

S.I.K.O.R.K.A.

System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz



Specyfikacja Techniczna

Wersja 1.0

Data Grudzień 2025

Zespół Wifi nie działa

Hackathon Hacknation/Scenariusze Jutra

„Mała, ale inteligentna – jak sikorka, która szybko przetwarza informacje z otoczenia”

Spis Treści

1. Wprowadzenie
 1. Nazwa Projektu
 2. Cel Projektu
 3. Potencjał Wdrożeniowy
2. Architektura Systemu
 1. Diagram C4 – Poziom Kontekstu
 2. Diagram C4 – Poziom Kontenerów
 3. Diagram C4 – Poziom Komponentów
3. Diagram Sekwencji – Submit Job Flow
4. Obsługiwane Formaty Danych
 1. Dokumenty
 2. Źródła Webowe
5. Stos Technologiczny
 1. Frontend
 2. Backend
 3. Baza Danych
 4. LLM & AI
6. Model Danych
 1. Diagram ERD
 2. Typy Węzłów
 3. Typy Relacji
7. API Endpoints
 1. Główne Endpointy
 2. Przykład Request/Response

8. Pipeline Przetwarzania

1. Szczegóły Kroków

9. Struktura Raportu

10. Konfiguracja i Deployment

1. Zmienne Środowiskowe

2. Uruchomienie

3. Wymagania Sprzętowe

11. Bezpieczeństwo

12. Analiza Kosztów LLM

1. Cloudflare Workers AI

2. Self-Hosted (Ollama)

3. Rekomendacja

13. Rozszerzalność

1. Łatwa Podmiana LLM

2. Dodawanie Nowych Formatów

3. Graf Wiedzy

14. Podsumowanie

S.I.K.O.R.K.A.

System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz



Specyfikacja Techniczna

Wersja 1.0

Data Grudzień 2025

Zespół Wifi nie działa

Hackathon Hacknation/Scenariusze Jutra

„Mała, ale inteligentna – jak sikorka, która szybko przetwarza informacje z otoczenia”

Spis Treści

1. [Wprowadzenie](#)
 2. [Architektura Systemu](#)
 3. [Diagram Sekwencji](#)
 4. [Obsługiwane Formaty Danych](#)
 5. [Stos Technologiczny](#)
 6. [Model Danych](#)
 7. [API Endpoints](#)
 8. [Pipeline Przetwarzania](#)
 9. [Struktura Raportu](#)
 10. [Konfiguracja i Deployment](#)
 11. [Bezpieczeństwo](#)
 12. [Analiza Kosztów LLM](#)
 13. [Rozszerzalność](#)
 14. [Podsumowanie](#)
-

1. Wprowadzenie

1.1 Nazwa Projektu

S.I.K.O.R.K.A. – System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz

Nazwa projektu nawiązuje do ptaka sikorki – małego, ale niezwykle inteligentnego i adaptacyjnego stworzenia, które potrafi szybko przetwarzać informacje z otoczenia i dostosowywać swoje zachowanie. Podobnie nasz system szybko przetwarza różnorodne źródła danych i adaptuje się do potrzeb analityka.

1.2 Cel Projektu

System **S.I.K.O.R.K.A.** to narzędzie analityczne wykorzystujące modele LLM (Large Language Models) do automatycznej ekstrakcji faktów, generowania predykcji i tworzenia raportów analitycznych dla fikcyjnego państwa **Atlantis**. System wspiera proces podejmowania decyzji strategicznych poprzez:

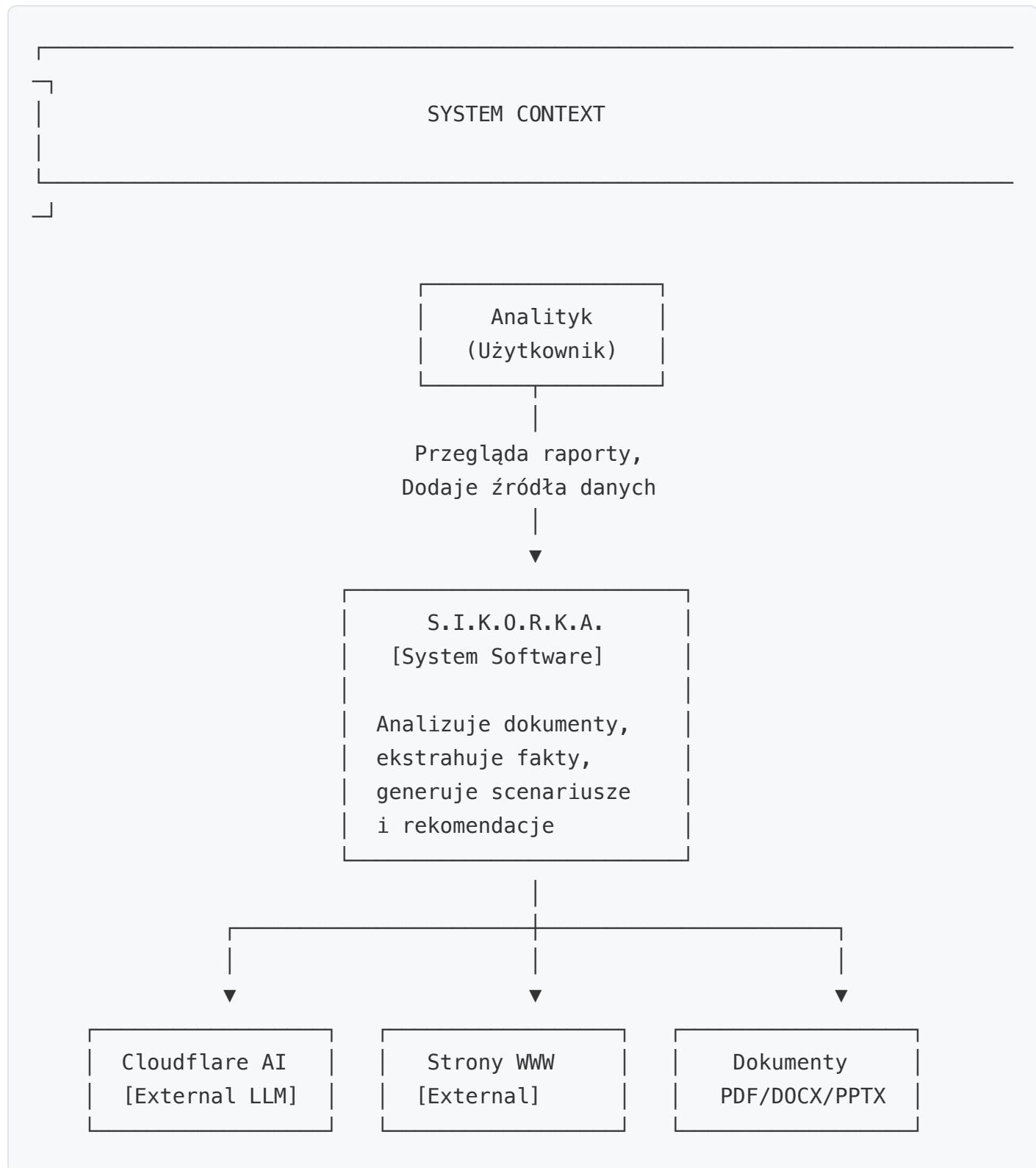
- Automatyczną ekstrakcję kluczowych faktów z różnorodnych źródeł
- Identyfikację potencjalnych scenariuszy (pozytywnych i negatywnych)
- Wykrywanie brakujących informacji (unknowns)
- Generowanie kompleksowych raportów z rekomendacjami

1.3 Potencjał Wdrożeniowy

| Aspekt | Opis |
|---------------------------|---|
| Konteneryzacja | Pełna konteneryzacja Docker – system działa na dowolnej infrastrukturze wspierającej Docker |
| Modele Open Source | Wykorzystanie wyłącznie modeli open-source (Llama 3.1, Qwen3) – brak vendor lock-in |
| Elastyczność LLM | Prosty mechanizm podmiany providera LLM (zmienna <code>LLM_PROVIDER</code>) |
| Koszty | Podczas intensywnych testów zużyto ~10k neuronów Cloudflare (~\$0.11) |
| Self-hosting | Możliwość hostowania własnego LLM (Ollama) z określonym, skalowalnym kosztem |
| Skalowalność | Architektura mikroserwisowa umożliwia niezależne skalowanie komponentów |

2. Architektura Systemu

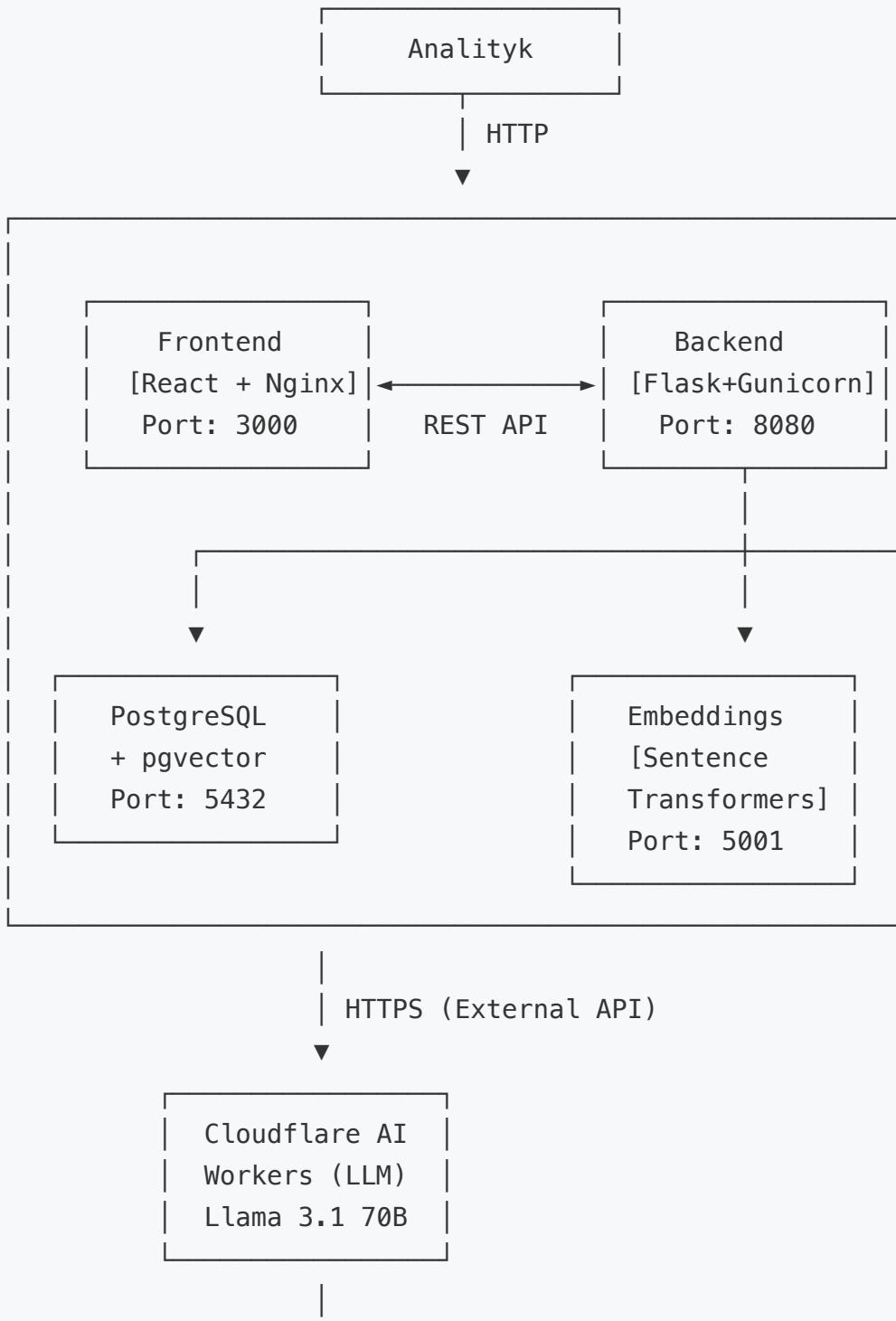
2.1 Diagram C4 – Poziom Kontekstu



2.2 Diagram C4 – Poziom Kontenerów

CONTAINER DIAGRAM

Docker Network: hacknation–network



Alternatywnie:

Ollama LLM
(Self-hosted)
Port: 11434

2.3 Diagram C4 – Poziom Komponentów (Backend)

COMPONENT DIAGRAM – BACKEND

Flask Application

| /api/submit | | /api/jobs/:id | | /api/nodes | | /health |

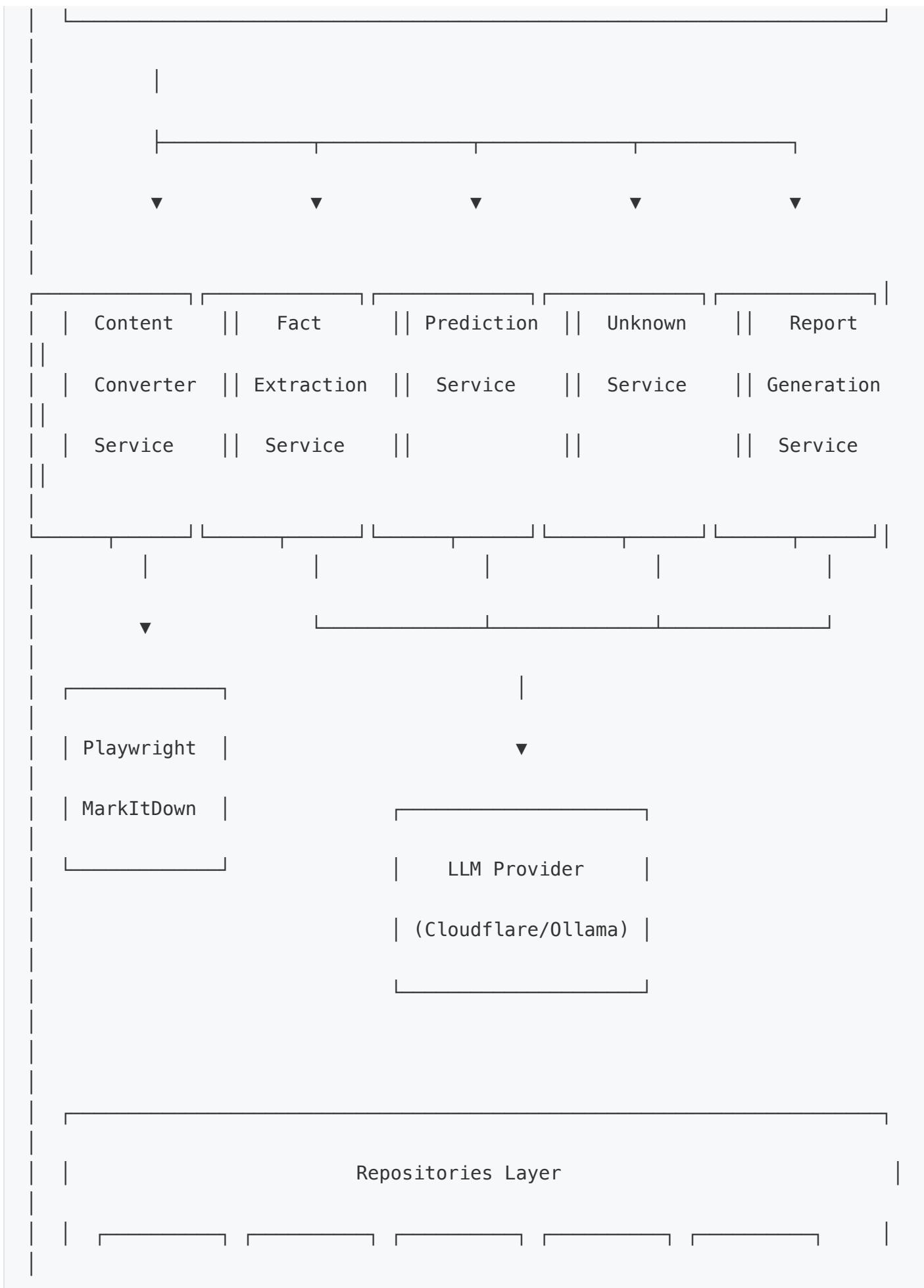
| [Endpoint] | | [Endpoint] | | [Endpoint] | | [Endpoint] |

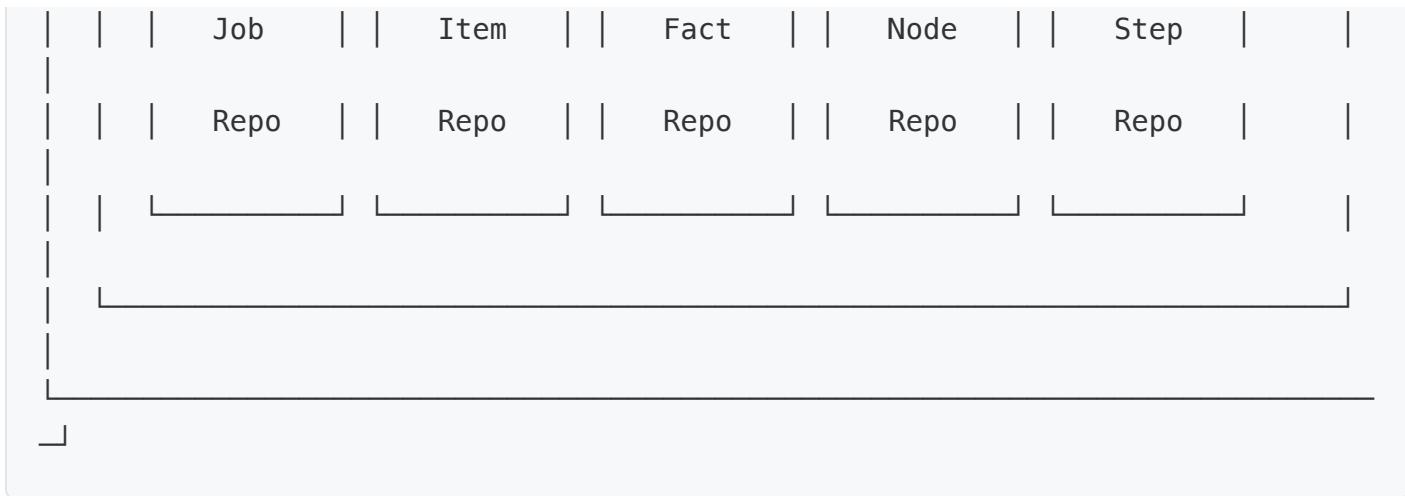
| | |

▼ ▼ ▼

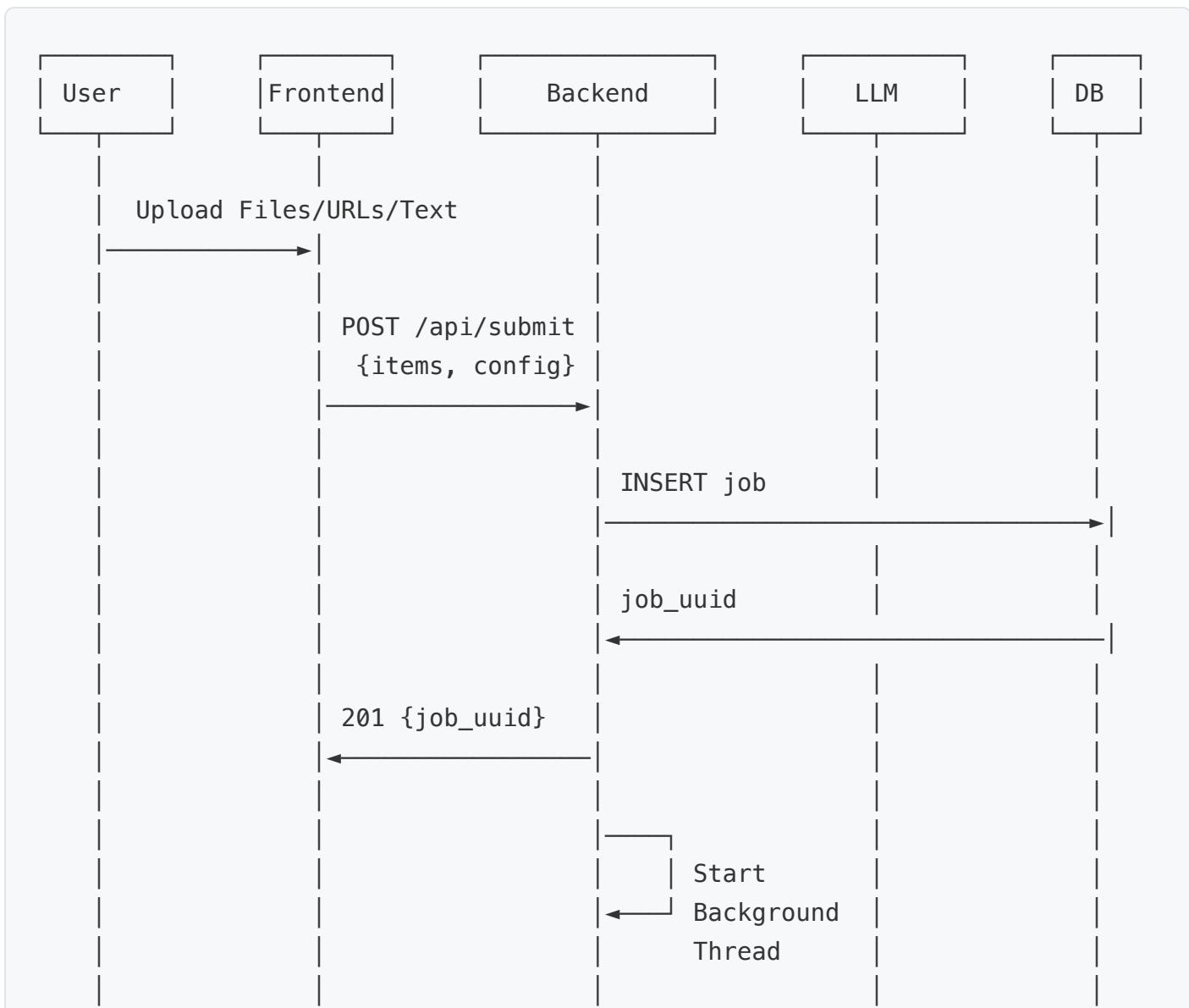
ProcessingService

[Orchestrator Pattern]





3. Diagram Sekwencji – Submit Job Flow



|| STEP 1: Content Conversion ||

PDF/DOCX → Markdown

URL → Playwright → HTML → MD

|| STEP 2: Fact Extraction ||

Extract facts

[facts list]

Store facts as nodes

|| STEP 3: Prediction Extract ||

Extract predictions

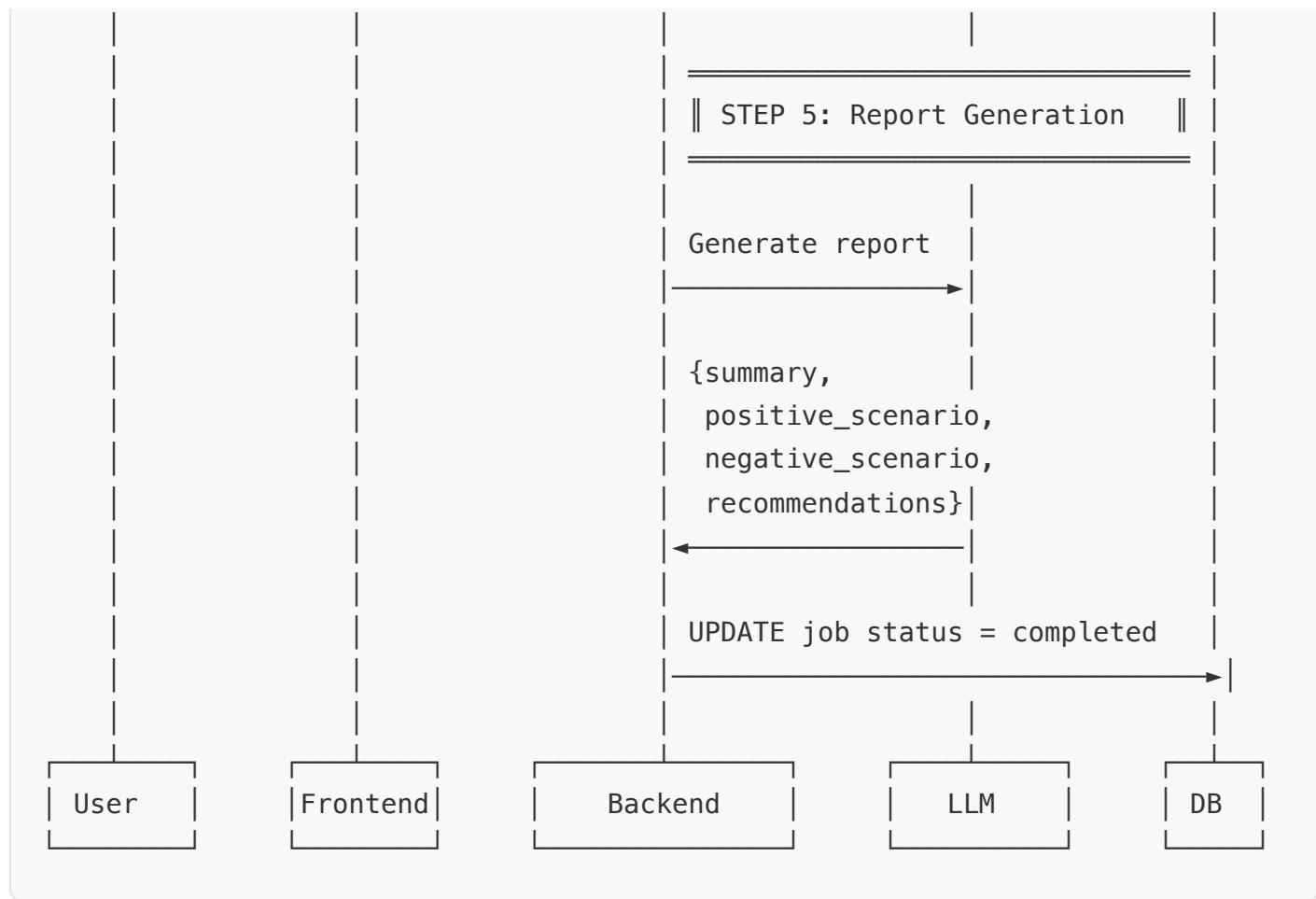
[predictions]

Store predictions + relations

|| STEP 4: Unknowns Extraction ||

Identify gaps

[unknowns]



4. Obsługiwane Formaty Danych

System akceptuje różnorodne formaty wejściowe:

4.1 Dokumenty

| Format | Rozszerzenie | Biblioteka konwersji |
|------------|--------------|----------------------|
| PDF | .pdf | MarkItDown |
| Word | .docx | MarkItDown |
| PowerPoint | .pptx | MarkItDown |
| Excel | .xlsx | MarkItDown |

| Format | Rozszerzenie | Biblioteka konwersji |
|--------|--------------|----------------------|
| Tekst | plain text | Native |

4.2 Źródła Webowe

| Typ | Obsługa | Technologia |
|------------------------|---------|--|
| Strony statyczne | ✓ | requests + BeautifulSoup |
| Strony dynamiczne (JS) | ✓ | Playwright – czeka na pełne załadowanie JS |
| SPA (Single Page Apps) | ✓ | Playwright z <code>wait_until='networkidle'</code> |

Kluczowa funkcjonalność: System wykorzystuje Playwright do renderowania stron dynamicznych. Po nawigacji do URL czeka na zakończenie aktywności sieciowej (`networkidle`), co gwarantuje pobranie treści generowanych przez JavaScript.

5. Stos Technologiczny

5.1 Frontend

| Technologia | Wersja | Rola |
|--------------|--------|-------------------|
| React | 18.x | Framework UI |
| Vite | 6.x | Build tool |
| React Router | 6.x | Routing |
| Tailwind CSS | 3.x | Stylowanie |
| Lucide React | - | Ikony |
| Nginx | latest | Production server |

5.2 Backend

| Technologia | Wersja | Rola |
|----------------|--------|-------------------------|
| Python | 3.11+ | Runtime |
| Flask | 3.0.0 | Framework API |
| Gunicorn | 22.0.0 | WSGI Server (4 workers) |
| psycopg2 | 2.9.9 | PostgreSQL driver |
| pgvector | 0.2.4 | Vector operations |
| Playwright | latest | Web scraping dynamiczny |
| MarkItDown | latest | Konwersja dokumentów |
| BeautifulSoup4 | 4.12.2 | HTML parsing |

