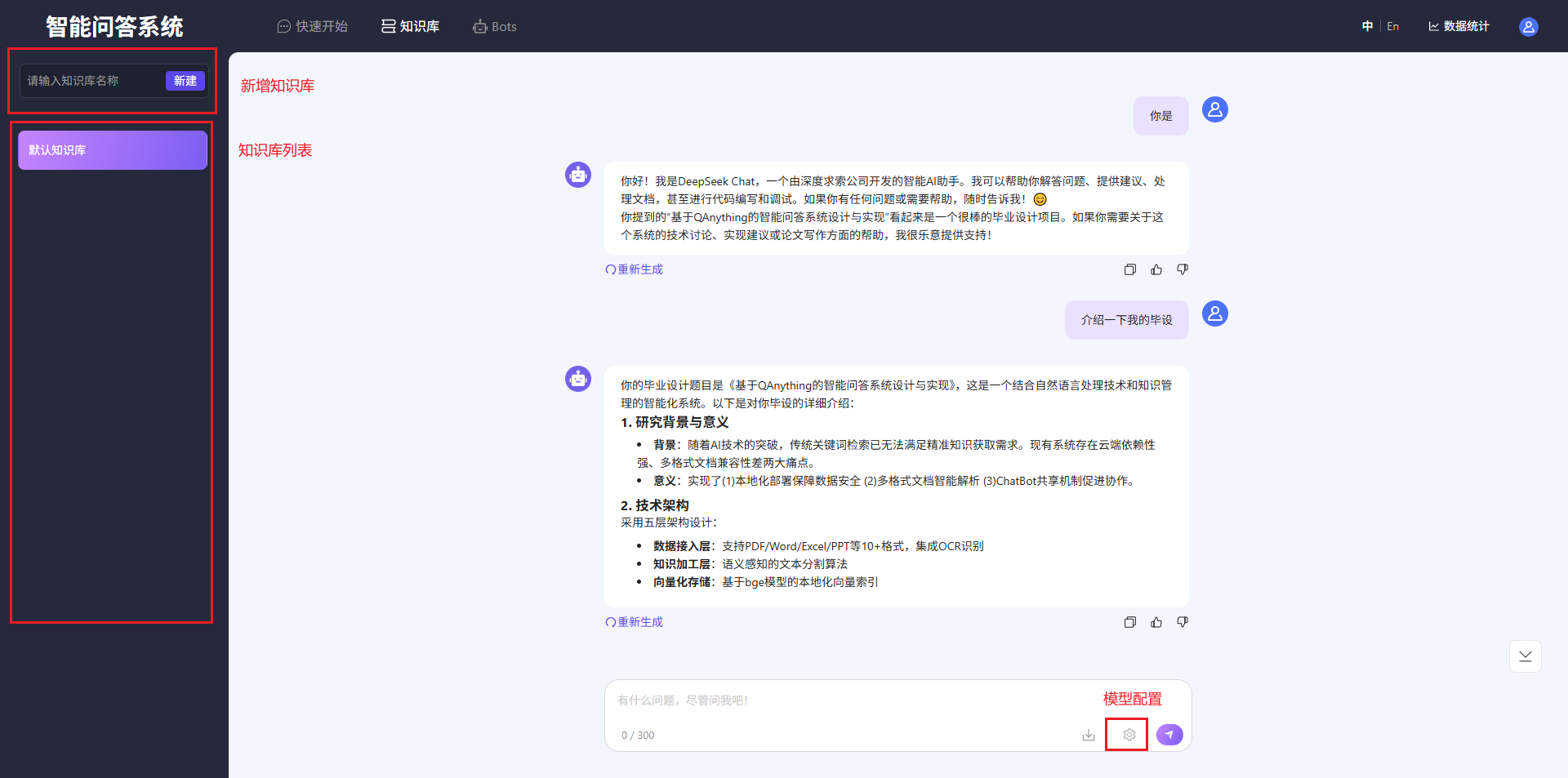
在知识库管理栏，用户仅需输入知识库名称然后点击“新建”按钮，就可迅速创建知识库，并且还支持对已存在的知识库开展重命名操作，在问答操作栏，新增加了“模型配置”选项，它能让用户依照自身需求对模型参数做个性化配置，以此得到更加精准且更契合特定场景的问答结果。这般一来，知识库问答模块在维持原有核心功能的前提下，提供了更高的灵活性与可定制性，知识库问答界面如图5-7所示。

图5-7知识库问答界面

相比于快速开始模块，知识库模块为用户提供了更灵活的模型配置能力。不仅限于快速开始模块系统内置的DeepSeek-V3模型，还可以根据需求自由拓展和配置其他模型。

整个配置采取弹窗形式，如图5-8所示，内容采用清晰的Form表单以引导用户完成配置，表单字段类型丰富，“模型提供方”使用下拉菜单形式，提供可选项OpenAI、Ollama等，普通输入框、密码框、滑动条以及数字输入框，提供用户全面的交互方式。可以配置输出token数、随机性、积累概率阈值、上下文长度，分别表示限制单次响应的最大Token数量、生成结果的随机性、生成内容的聚焦程度、模型读取的上下文长度。所有模型配置都会自动持久化保存到浏览器本地存储（localStorage）中，确保用户无需在每次使用时重新配置，增强用户体验。

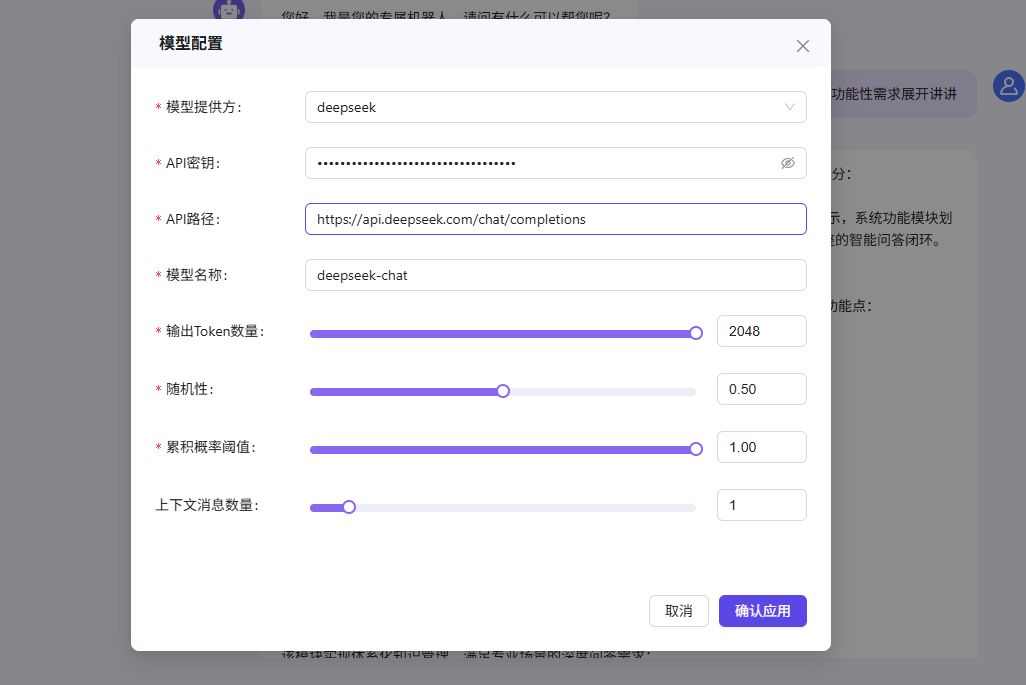


图5-8模型配置弹窗

### 5.1.4 Bots

Bots模块是在知识库问答系统基础上构建的智能化对话代理，通过一套完整的配置体系实现了知识服务的个性化封装。该模块采用“预设”+ “自定义”的模式，系统内置了多个专业领域的预配置Bot（如技术顾问、法律助手、医疗咨询等），每个Bot都经过个性化配置，具备领域特定的欢迎语、角色设定和Prompt模板。

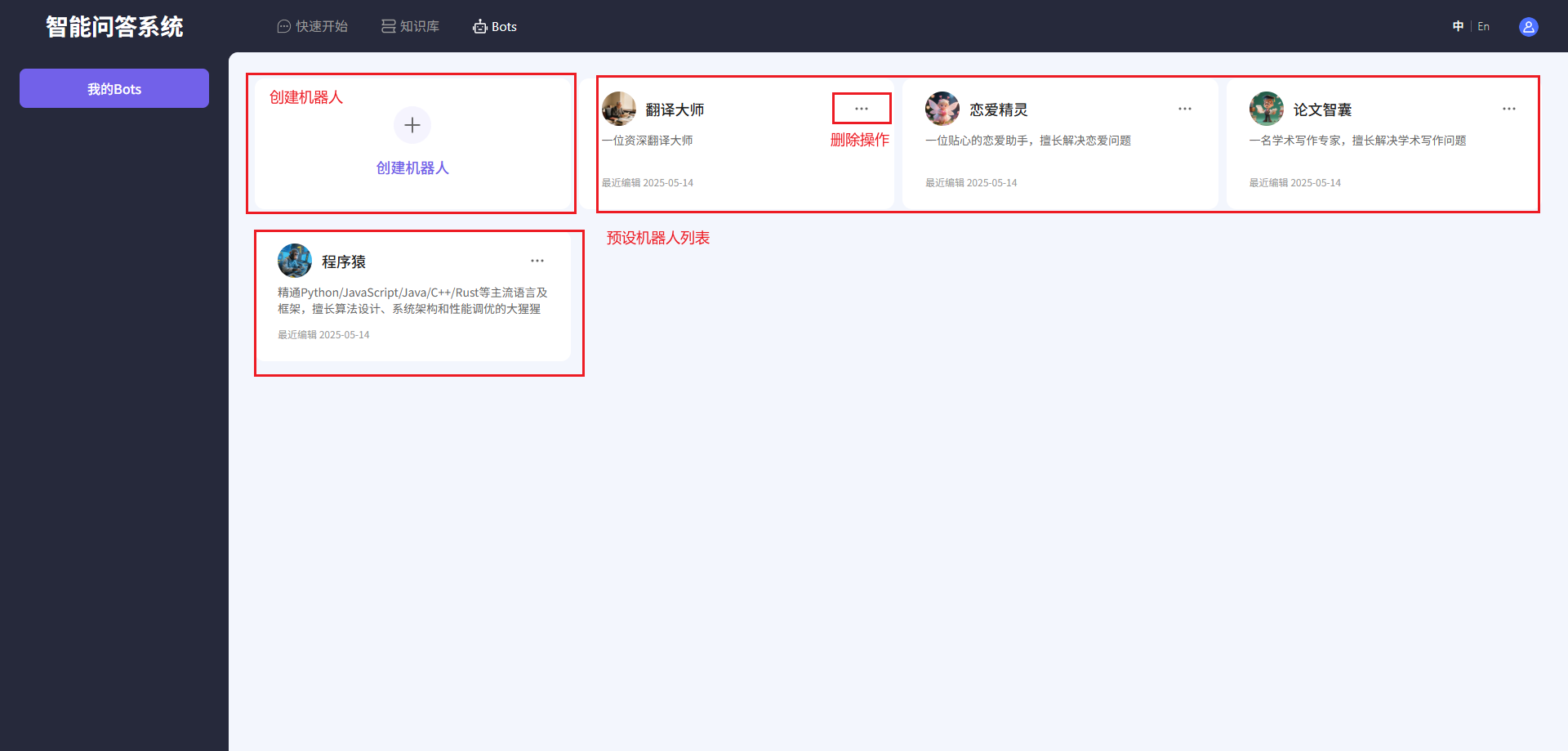
用户仅需完成名称定义、头像上传、功能描述等基础配置即可生成专属智能助手。这种设计为用户提供了充分的定制空间。Bots列表界面采用卡片式布局，展示各Bot的图标、名称和简介，如图5-9所示。

