

1. Wprowadzenie

Problem biznesowy: Agent asystujący przy planowaniu posiłków. Ma wyszukiwać przepisy na podstawie tego co użytkownik ma w lodówce, tworzyć listy zakupów na podstawie przepisów, które użytkownik chce wykonać.

Użyte technologie: Neo4j, Langchain, Tavily, Azure Cognitive Services, Azure Embedding.

2. Opis modułów

Przechowywanie danych: Dane *lodówki* i *listy zakupów* przechowywane są lokalnie w folderze: **data** w formacie **json**. Zarządzanie nimi(*odczyt, zapis, dodawanie, usuwanie*) odbywa się za pomocą klasy **ListBlueprint**, która jest implementowana jest odpowiednio przez klasy: **Fridge** i **ShoppingList**.

Odczyt danych z pisma ręcznego: System pozwala na odczyt danych z pisma ręcznego. Zajmuje się tym klasa: **PhotoReader**. Zdjęcie jest odczytywane w postaci *bajtów*, następnie jest analizowane przez **ImageAnalysisClient**. Dane odczytane przez klienta zostają odpowiednio przetworzone przez *llm* (w tym projekcie użyto: **gpt-5-nano**), oraz zwrocione jako **json**.

Zaciąganie danych ze strony AniaGotuje: Dane ze strony aniatgotuje.pl są zaciągane za pomocą **scraperaAniaGotujeSpider**(startując z url: <https://api.aniatgotuje.pl/client/posts/search?perPage=200&page=0&sort=publish,desc>) i dodawane w ustrukturyzowanej formie za pomocą **RecipeDataPipeline**.

Agent: Za zarządzanie systemem i przetwarzanie zapytań użytkownika odpowiada

Agent(FridgeChatbot), znajdujący się w pliku **chatbot_with_tools.py**. Prompty znajdują się w pliku **fridge_tools.py** i **shopping_list_tools**. Konfiguracje **RAG** i **Graph RAG** znajdują się w **rag_tools.py**.

Front: Front został wykonany w **streamlit** i znajduje się w pliku **app.py**. Jest prosty i czytelny, na samym początku użytkownika widać pole tekstowe zachęcające do rozpoczęcia integracji z asystentem, po prawej stronie widoczne jest podsumowanie co użytkownik ma odpowiednio w *Lodówce* i na *Liście zakupów*, a z prawej znajduje się lista wyboru widoku: *Chatbot*, *Lodówka*, *Lista zakupów*, oraz lista z wyborem opcji. Wybranie opcji *Neo4j* pozwala na dalszy wybór: *RAG*, albo *Graph RAG*.

Toole:

- **read_fridge** -> Zwraca listę składników z lodówki
- **add_ingredient_to_fridge** -> Dodaje nowy składnik do listy składników w lodówce
- **remove_ingredient_from_fridge** -> Usuwa składnik z listy składników w lodówce
- **add_ingredient_to_shopping_list** -> Dodaje nowy składnik do listy zakupów
- **add_missing_ingredients_for_recipe** -> Dodaje do listy zakupów produkty potrzebne do wykonania przepisu, których nie ma w lodówce. Używany, gdy użytkownik wyraźnie o to poprosi
- **create_shopping_list_file_for_recipe** -> Zwraca listę brakujących składników, gdy użytkownik nie chce dodawać tych składników do listy

3. Knowledge Graph

Przykładowy fragment grafu:



Każdy **przepis** ma relację: **CONTAINS** ze składnikami, które go tworzą, **BELONGS_TO** z rodzajem kuchni do której należy(np. polska, włoska itp), **SUITABLE_FOR** dla rodzaju diety (np. bezglutenowa). Pozwala to na dokładne wyszukiwanie przepisów w zależności od preferencji użytkownika i od składników, które ma, bądź chce użyć.

4. Zaawansowane zapytania

Przykładowa konwersacja z chatbotem:

Asystent Kulinarny

🤖

💡 W Twojej lodówce znajdują się następujące składniki:

- Jajka: 6 szt.
- Mleko: 200 ml
- Banany: 2 szt.
- Pomidor: 1 szt.

