# Documentation : Intégration d’un Chatbot RAG dans un Wordpress local

Yohan Ruffieux, 09.07.2024

## Mise en place du chatbot

Pour mettre en place le chatbot, j’ai simplement suivi le tutoriel suivant :

<https://realpython.com/build-llm-rag-chatbot-with-langchain/#demo-an-llm-rag-chatbot-with-langchain-and-neo4j>

J’ai dû adapté un tout petit peu le fichier chatbot\_api\src\main.py pour que ne pas me faire bloquer par le CORS lorsque je faisais des requêtes depuis mon site Wordpress :

from fastapi import FastAPI, HTTPException  
from fastapi.middleware.cors import CORSMiddleware  
from models.hospital\_rag\_query import HospitalQueryInput, HospitalQueryOutput  
from utils.async\_utils import async\_retry  
from agents.hospital\_rag\_agent import hospital\_rag\_agent\_executor  
  
app = FastAPI(  
 title="Hospital Chatbot",  
 description="Endpoints for a hospital system graph RAG chatbot",  
)  
  
origins = [  
 "http://localhost",  
 "http://localhost:8000",  
 "http://localhost/chatbot/mon-chatbot-personnalise/",  
]  
  
app.add\_middleware(  
 CORSMiddleware,  
 allow\_origins=origins,  
 allow\_credentials=True,  
 allow\_methods=["GET", "POST"],  
 allow\_headers=["\*"],  
)  
  
  
@async\_retry(max\_retries=10, delay=1)  
async def invoke\_agent\_with\_retry(query: str):  
 *"""  
 Retry the agent if a tool fails to run. This can help when there  
 are intermittent connection issues to external APIs.  
 """* return await hospital\_rag\_agent\_executor.ainvoke({"input": query})  
  
  
@app.get("/")  
async def get\_status():  
 return {"status": "running"}  
  
  
@app.post("/hospital-rag-agent", response\_model=HospitalQueryOutput)  
async def query\_hospital\_agent(  
 query: HospitalQueryInput,  
) -> HospitalQueryOutput:  
 try:  
 query\_response = await invoke\_agent\_with\_retry(query.text)  
 query\_response["intermediate\_steps"] = [  
 str(s) for s in query\_response.get("intermediate\_steps", [])  
 ]  
 return query\_response  
 except Exception as e:  
 raise HTTPException(status\_code=500, detail=str(e))

## Création d’un Wordpress local avec XAMPP

J’ai suivi cette vidéo :

<https://elementor.com/academy/install-wordpress-xampp/?psafe_param=1&utm_source=google&utm_medium=cpc&utm_campaign=13060922353&utm_term=&lang=&gad_source=1&gclid=EAIaIQobChMIw_uzz8-ZhwMVCJSDBx09EgGPEAAYASAAEgJLwfD_BwE>

## Création d’un plugin Wordpress

J’ai installé l’extension « File Manager » pour pouvoir manipuler les fichiers de Wordpress :

Une image contenant texte, capture d’écran, Police, Système d’exploitation

Description générée automatiquement

Ensuite je suis allé dans le File Manager et j’ai créé des nouveaux dossiers et fichiers :

\wp-content\plugins\my-chatbot.php

\wp-content\plugins\assets\js\chatbot.js

\wp-content\plugins\assets\css\chatbot.css

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

J’ai téléchargé en local les fichiers et je les ai modifiés : <https://github.com/y-ruffieux/chatbot-wp/tree/main>

(code généré majoritairement avec chatgpt, peut donc sûrement être amélioré)

## Activer le plugin

Il suffit d’aller dans la liste d’extension et de cliquer sur « Activer »

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

## Insérer le plugin dans une page Wordpress

Il suffit de mettre entre crochet le nom du « composant » qu’on a mis dans « add\_shortcode » dans le php dans l’éditeur de page :

Une image contenant texte, Police, capture d’écran, blanc

Description générée automatiquement

## Résultat final

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement