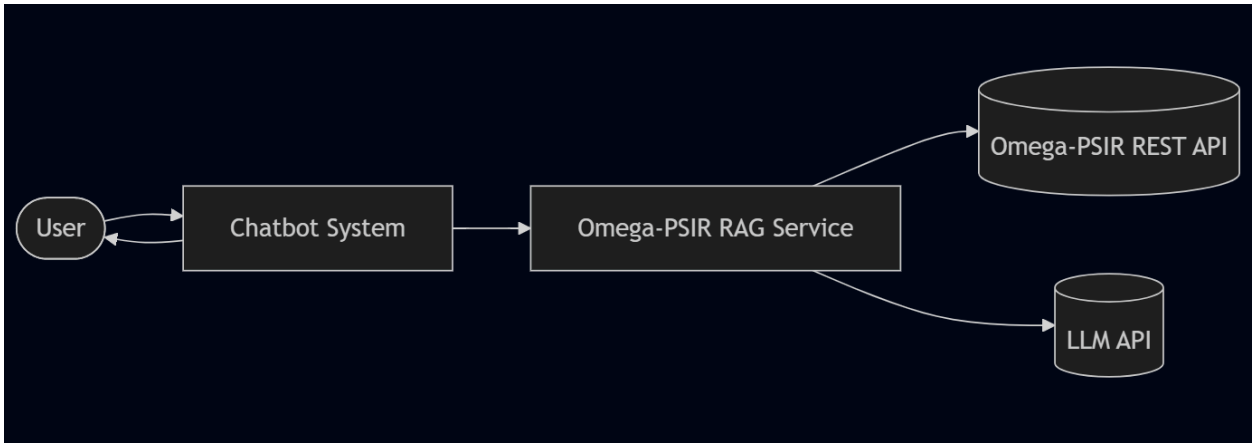
Architektura

1. Perspektywa kontekstowa - C4 level 1 (context)

   System umożliwia użytkownikowi zadawanie pytań dotyczących zasobów Bazy Wiedzy Omega-PSIR w języku naturalnym. Chatbot współpracuje z mikroserwisem RAG, który wyszukuje odpowiednie dokumenty na podstawie semantycznych embeddingów oraz dostarcza kontekst dla modelu językowego generującego odpowiedź.

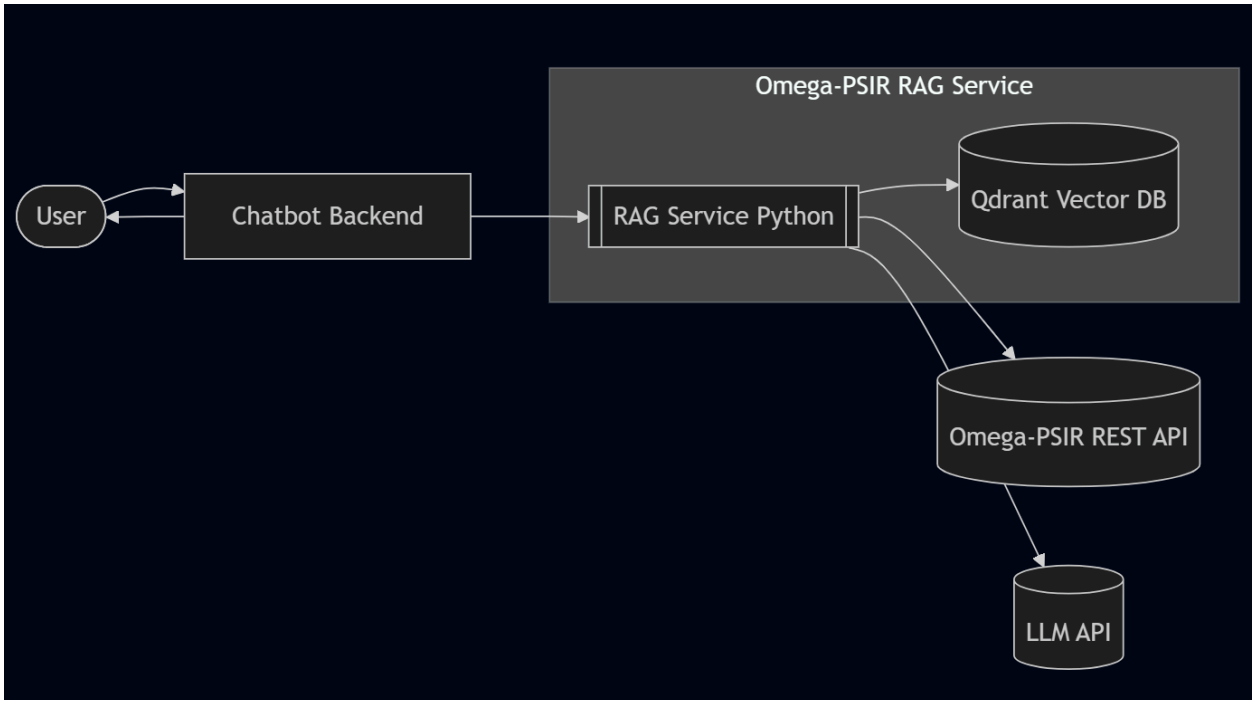


   User - zadaje pytania przez interfejs chatbota.
   Chatbot (system innego zespołu) - odbiera pytania od użytkownika i komunikuje się z mikroserwisem RAG
   Omega-PSIR RAG Service (nasz system) - odbiera zapytania, wykonuje wyszukiwanie semantyczne, dostarcza kontekst.
   Omega-PSIR REST API - dostarcza źródłowe dane, z których aktualizowana jest baza wektorowa.
   LLM API - generuje odpowiedź w języku naturalnym na podstawie kontekstu zwróconego przez nasz system.

2. Perspektywa kontenerów - C4 level 2 (containers)

   System jest wdrożony na jednym serwerze i składa się z 2 kontenerów uruchomieniowych:

   - RAG Service - Python. Wykonywanie wektoryzacji, aktualizacji indeksu, wyszukiwania semantycznego.
   - Qdrant Vector Database - Docker. Przechowywanie embeddingów i metadanych dokumentów oraz wyszukiwanie wektorowe.



Komunikacja między kontenerami i systemami zewnętrznymi:

- Chatbot -> RAG Service - HTTP/REST
- RAG Service -> Qdrant DB - REST API
- RAG Service -> Omega-PSIR REST API - HTTP
- RAG Service -> LLM API - HTTP/REST

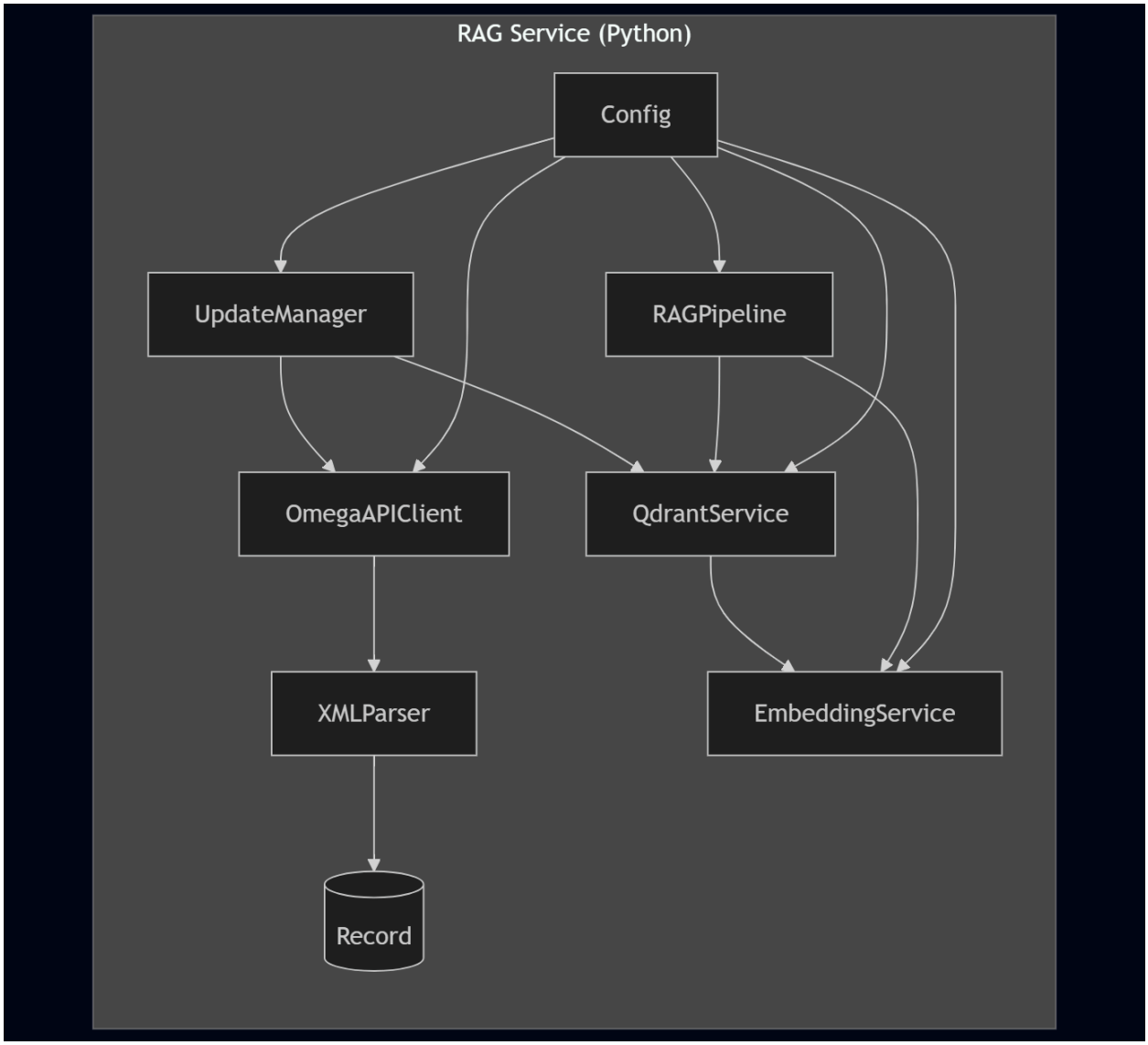
Zachowanie systemu:

- RAG Service okresowo pobiera nowe dane z Omega-PSIR REST API
- Zaktualizowane rekordy są wektoryzowane i zapisywane w Qdrant DB.
- W momencie zapytania użytkownika, RAG Service generuje embedding zapytania, wyszukuje najbardziej trafne dokumenty, a następnie przekazuje kontekst do LLM API w celu wygenerowania odpowiedzi.

3. Perspektywa komponentów - C4 level 3 (components)

Kontener RAG Service jest zbudowany z następujących komponentów:

- RAGPipeline: obsługa zapytania od chatbota: retrieval + generowanie odpowiedzi.
- QdrantService: komunikacja z bazą wektorową, zapis i aktualizacja rekordów.
- EmbeddingService: generowanie embeddingów oraz detekcja języka.
- OmegaAPIClient: pobieranie danych z Omega-PSIR REST API.
- XMLParser: przetwarzanie danych XML na obiekty Record.
- UpdateManager: cykliczna aktualizacja rekordów i embeddingów.
- Config: ładowanie i udostępnianie ustawień konfiguracyjnych.
- Record: model domenowy opisujący rekord z bazy Omega-PSIR.


RAG Service (Python)
Config
UpdateManager
RAGPipeline
OmegaAPIClient
QdrantService
XMLParser
EmbeddingService
Record