Najwięcej masz jajek (6 sztuk).

🤖

💡 Znalazłem kilka przepisów wykorzystujących jajka. Oto one:

1. **Muffinki z truskawkami**
Link: [przepis tutaj](#)
Brakujące składniki: czekolada biała, cukier drobny, olej, mąka pszenna tortowa, świeże truskawki, mleko, soda i proszek do pieczenia
2. **Stefanka**
Link: [przepis tutaj](#)
Brakujące składniki: kasza manna (również błyskawiczna), masło, miód naturalny (najlepiej płynny), czekolada gorzka (np. 64%), mleko (dowolne), soda oczyszczona, kwaśny dżem (np. z czarnej porzeczkii), cukier, cukier puder, mąka pszenna uniwersalna

3. Skyrnik czekoladowy

Link: [przepis tutaj](#)

Brakujące składniki: borówki lub jagody leśne, Skyr naturalny (nie pitny), maliny lub jezyny, kakao, ewentualna dekoracja (polewa czekoladowa z masła), erytrol lub zwykły cukier, budyń waniliowy bez cukru

4. Jajka na twardo

Link: [przepis tutaj](#)

Brakujące składniki: brak

5. Jajka na miękko

Link: [przepis tutaj](#)

Brakujące składniki: brak

6. Babka z serem Mascarpone

Link: [przepis tutaj](#)

Brakujące składniki: skórka otarta z cytryny, mąka pszenna tortowa, ser mascarpone, cukier drobny, proszek do pieczenia, olej

7. Gniazdka

Link: [przepis tutaj](#)

Brakujące składniki: sok z mandarynki lub cytryny, olej roślinny (np. rzepakowy), sól, woda, cukier puder, mąka pszenna tortowa, mleko

Jeśli któryś z przepisów Cię interesuje, daj mi znać, a przygotuję listę brakujących składników do jego wykonania!



Interesuje mnie babka z serem mascarpone, więc proszę przygotuj listę brakujących składników do jego wykonania



Do wykonania babki z serem mascarpone dodano do listy zakupów następujące brakujące składniki:

1. Skórka otarta z cytryny
2. Mąka pszenna tortowa
3. Ser mascarpone
4. Jajka
5. Cukier drobny
6. Proszek do pieczenia
7. Olej

Jeśli chcesz, mogę również wygenerować plik z tą listą. Daj znać!



Przecież mam jajka, czyżbym potrzebował ich więcej?

