UDC 004.9

VIRTUAL ASSISTANT OF THE COMPANY'S SUPPORT SERVICE BASED ON LOCAL LLM

KUROCHKA KONSTANTIN SERGEEVICH,
Senior Lecturer
ARISTOTE BOWAYE DILOMBA
Student
Sukhoi State Technical University of Gomel

Abstract: The object of development is a virtual assistant for the company's support service based on a local LLM.
The aim of the work is to create a virtual assistant for the company's support service based on a local LLM to improve customer service efficiency and reduce support costs. The main result of the work: a virtual assistant for the company's sup-port service based on a local LLM with user registration and authentication, a chat interface, a knowledge base, and integration with a ticket management system.
Keywords: virtual assistant, local LLM, support service, natural language processing, chatbot, customer support, TinyLlama.

ВИРТУАЛЬНЫЙ ПОМОЩНИК СЛУЖБЫ КОМПАНИИ НА БАЗЕ ЛОКАЛЬНОЙ LLM

Курочка Константин Сергеевич,
Аристот Бовайе Диломба

Аннотация: Объектом разработки является виртуальный помощник службы поддержки компании на базе локальной LLM.
Цель работы: создание виртуального помощника службы поддержки компании на базе локальной LLM для повышения эффективности обслуживания клиентов и снижения затрат на поддержку.
Основной результат работы: виртуальный помощник службы поддержки компании на базе локальной LLM с регистрацией и аутентификацией пользователей, чат-интерфейсом, базой знаний и интеграцией с системой управления заявками.
Ключевые слова: виртуальный помощник, локальный LLM, служба поддержки, обработка естественного языка, чат-бот, поддержка клиентов, TinyLlama.

The architecture of ASSIST-NEXUS is designed to be modular, scalable, and maintainable, following best practices in software engineering and system design. The system is built using a microservices architecture, with each component responsible for a specific aspect of the virtual assistant's functionality (Figure 1).

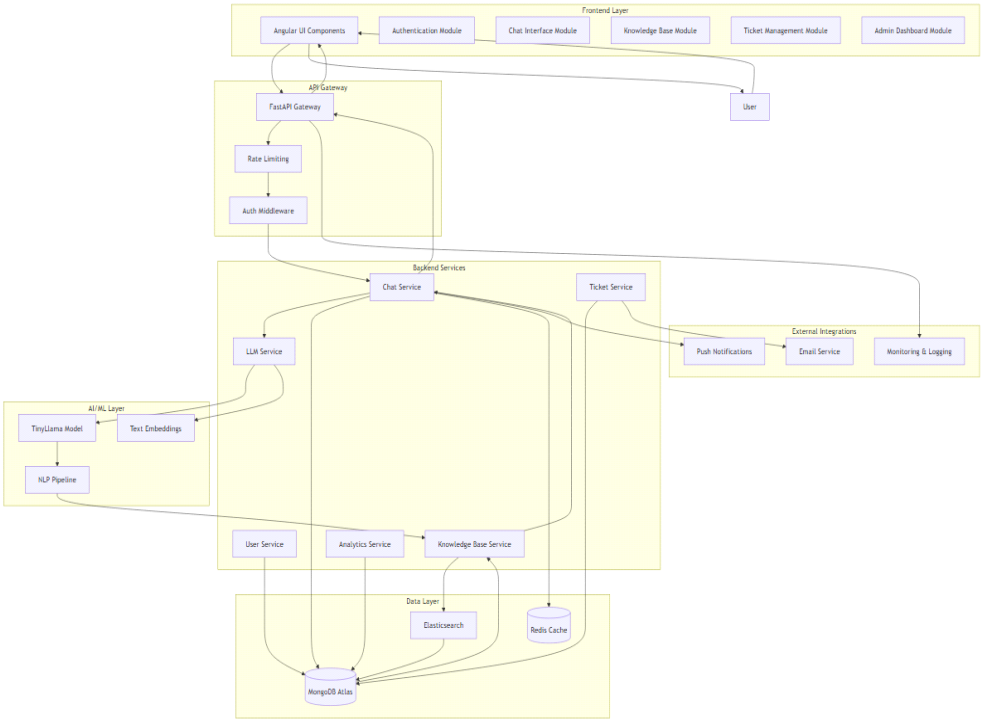

Fig. 1. Modular Architecture Diagram of Assist-Nexus

1 High-Level Architecture

The high-level architecture of ASSIST-NEXUS consists of the following key components:

– Frontend (Angular): The frontend is responsible for rendering the user interface and handling user interactions. It communicates with the backend via HTTP requests and WebSockets for real-time updates;

– Backend (FastAPI): The backend is responsible for processing user re-quests, interacting with the LLM and database, and returning responses to the frontend. It is implemented as a set of RESTful APIs and WebSocket endpoints;

– LLM Service (TinyLlama): The LLM service is responsible for generating responses to user inquiries. It is implemented as a separate microservice that ex-poses an API for text generation, allowing it to be scaled independently of the main backend;

– Database (MongoDB Atlas): The database is responsible for storing user data, conversation history, and knowledge base articles. It provides persistence and data retrieval capabilities for the application;

– Cache: The cache is used to store frequently accessed data, such as user sessions and knowledge base articles, reducing database load and improving response times;

– Search Service: The search service is responsible for indexing and searching knowledge base articles, providing fast and relevant search results for user inquiries;

– Analytics Service: The analytics service collects and visualizes metrics on system performance, user interactions, and assistant effectiveness.

2 Component Interaction

The interaction between components follows a structured process to ensure smooth communication and efficient handling of user requests.

The user initiates interaction through the frontend, sending a request in natural language via the chat interface. This request is transmitted to the backend through an HTTP request or WebSocket message, enabling real-time communication.

Upon receiving the request, the backend performs several key tasks. First, it authenticates the user and verifies their permissions to ensure secure access. It then retrieves the conversation history from the database, allowing continuity in dialogue. Additionally, the backend searches the knowledge base for relevant articles using the search service, ensuring the response is enriched with available information. Once all neces-

sary data is gathered, the backend forwards the user request, conversation history, and relevant knowledge base articles to the LLM service for processing.

The LLM service analyzes the provided input and generates a response by considering the user's query, conversation context, and knowledge base references. This ensures accuracy and relevance in the assistant's reply.

The backend receives the generated response and executes several additional operations. It stores the user request and assistant response in the conversation history to maintain a detailed record. Analytics data on the interaction is also updated, providing valuable insights into user behavior and system efficiency. Finally, the backend returns the response to the frontend for display.

The frontend presents the response to the user within the chat interface, completing the primary interaction cycle.

In cases where the LLM service is unable to generate a satisfactory response, the backend creates a support ticket within the ticket management system and notifies a human agent, ensuring that complex queries are addressed effectively.

ASSIST-NEXUS uses MongoDB Atlas as its primary database (see Figure – 2) , leveraging its flexible document-based data model, scalability, and high performance. The database is designed to store various types of data, including user information, conversation history, knowledge base articles, and support tickets.

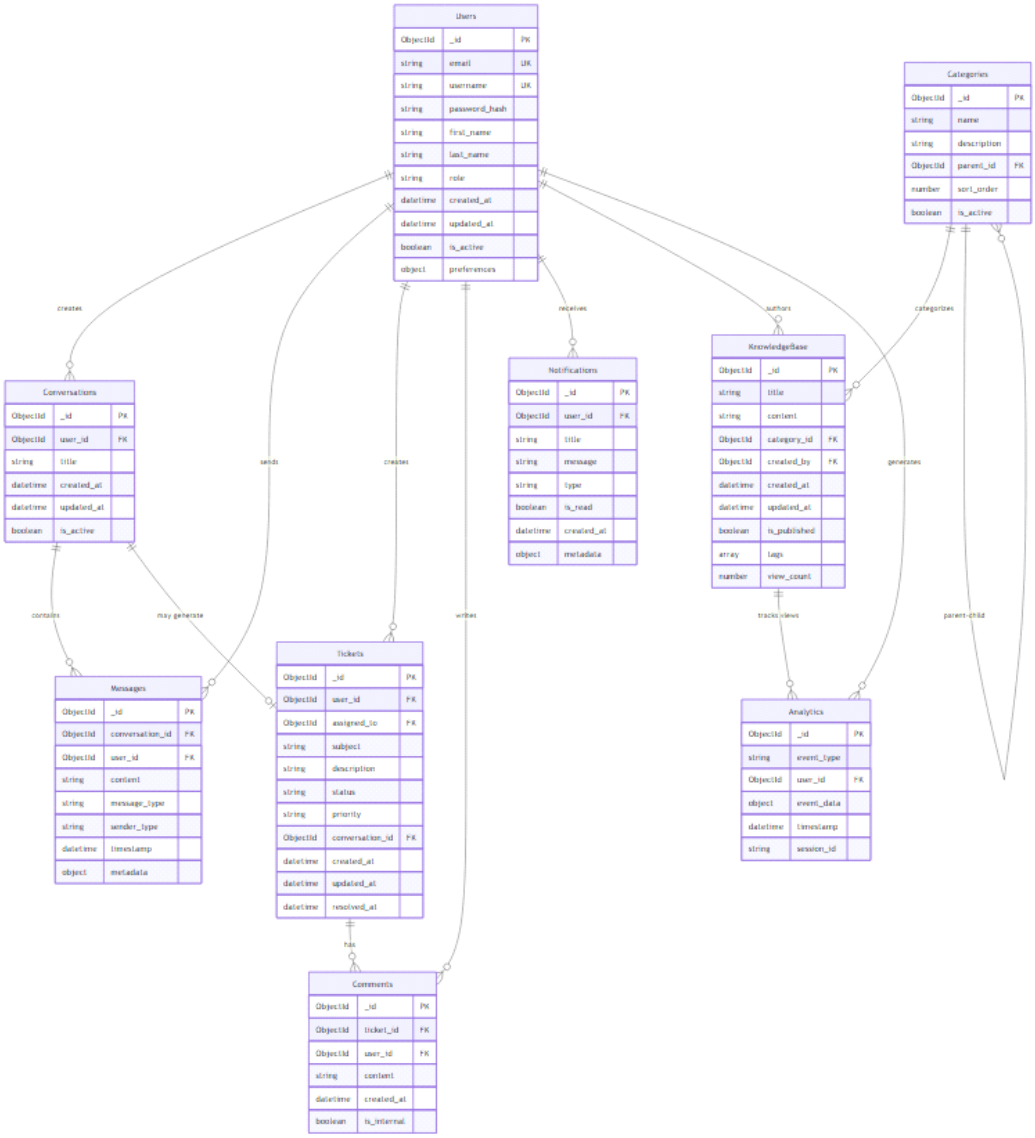

Fig. 2. Database Collections scheme

2.3.1 Database Collections

The MongoDB database consists of the following main collections:

– Users: Stores user information, including authentication details, profile data, and preferences;
– Conversations: Stores the history of conversations between users and the virtual assistant;
– Messages: Stores individual messages within conversations, including both user inputs and assistant responses;
– KnowledgeBase: Stores articles containing information about the company's products, services, and policies;
– Categories: Stores categories for organizing knowledge base articles;
– Tickets: Stores information about support tickets, including the user who created the ticket, the subject, the description, and the status;
– Comments: Stores comments on support tickets, allowing for communication between users and support agents;
– Notifications: Stores notifications for users, such as ticket updates and system announcements;
– Settings: Stores system-wide settings and configuration options;
– Analytics: Stores aggregated analytics data on system usage and performance.

References

1. FastAPI: A modern, fast (high-performance) web framework for building APIs with Python. [https://fastapi.tiangolo.com/]
2. Angular: A platform and framework for building single-page client applica-tions using HTML, CSS, and TypeScript. [https://angular.io/]
3. TinyLlama: A compact and efficient large language model (LLM) designed for local deployment. [https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0]
4. MongoDB Atlas: A fully managed cloud database service for MongoDB, a document-oriented NoSQL database. [https://www.mongodb.com/atlas]
5. Sentence Transformers: Python framework for state-of-the-art sentence, text and image embeddings. [https://www.sbert.net/]
6. JSON Web Tokens (JWT): An open, industry standard RFC 7519 method for representing claims securely between two parties. [https://jwt.io/]
7. Model-View-Presenter (MVP): A software architectural pattern for developing user interfaces that divides the application logic into three interconnected elements. [https://en.wikipedia.org/wiki/Model-view-presenter]