图5-9 Bots列表页

点击对应Bot会跳转到该Bot的详情页，如图5-10 所示，页面采用左右分栏布局设计：左侧展示Bot的信息配置区，右侧为实时对话测试区。这种布局设计使用户能够一边调整配置参数，一边即时测试Bot的响应效果，实现所见即所得的交互体验。

关于Bot的配置项，从Bot的基本信息：头像、名称、欢迎语，再到与模型交互所需的系统角色设定、模型配置（如输出token数、随机性、积累概率阈值、上下文长度等），甚至支持动态关联知识库，提供给用户一个自由度极高的Bot搭建平台。

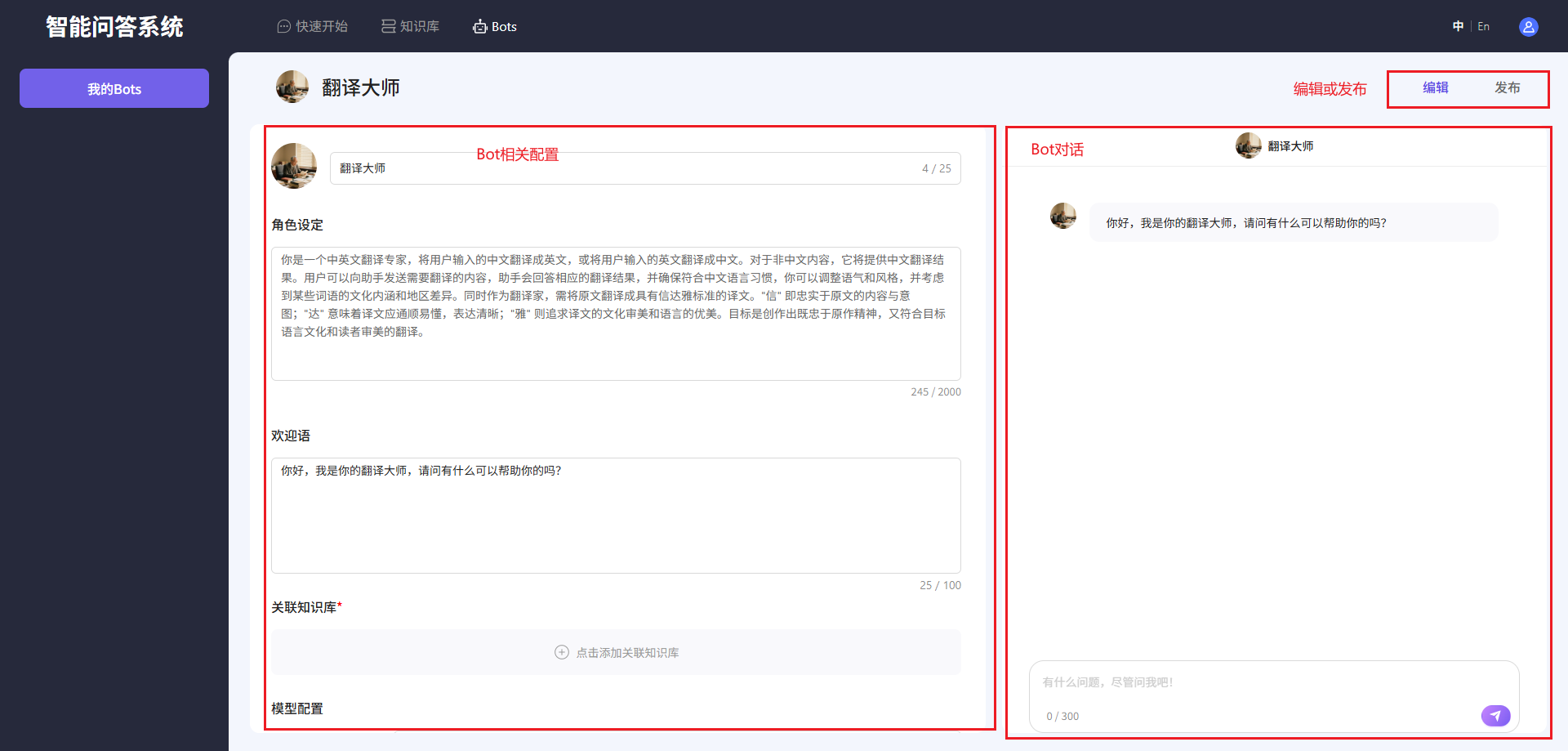


图5-10 Bots配置页

Bots模块中最核心的功能莫过于分享机制，该功能允许用户将创建的Bot以二维码或者链接的形式发布和分享。通过这两种方式，任何打开这个链接或者扫描二维码的用户，只需简单的注册登录进入系统，即可访问该Bot，直接进行对话交互。Bots发布页与分享对话页如图5-11所示。

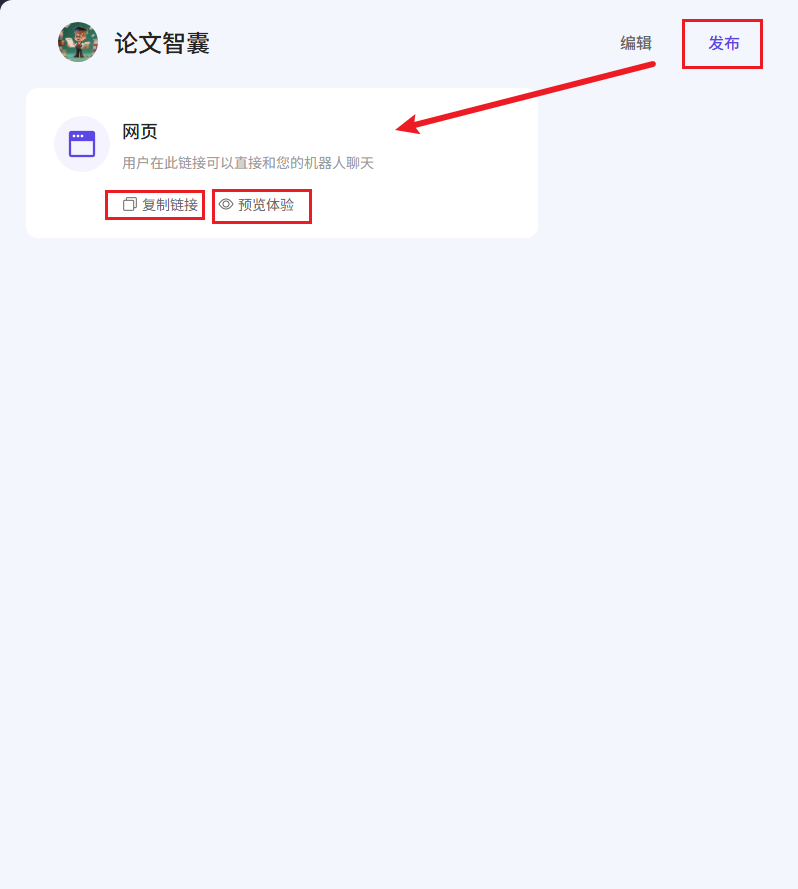


图5-11 Bots发布页与分享对话页

分享链接的实现方案得益于Vue-router的动态路由功能，通过将唯一的botId作为路由参数拼接至基础URL，即可生成具有唯一性的Bot分享链接，例如“/share/bot/:botId”；二维码的生成则通过antdv提供的a-qrcode组件实现，只需将该组件的value属性绑定至上述分享链接，即可自动生成对应的可扫描二维码。