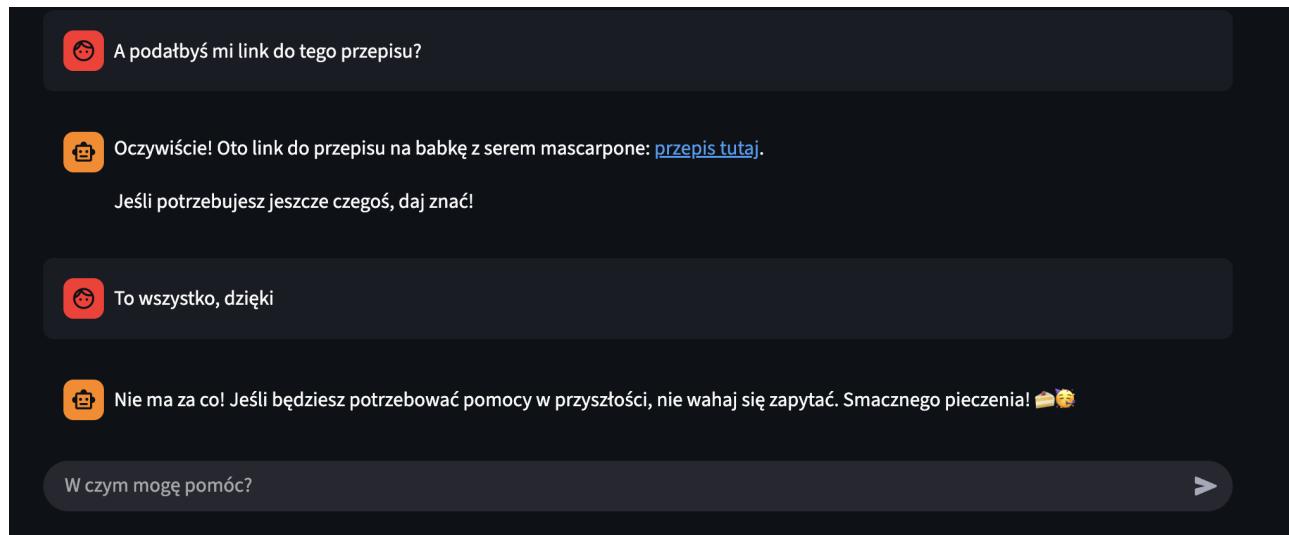


Masz 6 jajek w lodówce, co powinno być wystarczające do przygotowania babki z serem mascarpone, ponieważ zazwyczaj do tego przepisu potrzebne są 2-4 jajka, w zależności od dokładnej receptury.

Zaktualizowana lista brakujących składników do babki z serem mascarpone:

1. Skórka otarta z cytryny
2. Mąka pszenna tortowa
3. Ser mascarpone
4. Cukier drobny
5. Proszek do pieczenia
6. Olej

Chcesz dodać te składniki do listy zakupów, czy wolisz, abym wygenerował plik z listą?



5. Metryki

Pomiary znajdują się w osobnym pliku pdf.

Wnioski:

GraphRag trwa prawie dwukrotnie dłużej, ale częściej proponuje dodanie brakujących składników do listy zakupów, oraz nawet gdy twierdził, że nie wie co mam w lodówce, to proponował przepis uwzględniający zawartość lodówki. RAG za to działał szybciej, brzmiał przekonująco, ale gdy popełniał błędy, to bardzo dziwaczne jak pomylenie produktu z danym. Takich błędów nie popełniał GraphRag

6. Wnioski i rekomendacje

7. Załączniki

Miejsce przechowywania projektu: [Repozytorium na GitHub](#)

Konfiguracja:

1. Wszystkie potrzebne paczki znajdują się w pliku: **requirements.txt**, należy je pobrać.

2. Stworzyć kontener z **neo4j**

1. Pobrać obraz **neo4j**

2. Wywołać komendę:

```
docker run \ -p 7474:7474 -p 7687:7687 \ -v $PWD/data:/data -v  
$PWD/plugins:/plugins \ --name neo4j-apoc \ -e  
NE04J_apoc_export_file_enabled=true \ -e  
NE04J_apoc_import_file_enabled=true \ -e  
NE04J_apoc_import_file_use_neo4j_config=true \ -e  
NE04JLABS_PLUGINS=\["apoc"\] \ neo4j
```

3. Wywołać komendę: **.venv/bin/activate** , żeby aktywować środowisko wirtualne

4. Przygotować plik: **.env**, żeby posiadał następujące dane:

```
AZURE_OPENAI_ENDPOINT="https://s20519.openai.azure.com/openai/deployments/gpt-4o-mini/chat/completions?api-version=2025-01-01-preview"  
AZURE_OPENAI_API_KEY="" OPENAI_API_VERSION="2025-01-01-preview"  
TAVILY_API_KEY=""  
COGNITIVE_API="https://fridgepoc.cognitiveservices.azure.com/"  
COGNITIVE_KEY="" NE04J_PASSWORD="" AZURE_EMBEDDING_API_KEY=""
```

AZURE_EMBEDDING_API_VERSIONS AZURE_EMBEDDING_ENDPOINT

AZURE_EMBEDDING_MODEL (*klucze i hasło do **neo4j** należy wypełnić swoimi danymi*)

5. Uruchomić scraper komendą: `scrapy crawl aniatotuje_spider`

6. Uruchomić aplikację komendą: `streamlit run app.py`