

AGENT-LLM

数据库大作业展示

胡瑞康,莫子昊,陈俊帆

大纲

1. 团队成员	2	3.3.1 表格增删改查	16
2. 技术概览	4	3.3.2 知识上传	18
2.1 后端	5	3.3.3 其余页面	19
2.1.1 概述	5	3.4 模型封装	20
2.1.2 FastAPI 优势	6	3.5 兼容 Openai API	21
2.1.3 RAG 原理	7	3.6 知识库 RAG 搜索	23
2.2 数据库	10		
2.3 前端	12		
3. 效果展示	13		
3.1 前台	14		
3.2 使用其他前台程序	15		
3.3 后台	16		

1. 团队成员

- 胡瑞康:整体项目结构设计,知识库 RAG 搜索,模仿 OpenAI API 实现,前端修改
- 莫子昊,陈俊帆: 表格增删改查实现,后台测试,知识上传功能实现

2. 技术概览

2.1 后端

2.1.1 概述

- 基于 Python fastapi 开发,使用非阻塞式协程
- 用 psycopg3 协程链接数据库
- 封装统一 Model 接口对接数据表,减少重复 sql 撰写
- 通过 AsyncOpenAI 库链接兼容 OpenAI 接口的国产大模型平台 DeepSeek,智谱
- 根据 OpenAI 接口规范,模拟出了他们的 chat 接口,方便任何支持对接 OpenAI 的项目使用我们 API

2.1 后端

2.1.2 FastAPI 优势

对比来源：<https://www.techempower.com/>

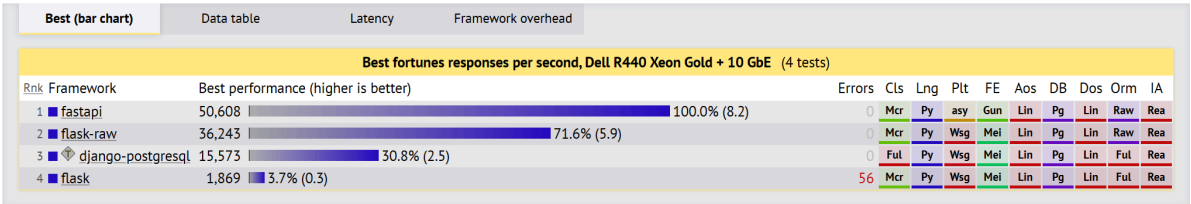


Figure 1: 22 年结果

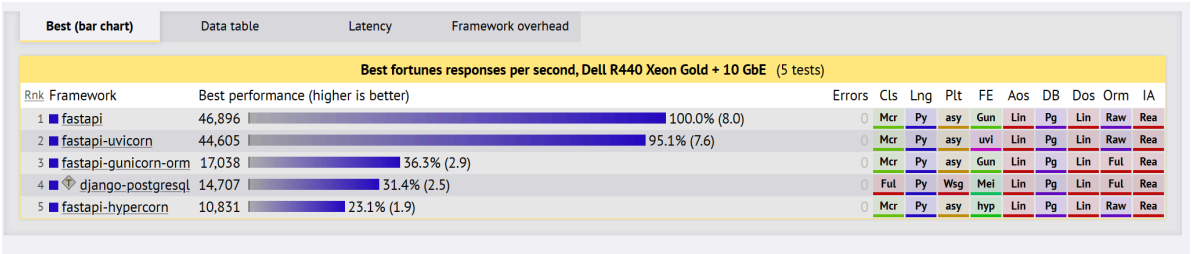


Figure 2: 23 年结果

2.1 后端

2.1.3 RAG 原理

生成式语言模型（如 GPT、BART 等）在多种文本生成任务中表现卓越，尤其在语言生成和上下文理解方面。然而，纯生成模型在处理事实类任务时存在一些固有的局限性。

例如，由于这些模型依赖于固定的预训练数据，它们在回答需要最新或实时信息的问题时，可能会出现“编造”信息的现象，导致生成结果不准确或缺乏事实依据。

2.1 后端

检索模型（Retriever）能够通过海量文档中快速找到相关信息，解决事实查询的问题。然而，传统检索模型（如 BM25）在面对模糊查询或跨域问题时，往往只能返回孤立的结果，无法生成连贯的自然语言回答。

检索增强生成（Retrieval-Augmented Generation, RAG）应运而生。RAG 通过结合生成模型和检索模型的优势，实时从外部知识库中获取相关信息，并将其融入生成任务中，确保生成的文本既具备上下文连贯性，又包含准确的知识。

2.1 后端

当用户提问时，先将用户问题在知识库中进行向量搜索，通过语义相似度匹配的方式查询到相关的内容，然后再将用户问题和搜索到的相关知识提供给大模型，使得大模型获得足够完备的知识来回答问题，以此获得更可靠的问答结果。

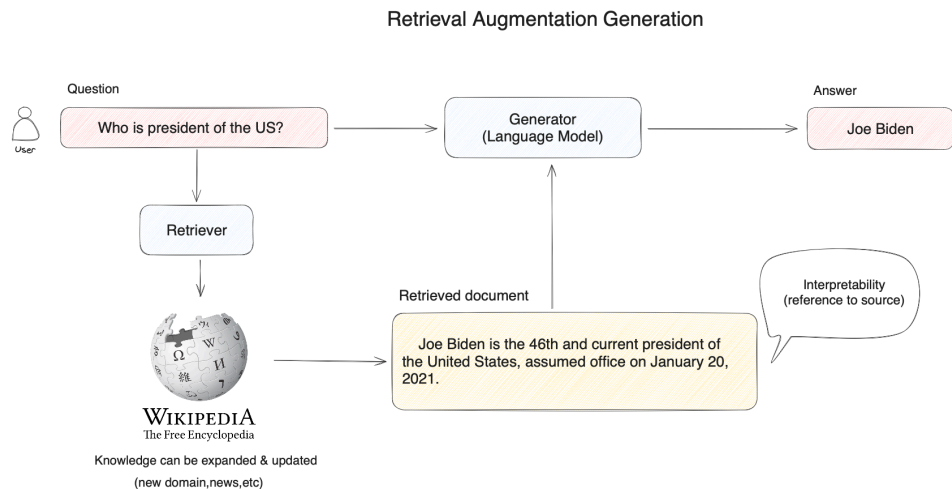


Figure 3: RAG

2.2 数据库

使用安装了 DataVec 向量引擎插件的 openGauss 数据库

DataVec 是一个基于 openGauss 的向量引擎，目前支持的向量功能有：精确和近似的最近邻搜索、L2 距离 & 余弦距离 & 内积、向量索引、向量操作函数和操作符。作为 openGauss 的内核特性，DataVec 使用熟悉的 SQL 语法操作向量，简化了用户使用向量数据库的过程。

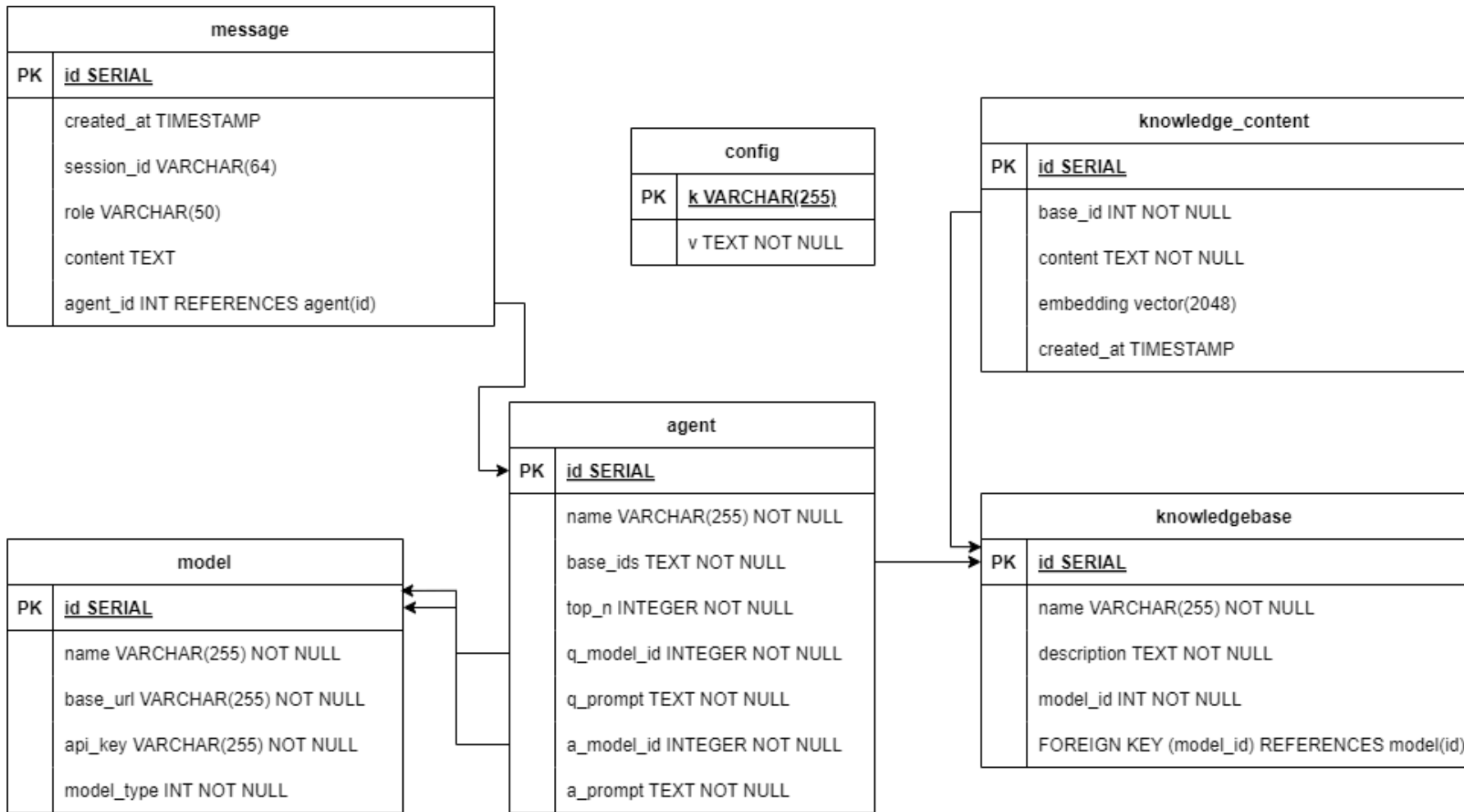


Figure 4: 数据库 ER 图

2.3 前端

- 后台使用 LayuiAdmin 配合 Jinja2 模板引擎
- 前台使用 Github 上的 chatgpt-web 做了一定修改对接后端 API, 传递对话 id

3. 效果展示

3.1 前台

各位同学可以使用校园网访问 <https://db.dorm.skyw.cc> 预览网页

目前导入了网络中心的一些 QA 做知识库,可以问该类型问题测试,比如校园网,电子邮箱等问题.



Figure 5: 二维码

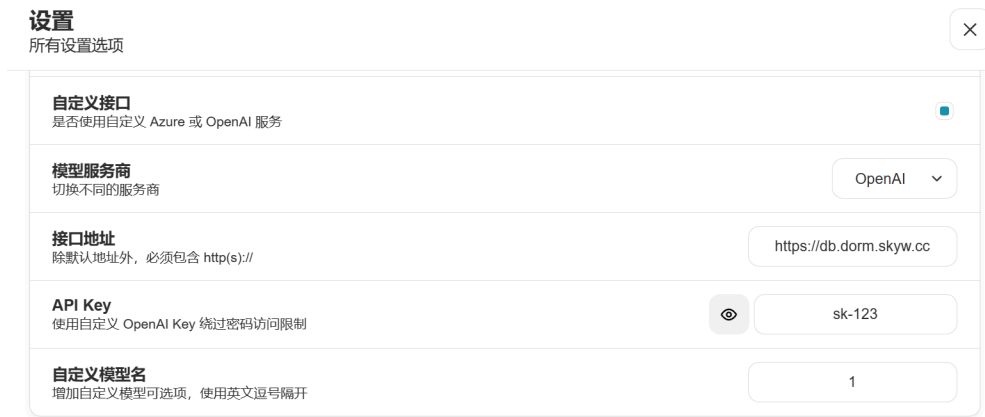


Figure 6: 前台

3.2 使用其他前台程序

由于我们支持 openai 接口，所以可以使用任何支持 openai 接口的前台程序，比如 nextchat。

可以访问 `nextweb.sysumsc.cn` 体验，输入 `db.dorm.skyw.cc` 作为 API 地址, `sk-123` 为密钥，即可使用



The image shows a settings window titled "设置" (Settings) with a subtitle "所有设置选项" (All settings options). It contains several configuration fields:

- 自定义接口** (Custom Interface): A toggle switch labeled "是否使用自定义 Azure 或 OpenAI 服务" (Whether to use custom Azure or OpenAI services), which is currently turned on.
- 模型服务商** (Model Provider): A dropdown menu labeled "切换不同的服务商" (Switch to different providers), currently set to "OpenAI".
- 接口地址** (Interface Address): A text input field labeled "除默认地址外，必须包含 http(s)://" (Besides the default address, it must contain http(s)://"), with the value "https://db.dorm.skyw.cc".
- API Key**: A text input field labeled "使用自定义 OpenAI Key 绕过密码访问限制" (Use custom OpenAI Key to bypass password access restrictions), with the value "sk-123". There is an eye icon to toggle visibility.
- 自定义模型名** (Custom Model Name): A text input field labeled "增加自定义模型可选项，使用英文逗号隔开" (Add custom model options, use English commas to separate), with the value "1".

Figure 7: nextchat

3.3 后台

3.3.1 表格增删改查

表格增删改查，在业界一般称为 CRUD 操作，是指对数据库中的数据进行增加、查询、修改和删除操作。

借助 LayuiAdmin，实现了搜索，分页，添加，修改，删除，批量删除等功能。

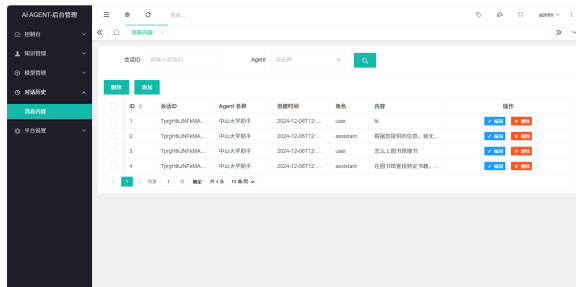


Figure 8: 消息内容

3.3 后台

3. 效果展示

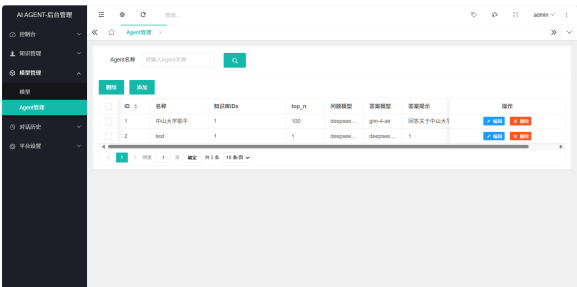


Figure 9: 智能体管理

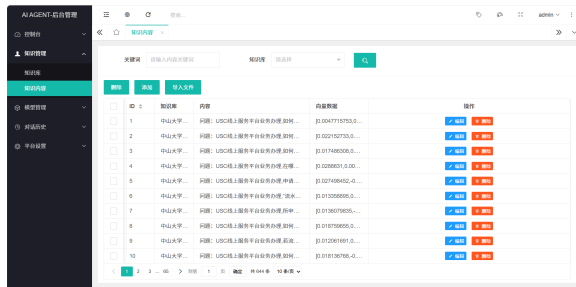


Figure 11: 知识内容

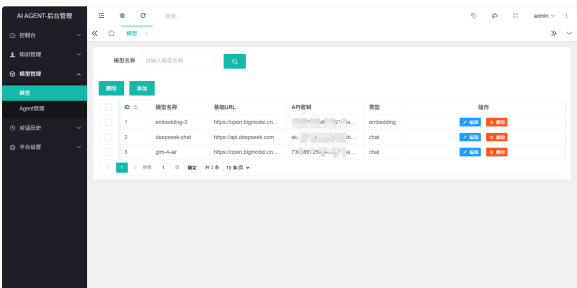


Figure 10: 模型管理

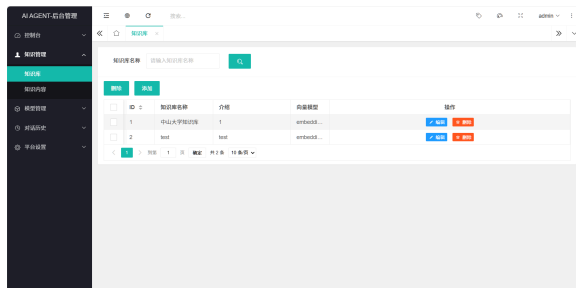


Figure 12: 知识库

3.3.2 知识上传

知识上传导入功能主要包括文件上传、文件导入任务的提交、导入任务的异步处理以及文件内容的拆分和保存。

管理员可以在后台将外部文件中的知识内容导入到系统中，并自动生成嵌入向量，以便后续的知识检索和应用

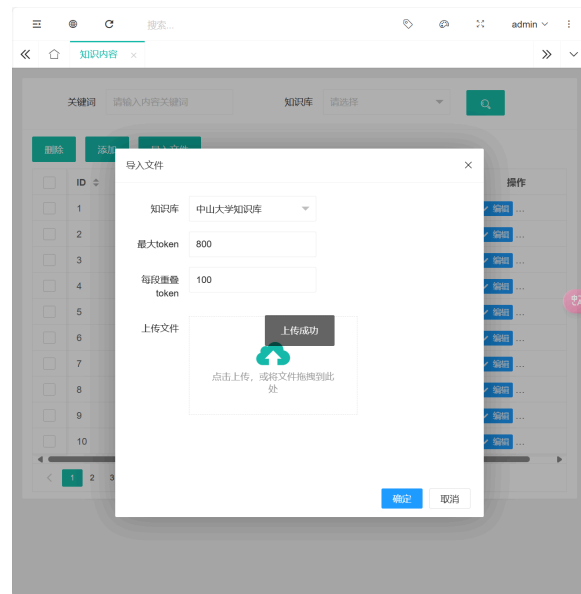


Figure 13: 知识库

3.3 后台

3. 效果展示

3.3.3 其余页面

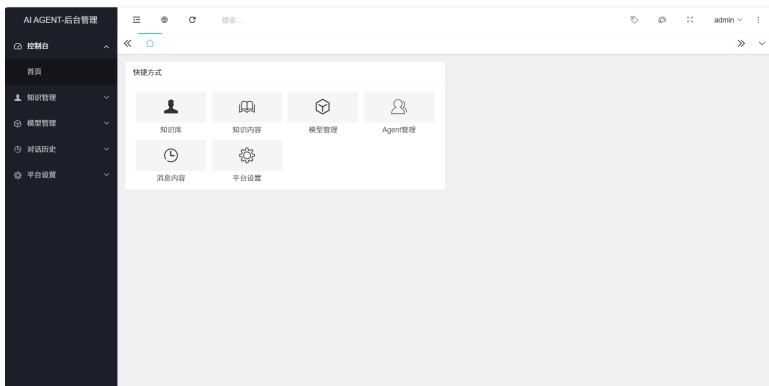


Figure 14: 后台首页

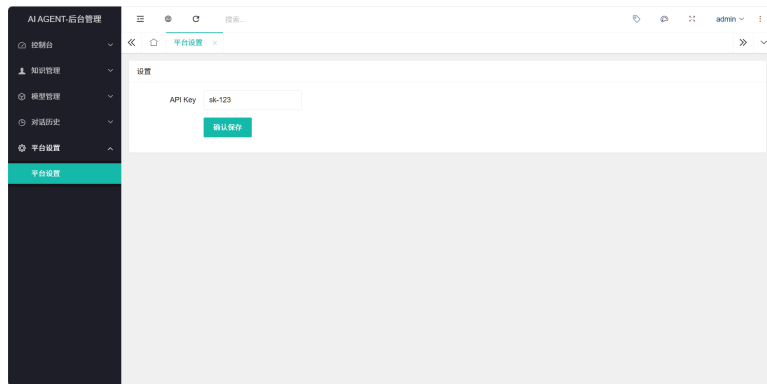


Figure 15: 平台设置

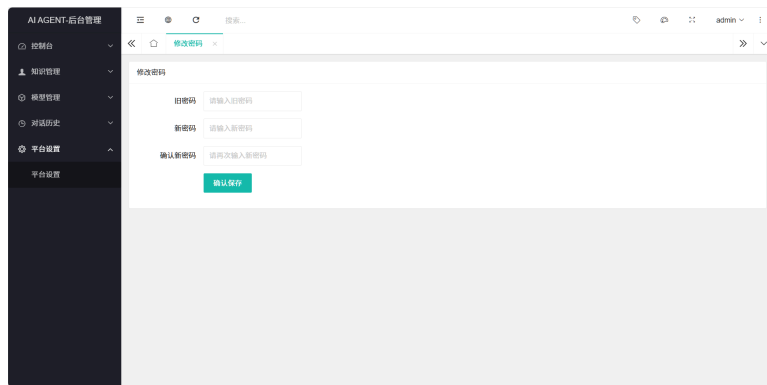


Figure 16: 密码设置

3.4 模型封装

为了方便后台增删改查的实现，创建了 BaseModel，封装了数据库的增删改查操作。

每个表的模型类继承自

BaseModel 类，可以直接调用父类的方法，减少了重复代码的编写。

```
class BaseModel:
    table_name: str # 子类需要定义表名

    解释代码 | 注释代码 | 生成单测 | ×
    def __init__(self):
        self.db_schema = settings.db_schema

    解释代码 | 注释代码 | 生成单测 | ×
    async def save(self, data: dict):
        """保存数据到表中"""
        async with db.pool.connection() as conn:
            async with conn.cursor(row_factory=psycopg.rows.dict_row) as cur:
                keys = ', '.join(data.keys())
                values = ', '.join([f"%({k})s" for k in data.keys()])
                sql = f'INSERT INTO "{self.db_schema}"."{self.table_name}" ({keys}) VALUES ({values})'
                await cur.execute(sql, data)
                result = await cur.fetchone()
                return dict(result) if result else None

    解释代码 | 注释代码 | 生成单测 | ×
    async def update(self, id: int, data: dict):
        """更新表中的数据 (参数化查询版)"""
        set_clause = ', '.join([f'"{k}" = %({k})s' for k in data.keys()])
        sql = f'UPDATE "{self.db_schema}"."{self.table_name}" SET {set_clause} WHERE id = {id}'
        data['id'] = id
        async with db.pool.connection() as conn:
            async with conn.cursor() as cur:
                await cur.execute(sql, data)
            return True

    解释代码 | 注释代码 | 生成单测 | ×
    async def delete(self, id: int):
        """删除表中的数据"""
        sql = f'DELETE FROM "{self.db_schema}"."{self.table_name}" WHERE id = {id}'
        async with db.pool.connection() as conn:
            async with conn.cursor() as cur:
                await cur.execute(sql, (id,))
            return True
```

Figure 17: 基类模型

3.5 兼容 Openai API

兼容 Openai 格式可以方便被各种第三方应用调用，第三方开发者无需对已有功能做修改

3. 效果展示

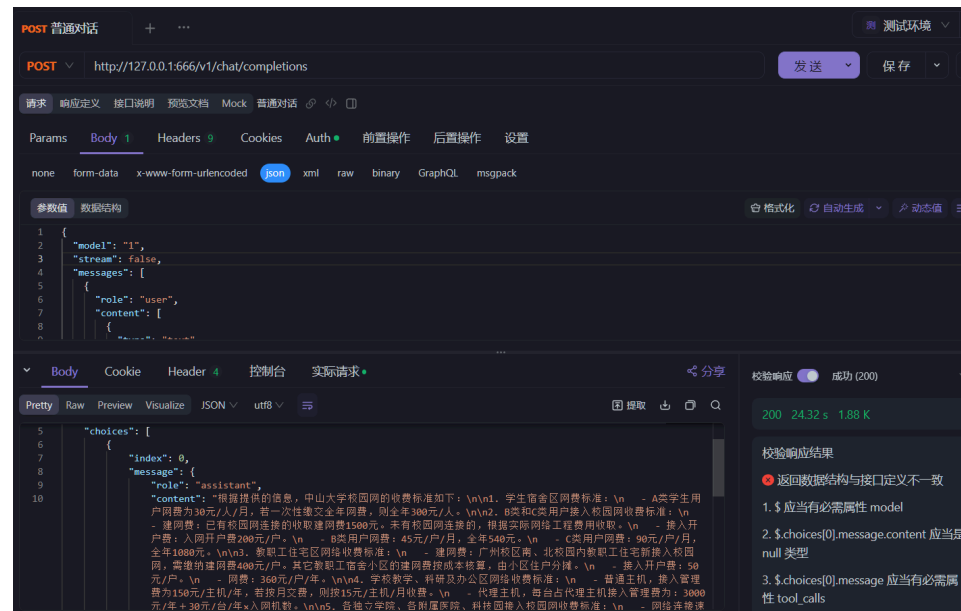


Figure 18: API 回答

3.5 兼容 Openai API

3. 效果展示

SSE (Server-Sent Events) 是一种 Web 技术，它允许服务器实时向客户端推送数据。

相比于传统的轮询和长轮询机制，SSE 提供了一种更高效且实时的数据推送方式。

用于给前端实现 API 对话。

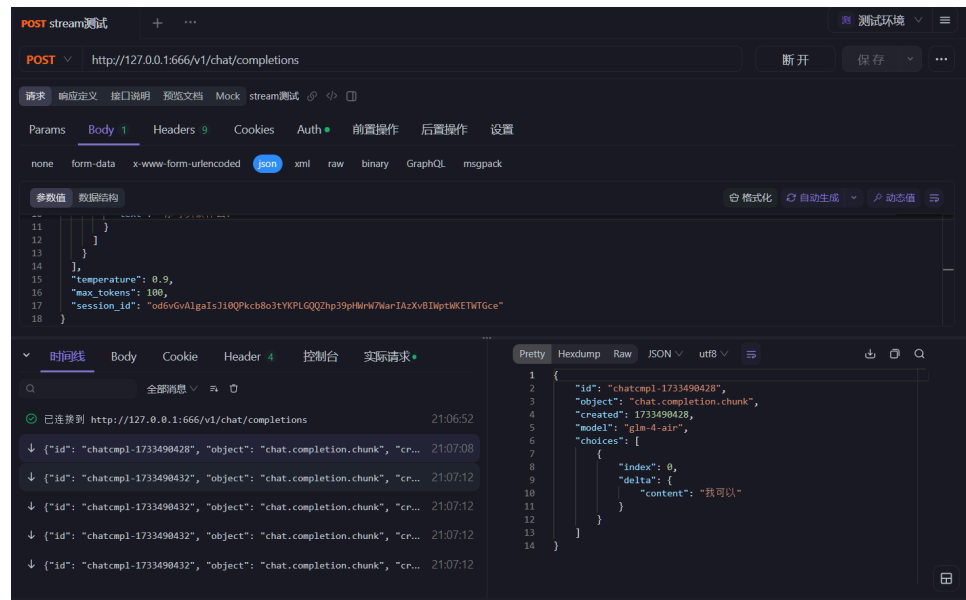


Figure 19: API-SSE 支持

3.6 知识库 RAG 搜索

用户问题会先被大模型扩展为更多问题，之后调用接口 Embedding，然后把向量在数据库中匹配相似的 TopN 个结果

由于 FastAPI 特性，链接数据库，调用大模型接口均使用协程 使得我们的搜索速度非常快

```
2024-12-06 21:12:00,991 - INFO - json_str:{
  "questions": [
    {
      "question": "校园网的费用"
    },
    {
      "question": "校园网的价格"
    },
    {
      "question": "校园网的收费标准"
    }
  ]
}
2024-12-06 21:12:00,992 - INFO - 问题优化耗时: 4.106912851333618秒
2024-12-06 21:12:02,130 - INFO - HTTP Request: POST https://open.bigmodel.cn/api/paas/v4/embeddings "HTTP/1.1 200 OK"
2024-12-06 21:12:02,301 - INFO - HTTP Request: POST https://open.bigmodel.cn/api/paas/v4/embeddings "HTTP/1.1 200 OK"
2024-12-06 21:12:02,476 - INFO - HTTP Request: POST https://open.bigmodel.cn/api/paas/v4/embeddings "HTTP/1.1 200 OK"
2024-12-06 21:12:02,542 - INFO - 向量搜索耗时: 1.5384314060211182秒
```

Figure 20: RAG 搜索