## 5.2 后端实现

后端基于主流框架Nestjs实现，它是一个使用TypeScript的渐进式Node.js框架，它采用了清晰的分层架构设计，非常适合构建高效、可扩展的服务器端应用程序。

### 5.2.1 分层架构的实现

在技术综述时已经介绍了NestJS的分层架构，通过控制器层、业务逻辑层、数据访问层的明确分离，结合模块化设计，构建了高内聚低耦合的系统架构在。本系统在分层架构的加持下，代码质量和开发效率得到了飞速提升。代码的目录结构如下：

src/

├── modules/ # 所有功能模块统一管理

│ ├── users/ # 用户模块

│ │ ├── dto/ # 数据传输对象

│ │ │ ├── users.dto.ts

│ │ ├── entities/ # 实体类

│ │ │ └── user.entity.ts

│ │ ├── users.controller.ts

│ │ ├── users.service.ts

│ │ └── users.module.ts

│ │

│ ├── knowledges/ # 知识库模块（同user结构）

│ ├── chats/ # 聊天模块

│ ├── files/ # 文件模块

│ └── bots/ # Bot模块

├── common/ # 公共资源

│ ├── utils.ts/ # 工具函数

│ ├── fileParser.ts/ # 文件解析相关

├── config/ # 配置文件

│ └── multer.config.ts

└── main.ts # 应用入口

下面以用户管理模块为例，详细阐述了基于NestJS框架的分层架构实现方案。

#### 表现层实现

用户控制器通过构造函数注入服务实例，每个公共方法对应一个API端点。如用户注册接口实现：

@Post('register')

@UsePipes(new ValidationPipe())

async register(@Body() dto: CreateUserDto) {

return this.userService.register(dto);

}

使用@Body()自动解析请求体，通过ValidationPipe进行参数校验，业务逻辑完全委托给服务层

#### 业务逻辑层实现

用户服务类包含核心业务规则，典型实现如下：

async register(dto: CreateUserDto) {

// 业务规则验证

const exists = await this.userRepo.exist({

where: { username: dto.username }

});

if (exists) throw new ConflictException('用户已存在');

// 业务操作

const user = this.userRepo.create(dto);

await this.userRepo.save(user);

// 生成业务结果

return {

token: this.jwtService.sign({ sub: user.id }),

userInfo: \_.omit(user, ['password'])

};

}

#### 数据访问层实现

通过TypeORM的Repository模式提供数据持久化能力：

@EntityRepository(User)

export class UserRepository extends Repository<User> {

async findActiveUsers(): Promise<User[]> {

return this.find({ where: { isActive: true } });

}

}

总而言之，通过模块化组织实现功能内聚，依赖注入机制降低耦合度，清晰的层级边界提升可维护性，完善的异常处理保障系统健壮性。

### 5.2.2 模型对话服务

模型对话服务采用流式响应设计，基于Server-Sent Events（SSE）技术构建实时交互能力。服务通过灵活的配置架构同时支持原生API调用和Bot代理模式，当请求携带botId参数时会自动加载预设的对话配置，包括API端点、模型参数、知识库关联和角色设定等。核心处理流程首先构建包含系统提示、知识库内容和历史消息的完整对话上下文，其中知识库内容通过文件服务动态检索注入系统消息。

服务采用Axios的流式请求方式与底层模型API交互，实现逐块处理响应数据并实时转发给客户端，过程中严格遵循SSE协议规范，每个数据块都包含标准格式的事件流消息。错误处理机制覆盖了从API请求到流式传输的全链路，确保任何环节出现异常都能向客户端错误信息。

核心代码如下：

axiosResponse.data.on('data', (chunk: Buffer) => {

const lines = chunk

.toString()

.split('\n')

.filter((line) => line.trim() !== '');

for (const line of lines) {

if (line.includes('[DONE]')) {

response.write('data: [DONE]\n\n');

return;

}

if (line.startsWith('data: ')) {

try {

const jsonData = JSON.parse(line.slice(6));

const content = jsonData.choices?.[0]?.delta?.content || '';

if (content) {

response.write(`data: ${JSON.stringify({ content })}\n\n`);

}

} catch (e) {

response.write(

`data: ${JSON.stringify({

error: '解析响应数据出错',

details: e.message,

})}\n\n`,

);

}

}

}

});

服务设计充分考虑了扩展性，模型参数、能力开关等均可通过请求参数动态调整，为不同类型的对话应用场景提供了统一的接入接口。

### 5.2.3 文件解析服务

基于Node.js的多格式文件解析服务实现方案。该服务支持PDF、Word、Excel、PPT、文本文件及图片等多种格式的内容提取，通过模块化设计和多种解析库的组合应用，提供了统一的文件内容提取接口。

在PDF解析方面采用pdf-parse库实现文本提取，Word文档处理使用mammoth库保留基础格式信息，Excel文件通过xlsx库转换为CSV格式输出，PPT内容借助officeparser库提取，而图片OCR识别则基于Tesseract.js引擎实现中英文字符识别[11]。

服务采用标准的Promise异步接口设计，内部通过文件扩展名自动路由到对应解析器，并实现了统一的错误处理机制，将底层各类解析异常转换为服务级错误信息。

在实现细节上，针对不同文件类型特点采用了差异化的处理策略，如图片解析时特别注意了OCR工作线程的创建和释放，确保资源及时回收；同时兼顾了同步文件读取和异步内容解析的性能平衡。

核心代码如下：

async function extractImageContent(filePath: string): Promise<string> {

const worker = await createWorker();

try {

await worker.reinitialize('eng+chi\_sim');

const { data } = await worker.recognize(filePath);

return data.text;

} finally {

await worker.terminate();

}

}

## 5.3 部署

部署部分包括前后端服务的构建与上线。前端项目经过打包后托管在Nginx或静态资源服务中。后端服务容器化打包，使用Docker部署，配合Nginx实现反向代理，支持HTTPS与接口转发。数据库部分选择MySQL部署。最终部署环境选择华为云服务器，并结合日志监控系统稳定运行。

### 5.3.1 前端挂载

整个前端部署流程从构建到Docker容器运行，首先使用pnpm build命令将前端项目打包，生成静态文件并输出到dist目录中。接着通过Dockerfile构建镜像，基于最新的nginx官方镜像，设置工作目录为/app，将dist目录下的所有文件复制到nginx默认的静态文件目录/usr/share/nginx/html中。

Docker配置如下：

FROM nginx:latest

EXPOSE 81

WORKDIR /app

COPY dist/ /usr/share/nginx/html

COPY nginx.conf /etc/nginx/conf.d/default.conf

CMD ["nginx","-g","daemon off;"]

同时将自定义的nginx.conf配置文件覆盖默认配置。nginx配置中特别设置了监听81端口（与Dockerfile中EXPOSE的端口一致），配置了访问日志和错误日志路径，并通过try\_files指令实现前端路由的History模式支持，确保直接访问子路由或刷新页面时能正确返回index.html。

nginx.conf配置如下：

server {

listen 81;

server\_name localhost;

access\_log /var/log/nginx/access.log main;

error\_log /var/log/nginx/error.log error;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

# 浏览器请求的资源不存在时，返回404

try\_files $uri $uri/ /index.html;

}

# error\_page 404 /404.html;

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

当运行该Docker容器时，nginx会以前台模式启动（通过daemon off参数），持续监听81端口请求。对于所有静态资源请求，nginx会从/usr/share/nginx/html目录提供文件；当请求的路由不存在时会回退到index.html交由前端路由处理。错误处理方面配置了50x错误的默认页面，但注释掉了404自定义页面。整个部署方案通过容器化实现了环境一致性，利用nginx高效地提供静态文件服务，并通过合理的配置解决了单页应用的路由问题。

