

Chatbot -	2
	4
API	7
ERD	9
[CHAT-64]	10
Flowchart	12
가	14



Chatbot - 제품 요구 사항

배포예정일	미정					
큰틀	Type	Key	요약	담당자	우 선...	상태
		CHAT-22	LLM to Chatbot	할당되지 않음		해야 할 일
	1 항목					방금 동기화
문서 상태	임시본					
문서 소유자	@hyein.jo					
설계자	@ 디자이너					
기술 책임자	@ 리더					
테크니컬 라이터	@ 작성자					
QA						

🎯 목표 및 성공 지표 🔗

목표	지표
요구사항 확립	DB 설계 문서 작성
DB 설계 및 구축	
API 설계 및 구현	
[BE] Python 으로 변경	

📋 요구사항 🔗

요구 사항	중요도	비고
여러 프로젝트에서 서비스를 사용할 수 있다.	보통	
각 프로젝트에는 다수의 사용자가 존재한다.	보통	
서비스는 프로젝트를 관리하되 사용자를 관리하지 않는다.	보통	
사용자는 해당 사용자의 자료를 식별하는 용도로만 사용된다.	낮음	
각 사용자는 서비스에 질문을 보내며 하나의 질문 당 하나의 대답을 리	높음	

던한다.		
대답은 비슷한 질문에 대한 답을 DB에서 불러오거나 LLM이 생성한다.	높음	
모든 질문과 대답은 저장된다.	높음	
각 사용자는 자신이 한 이전 질문을 확인하고 비활성화 할 수 있다.	높음	
각 프로젝트 당 하나 혹은 다수의 관리자가 존재한다.	낮음	
관리자는 담당 프로젝트의 전체 질문과 대답을 확인할 수 있지만 각 질문을 한 사용자는 식별할 수 없다.	보통	
관리자는 전체 질문의 갯수와 그 비중 그리고 자주 사용되는 질문을 확인할 수 있다.	보통	
관리자는 사용자의 질문으로 생성된 대답을 올바르게 수정할 수 있다.	보통	

⚠ 범위를 벗어남 ↻

요구사항 우선 확립

- DB 설계 및 구축
- API 설계 및 구현
- [BE] Python 으로 변경

환경구성

TO-BE: 기존 Access Token 방식 로그인에서 API KEY 로그인 방식으로 교체 예정

개발 Tool

Front-End (TYS Standard Framework) <https://dev-git.tongyang.co.kr/chatbot/chatbot.fe> 

구분	SW	버전	비고
IDE	VSCode	VSCodeUserSetup-x64-1.80.1	
JavaScript	nodejs	v18.xx.xx	
패키지매니저	pnpm	8.5.1	
Framework	React		
개발 환경	Vite		
형상관리	git	Git-2.41.0.3-64-bit	
웹 브라우저	Chrome	ChromeStandaloneSetup64	

참고 : nodejs의 버전에 따라 제공되는 내장객체 , 내장함수는 버전마다 상의

맨 앞의 18은 메이저 버전이라는 의미로 이 부분의 버전만 맞으면 이후 생기는 버전에 대해서는 호환 가능

<https://dev-git.tongyang.co.kr/standard/std.guide/-/blob/main/1.표준개발환경구성/1.로컬 개발환경 구성/프론트엔드 개발환경 설정가이드.md>

Back-End <https://dev-git.tongyang.co.kr/chatbot/chatbot.be> 

Java (TYS Standard Framework) 

구분	SW	버전	비고
IDE	Eclipse (STS) 4.x.x	4.x.x	spring-tool-suite-4.x.x 버전 설치 (IntelliJ도 사용 가능)
JDK	OpenJDK	11.x	
패키지매니저	mvn		
Framework	Spring Boot	2.7.10	
ORM	mybatis JPA		
형상관리	EGit(Eclipse)	최신	Git integration for Eclipse (Eclipse 내장 Plug-in 사용)
Git-scm(GUI/Bash)	git	최신	Git-2.29.0-64-bit.exe

redis			
-------	--	--	--

<https://dev-git.tongyang.co.kr/standard/std.guide/-/blob/main/1.표준개발환경구성/1.로컬 개발환경 구성/백엔드 개발환경 구성.md>

Python

구분	SW	버전	비고
IDE	VSCode	VSCodeUserSetup-x64-1.80.1	
Python	Python11	11.x	
패키지매니저	pip	최신	
Framework	FastAPI	최신	
ORM	sqlalchemy	2.0	
개발환경	pipenv	최신	<code>pip install pipenv</code>
형상관리	git	Git-2.41.0.3-64-bit	
redis			

환경변수 설정

파이썬 설치 경로 환경변수 추가

고급시스템설정 > 환경변수 추가 > Path > 새로만들기 설치경로 주소입력

ex) C:\Users\TY\AppData\Local\Programs\Python\Python311\Scripts

아래 내용은 가상환경을 설정할 폴더(코드 위치)로 이동한 후 진행합니다.

가상환경 생성

```
pipenv --python 11
```

가상환경 활성화

```
pipenv shell
```

파이썬 버전으로 인한 호환 문제가 생길 시 Pipfile 파일 제일 아래의 하위 버전 내용을 삭제합니다.

```
1 # Pipfile
2
3 [requires]
4 python_version = "3.11"
5 # python_full_version = "3.11.4" # 주석처리
```

정상적으로 접속 성공시 커맨드 창에서 (가상환경명) 주소> 로 표시 됩니다.


가상환경 내에서 필요 패키지 설치를 진행합니다.

```
pip install -r requirements.txt
```

아래 명령어를 이용해 fastapi 서버를 실행합니다.

```
uvicorn app.main:app --host 호스트 --port PORT
```

가상환경 종료 `exit`

 [FastAPI](#)  [SQLAlchemy](#)  [pipenv](#)

DataBase [↗](#)

구분	SW	버전	비고
	Postgresql	최신	<code>CREATE EXTENSION vector;</code>
	DBeaver	23.2.3	

API 설계

요구 사항	중요도	비고
각 사용자는 서비스에 질문을 보내며 하나의 질문 당 하나의 대답을 리턴한다.	높음	단건 조회/추가
대답은 비슷한 질문에 대한 답을 DB에서 불러오거나 LLM이 생성한다.	높음	단건 조회/추가
모든 질문과 대답은 저장된다.	높음	단건 추가
각 사용자는 자신이 한 이전 질문을 확인하고 비활성화 할 수 있다.	높음	단건 수정
관리자는 담당 프로젝트의 전체 질문과 대답을 확인할 수 있지만 각 질문을 한 사용자는 식별할 수 없다.	보통	다건 조회
관리자는 전체 질문의 갯수와 그 비중 그리고 자주 사용되는 질문을 확인할 수 있다.	보통	다건 조회
관리자는 사용자의 질문으로 생성된 대답을 올바르게 수정할 수 있다.	보통	단건 수정

- 질문 단건 조회/추가/수정
 - 조회
 - 사용자 질문 DB 조회
 - 사용자 이전 질문 및 답변 조회
 - 관리자 질문 수정을 위한 조회
 - 추가
 - 사용자 질문 및 답변 추가 → 내부로직
 - 수정
 - 사용자 질문 비활성화
 - 관리자 답변 수정
- 질문 리스트 조회
 - 프로젝트별 → 사용자 정보 X
 - 프로젝트 전체 질문 유형 별 조회(개수) → 리스트 비동기 로딩은?
- 프로젝트+사용자 별
 - 이전 질문 다건 조회

기능	Request	HTTP	URI
사용자 질문 중복 여부 확인 및 대답 조회	question, user, project	POST	/chat
이전 질문 중 질문 선택 조회 (사용자/관리자)	uuid	GET	/chat/{uuid}
이전 질문 목록 중 질문 선택 비활성화	uuid, (active)	PUT	/chat/disable
프로젝트 질문 중 답변 수정	uuid, answer	PUT	/chat/update

프로젝트 유형별 전체 질문 및 갯수 조회	project, type	GET	/chat/list/
프로젝트 전체 LLM 질문 및 갯수 조회	project, type	GET/ POST	/project/llm
사용자 별 질문 전체 조회	user, project	POST	/chat/list

- 관리자 페이지

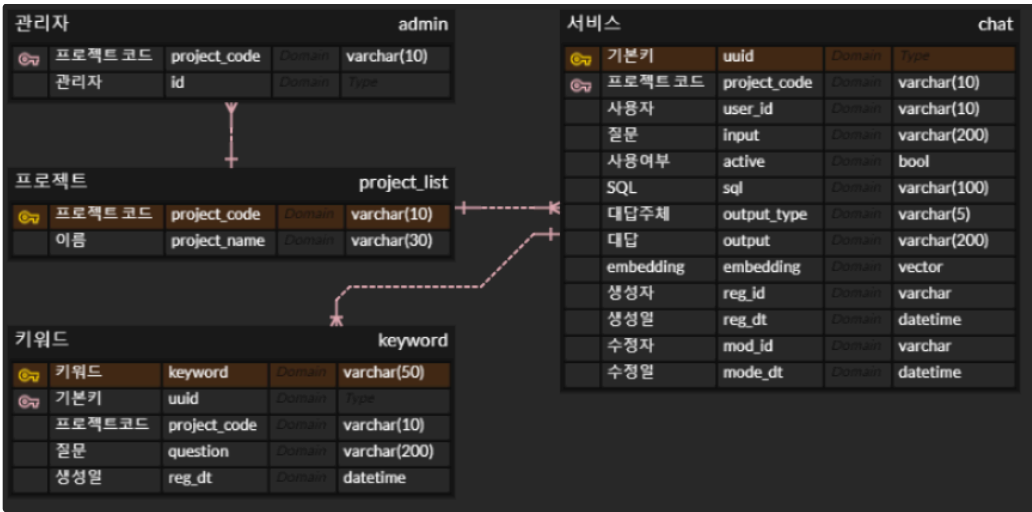
회원가입,

로그인, 관리자 등록

관리자 여부 확인,

API KEY 발급

ERD

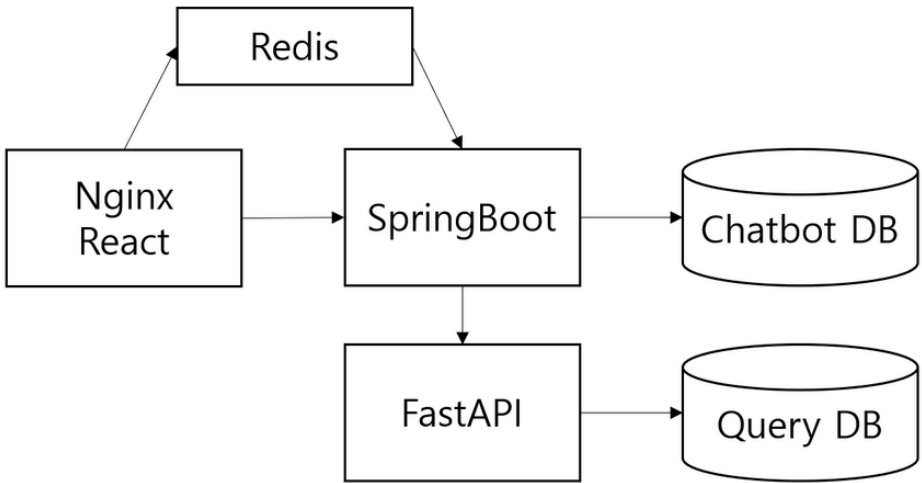


복합 index: project_code, user_id

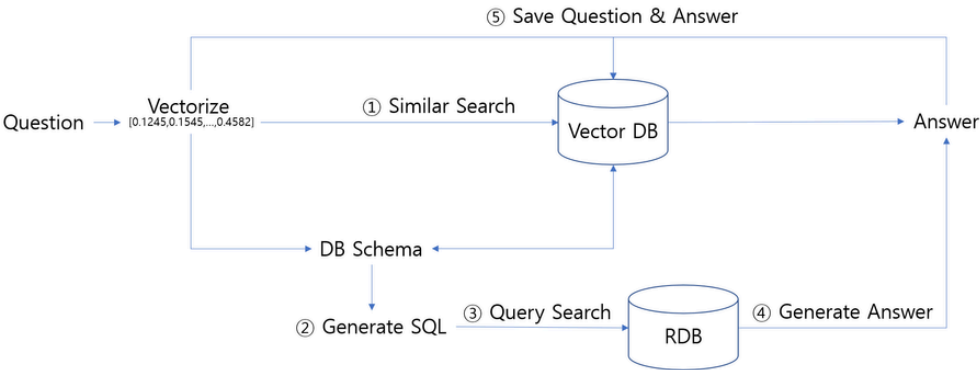
→ 빈번한 수정

[CHAT-64] 구조 확정

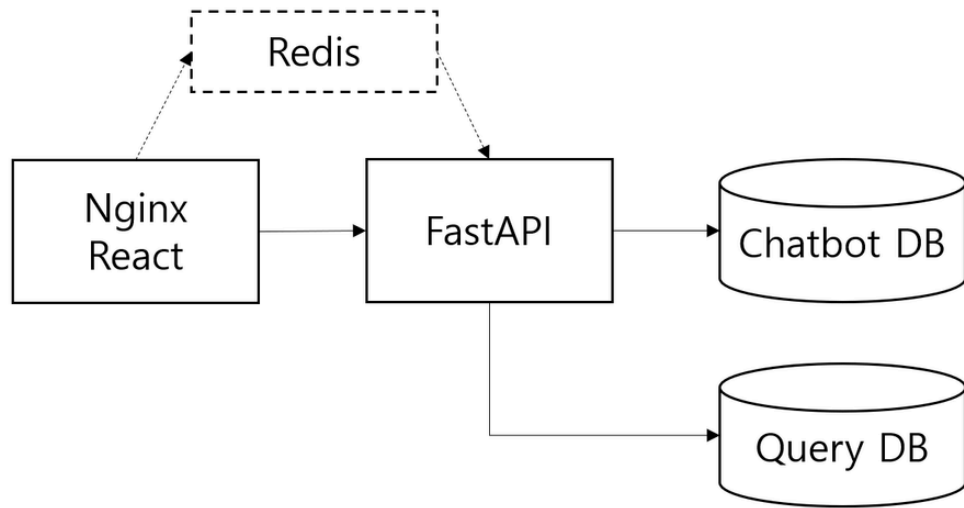
현재 (24.03.14) 구조



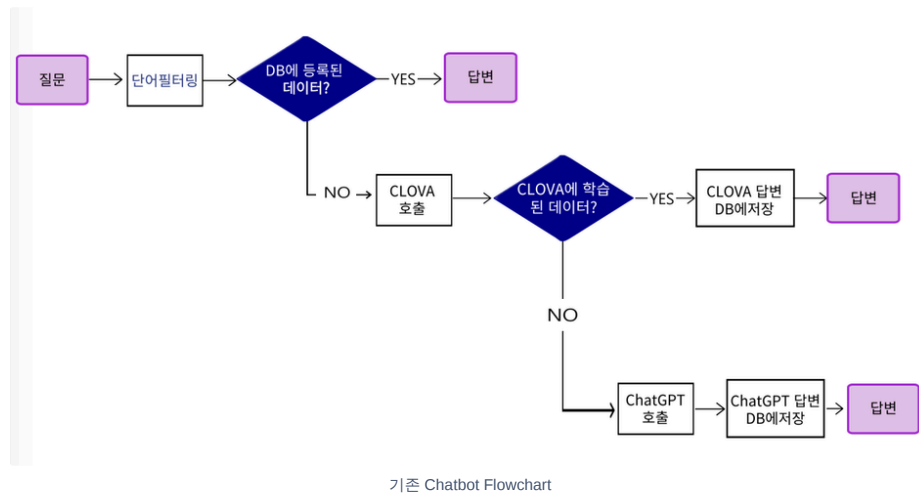
QnA Logic (tobe)



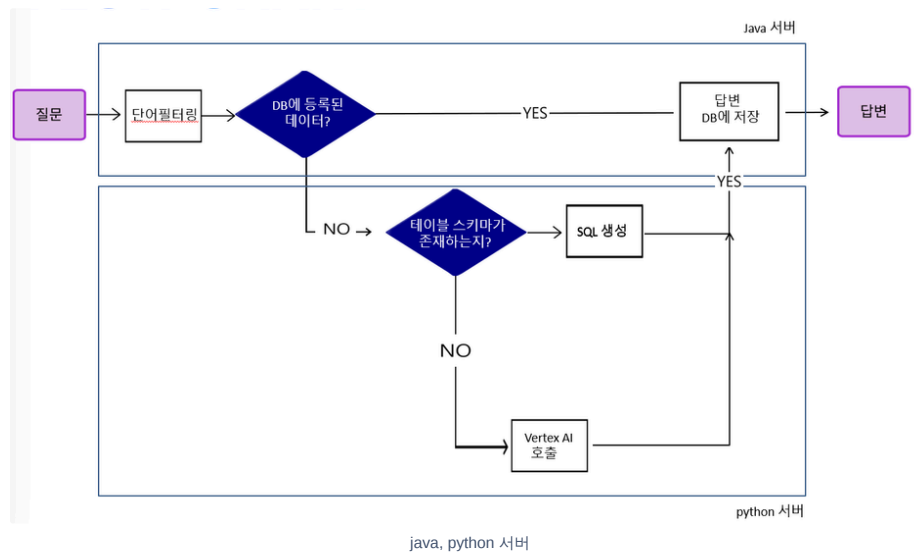
구조 (tobe)

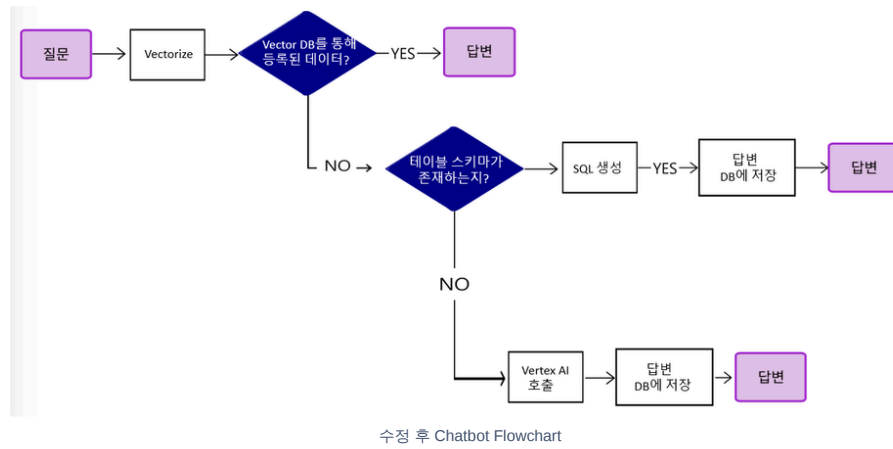


Flowchart



- 1. 질문이 들어오면 KOMORAN 라이브러리를 통해 명사들을 골라내는 필터링을 한 후, 그 명사들이 포함된 질문 데이터가 DB에 있다면 해당 답변을 출력한다.
- 2. 만약 DB에서 질문에 해당하는 답변 데이터가 없다면, CLOVA 챗봇 API를 호출하여 해당 질문에 해당하는 학습된 데이터를 답변으로 사용한다.
- 3. CLOVA에서도 나온 답변이 없다면, ChatGPT API를 호출하여 해당 질문에 해당하는 답변이 출력된다.





1. 질문이 들어오면 Embedding 모델을 이용해서 벡터로 만들어준다.
2. Vector DB를 통해 벡터를 가지고 텍스트 간의 거리 비교를 통해 유사한 질문을 찾아 해당 답변을 출력한다.
3. 만약 DB에서 질문에 해당하는 답변이 없다면, 사용자 질문과 관련된 데이터 테이블이 있는지 확인한다. 있다면, SQL문을 생성하여 해당 테이블에서 답변을 찾아 출력한다.
4. 질문과 관련된 테이블이 없다면, Vertex AI를 호출하여 해당 질문에 해당하는 답변이 출력된다.

추가 필요 작업

현재 Embedding 모델과 LLM은 LangChain으로 Wrapping된 Google VertexAI에서 제공하는 모델을 사용 중

Embedding 모델의 경우 영문 Embedding으로 한글 질문은 Google의 번역 API 로 영문으로 변환하여 임베딩 진행함 (영어질문만 하는 경우 번역기능 필요 없음)

모델과 번역은 다른 API로 교체 가능

GCP 프로젝트 생성 및 API 활성화 (Architecture팀 요청)

- Cloud Translation API
- Vertex AI API

프로젝트 service key 발급

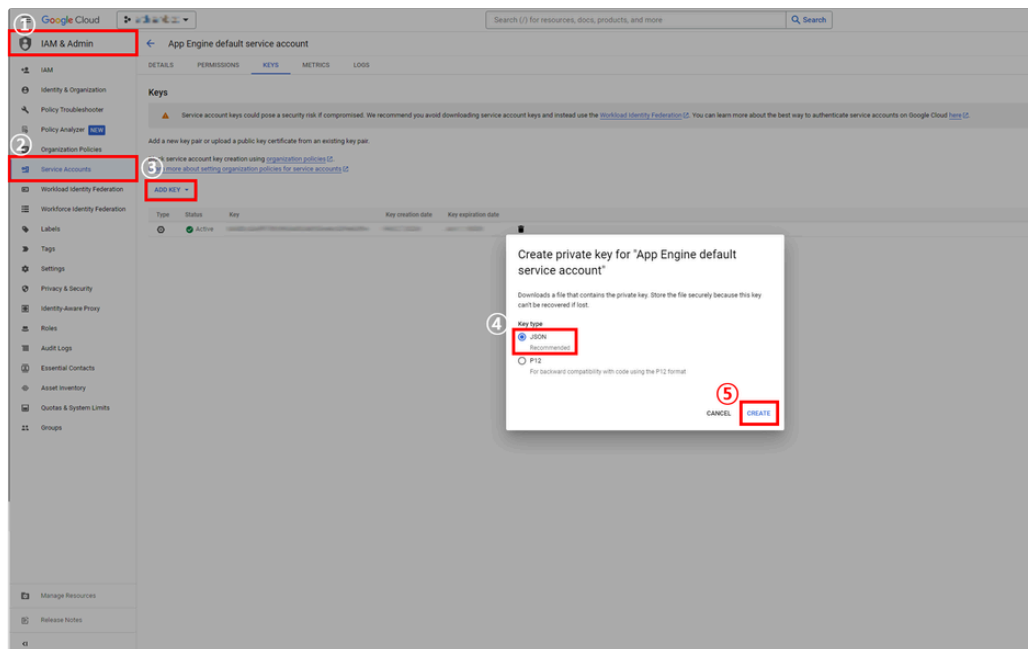


Table 정보 추가 방법

ex) Table 정보 텍스트 파일

```
1 /*
2 -- 테이블의 자세한 사용 용도
3 Table's detail information
4 -- 약어가 존재할 경우 각 약어의 fullname
5 some columns use abbreviation, each abbreviation's meaning follows:
6 pw = password
7 nm = name
8 desc = description
9 reg = regist
10 dt = date
11 mod = modify
12 */
```

```

13
14 -- Table Create 문
15 CREATE TABLE member (
16     user_id VARCHAR NOT NULL,
17     user_pw VARCHAR NOT NULL,
18     user_nm VARCHAR NOT NULL,
19     user_email VARCHAR NOT NULL,
20     user_desc VARCHAR,
21     live_gb VARCHAR,
22     lang VARCHAR,
23     use_yn VARCHAR DEFAULT 'Y'::character varying,
24     reg_id VARCHAR,
25     reg_dt TIMESTAMP WITHOUT TIME ZONE,
26     mod_id VARCHAR,
27     mod_dt TIMESTAMP WITHOUT TIME ZONE,
28     age INTEGER,
29     ip_address VARCHAR(20),
30     salary numeric,
31     CONSTRAINT member_pkey PRIMARY KEY (user_id)
32 )
33
34 /*
35 -- 실제 데이터가 아니어도 됩니다.
36 3 rows from member table:
37 user_id user_pw user_nm user_email user_desc live_gb lang use_yn reg_id reg_dt mod_id mod_dt age ip_a
38 rmattaus0 $2a$04$Tiplhx00WwzRKBjyeCpDC0kAczh1W/VKY1set4ZRwEcdLR6uL00uq Ronda rgracewood0@bbb.org None
39 gpochin1 $2a$04$1kze01YpziSwai/ZH9KCn.mZVKDowN.rC5biQDT8G.Tc2MMJch/hm Garey gbouts1@over-blog.com None
40 cfont2 $2a$04$Y0xFOXyGP8dp.gg7kIDJRuqLuv7THiUxaT/xJ5jk2VpVgYSmIkd1y Cointon coscannill2@google.com.hk None
41 */

```

Table 검색정보 저장 (LangChain 이용)

- DB 연결 및 DB 데이터를 이용한 Table 정보 텍스트 파일 생성

```

1 import glob
2 from langchain import SQLDatabase
3 from langchain_community.document_loaders import TextLoader
4 from langchain_community.vectorstores import PGVector
5 import re
6
7 # db에서 Table 정보 가져와서 텍스트 파일로 저장
8 # 직접 작성해도 됨
9
10 db = SQLDatabase.from_uri(db_uri) # DB 정보
11 table_info = db.get_table_info()
12
13 path = './tables'
14 tables=table_info.split('*/')
15 table_name = []
16
17 for table in tables:
18     if len(table) == 0:
19         tables.remove(table)
20     else:
21         p = re.compile('\s+')
22         table = re.sub(p, ' ', table)
23         p = re.compile(r'CREATE TABLE (.*) [(]\s', re.DOTALL)
24         result = p.findall(table)
25         table_name.append(result[0])

```

```

26 for i in range(len(tables)):
27     tables[i] += '*/'
28
29 # 파일 쓰기 모드로 열기
30 for i in range(len(table_name)):
31     with open(path+'/'+table_name[i]+'.txt', "w+", encoding='utf-8') as file:
32         file.write(tables[i])

```

- Table 정보 텍스트 파일 VectorDB 저장을 위한 Format 변경

```

1 path = '파일저장위치/*.txt'
2 file_list = glob.glob(path) ## 폴더 안에 있는 모든 파일 출력
3
4 tables=[]
5 for i in range(0, len(file_list)):
6     text = TextLoader(file_list[i]).load()
7     tables.extend(text)
8     os.remove(file_list[i]) # 확인 필요

```

- Vector Store 연결

```

1 # 문서로 vector store 생성(처음만 실행)
2 vector_store = PGVector.from_documents(
3     embedding=embeddings,
4     documents=tables,
5     connection_string=db_file,
6 )

```

```

1 vector_store = PGVector.from_existing_index(
2     embeddings,
3     collection_name = 'chatbot', # ProjectCode 사용
4     connection_string=db_file)

```

- Table 정보 추가

```

1 # 문서 추가
2 file_list = glob.glob(path) ## 폴더 안에 있는 모든 파일
3
4 tables=[]
5 for i in range(0, len(file_list)):
6     text = TextLoader(file_list[i]).load()
7     tables.extend(text)
8     os.remove(file_list[i]) # 확인 필요
9
10 vector_store.add_documents(documents=tables,
11                             embedding=embeddings,)

```