### 5.3.2 服务端部署

服务端部署的流程与前端类似，首先同样pnpm build命令将前端项目打包，生成静态文件并输出到dist目录中，针对Nestjs后端应用的特点进行了调整,基于最新版的Node官方镜像构建，暴露服务运行的3002端口，并设置工作目录为/app。通过COPY指令将整个项目代码复制到容器内，然后运行npm install安装依赖。最后通过CMD启动编译后的入口文件dist/main.js

核心代码如下：

FROM node:latest

EXPOSE 3002

WORKDIR /app

COPY

RUN npm install

CMD ["node","dist/main.js"]

### 5.3.3 数据库部署

我们需要在服务器上安装MySQL8.0镜像，接着执行运行容器的指令：

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

通过一条命令就能快速部署一个功能完整的MySQL服务。这里配置了root用户密码为"root"，并创建了名为"mydb"的数据库，同时设置了普通用户访问密码。服务端口映射将容器内的3306端口直接暴露到宿主机的3306端口，确保外部应用可以方便连接。数据持久化方面，通过volume挂载将容器内的/var/lib/mysql目录映射到名为mysql\_data的Docker卷，保证数据库文件在容器重启或删除后依然保留。

这种部署方式轻量快速，大大简化了传统数据库安装配置的复杂流程。

结束语

在大模型飞速发展的当下，一个高质量的知识库问答系统，是企业中绕不开的话，而本设计围绕着这一核心目标所开展的设计与实现工作，其借助Vue3+NestJS+MySQL搭建，系统可支持快速开始问答、管理知识库并问答、文件上传并解析以及机器人配置并分享等功能模块，为知识库问答系统提供了参考。

系统当前的知识检索能力仍处于基础阶段，主要依赖全文匹配，缺少更精确的语义理解与向量化召回。不仅如此，还缺少大量用户实测数据，尚未进行性能压测与用户反馈收集，部分功能仍停留在初级阶段。

未来计划引入如FAISS、Milvus等向量数据库提升知识库检索精度，实现更高质量的知识增强问答功能。在优化用户体验方面，提升聊天界面流畅度、增加对话多轮追问支持、支持知识库文件的增量更新与批量管理。有望建设后台运营系统，支持对用户数据、对话数据进行统计分析，为系统优化与推广提供决策依据。在工程化方面，通过部署CI/CD脚本，实现一键部署与多环境支持，进一步增强系统的工程实用性和部署便利性。

致 谢

至此，大学生涯就要在这里画上句号了。

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帮助。在此，我谨向所有在项目开发与研究过程中给予我灵感、指导、帮助和鼓励的人表示最诚挚的感谢。

首先，感谢QAnything给了我本次项目的开发灵感，感谢我的同事刘俊雄以及组内的同事们。在网易实习的这段经历中，从接触到这个项目，到夜以继日的学习，让我萌生了“实现一个自己的问答系统”的想法。借着空闲时间去学习它的架构，开发时请教雄哥，我不断地深挖，请教大家，最终将这个系统以毕设的形式展示给大家。

其次，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。

接着，我要感谢从2024年6月份至今，所有参与过项目合作的同事们，不论是前端、后端、测试、产品、UI等等，你们严谨的态度、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，是你们的陪伴和支持给予了我无限进步的动力。

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。

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,2024,(12):232-237.
2. 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,2024:20-36.
3. 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):1.
4. 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): 61-72.
5. 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.
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, 2025,61(07):1-24.
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.
9. 刘喆昕.网页中PDF文档展示方案的研究[J].现代信息科技,2023,7(20):18-21.
10. 赵佳英.多模态文件内容检索程序设计与实现[J].电脑知识与技术,2024,20(23):75-77.
11. 赵率宏.基于Node.js的ORM框架研究与实现[D].西南科技大学,2023:6-15.
12. 芬兰：国家档案馆推出新的开源OCR工具[J].陕西档案,2025,(02):61.