5.3 Baza Danych

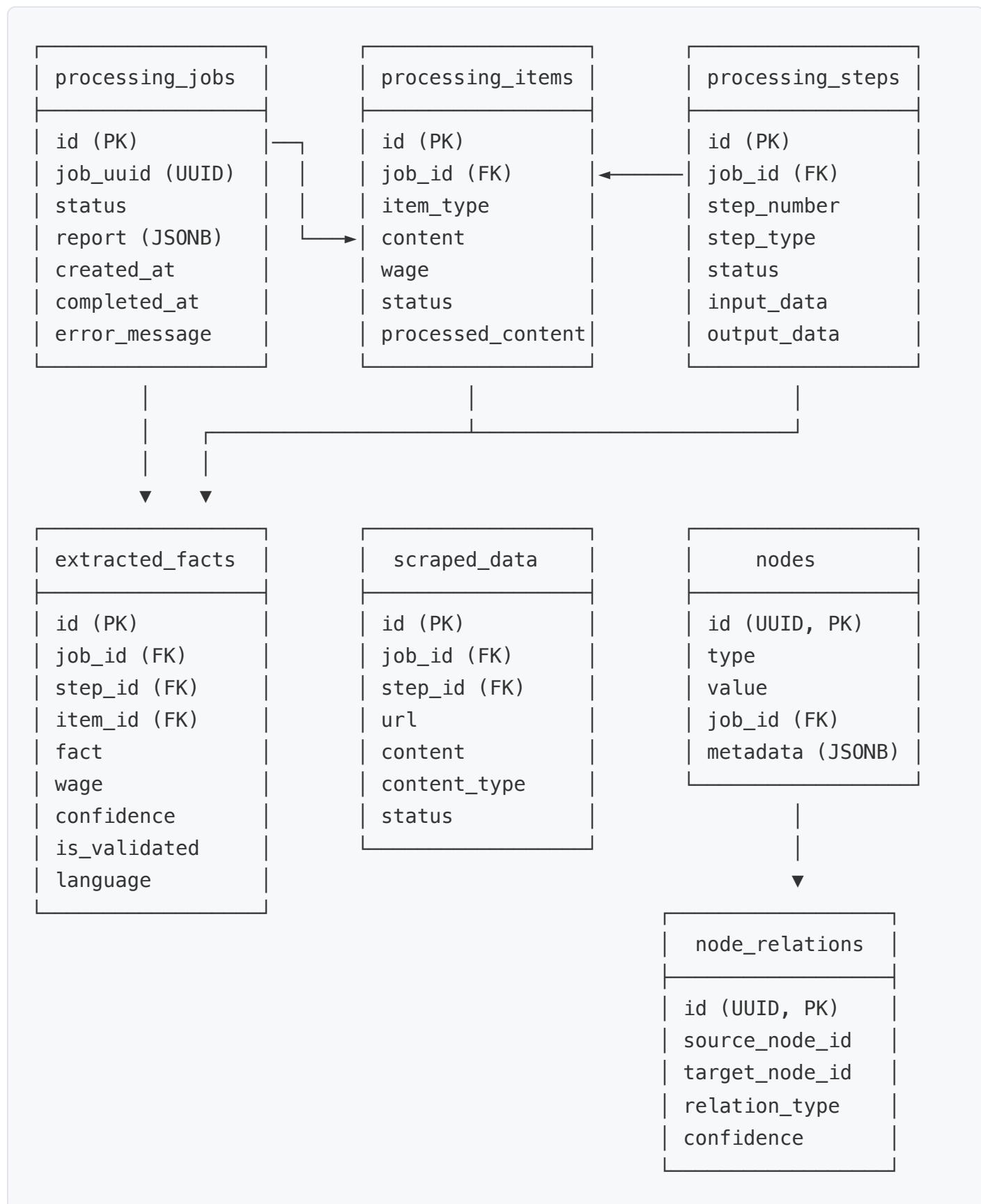
| Technologia | Wersja | Rola |
|-------------|-----------|------------------------|
| PostgreSQL | 16 | Główna baza danych |
| pgvector | extension | Wyszukiwanie wektorowe |

5.4 LLM & AI

| Provider | Model | Użycie |
|-----------------------|------------------------|-----------------------------|
| Cloudflare Workers AI | Llama 3.1 70B Instruct | Główny provider (hostowany) |
| Ollama | Qwen3:30b-a3b / Llama2 | Alternatywny (self-hosted) |
| Sentence Transformers | MiniLM-L12-v2 | Embeddingi multilingualne |

6. Model Danych

6.1 Diagram ERD



6.2 Typy Węzłów (Nodes)

| Typ | Opis |
|---------------------|--|
| fact | Wyekstrahowany fakt z dokumentu |
| prediction | Predykcja/prognoza wynikająca z faktów |
| missing_information | Zidentyfikowana luka informacyjna |
| report | Wygenerowany raport końcowy |

6.3 Typy Relacji

| Typ | Opis |
|--------------|--|
| derived_from | Węzeł wynika z innego węzła |
| supports | Węzeł wspiera/potwierdza inny węzeł |
| contradicts | Węzeł jest sprzeczny z innym węzłem |
| requires | Węzeł wymaga informacji z innego węzła |
| suggests | Węzeł sugeruje/implikuje inny węzeł |

7. API Endpoints

7.1 Główne Endpointy

| Metoda | Endpoint | Opis |
|--------|-------------|-----------------------------------|
| GET | /health | Health check |
| POST | /api/submit | Utworzenie nowego zadania analizy |

| Metoda | Endpoint | Opis |
|--------|---------------------------|---|
| GET | /api/jobs | Lista wszystkich zadań |
| GET | /api/jobs/{uuid} | Szczegóły zadania z faktami, predykcjami i raportem |
| GET | /api/jobs/{uuid}/nodes | Węzły powiązane z zadaniem |
| GET | /api/nodes/{id} | Szczegóły pojedynczego węzła |
| GET | /api/nodes/{id}/relations | Relacje węzła |

7.2 Przykład Request/Response

POST /api/submit

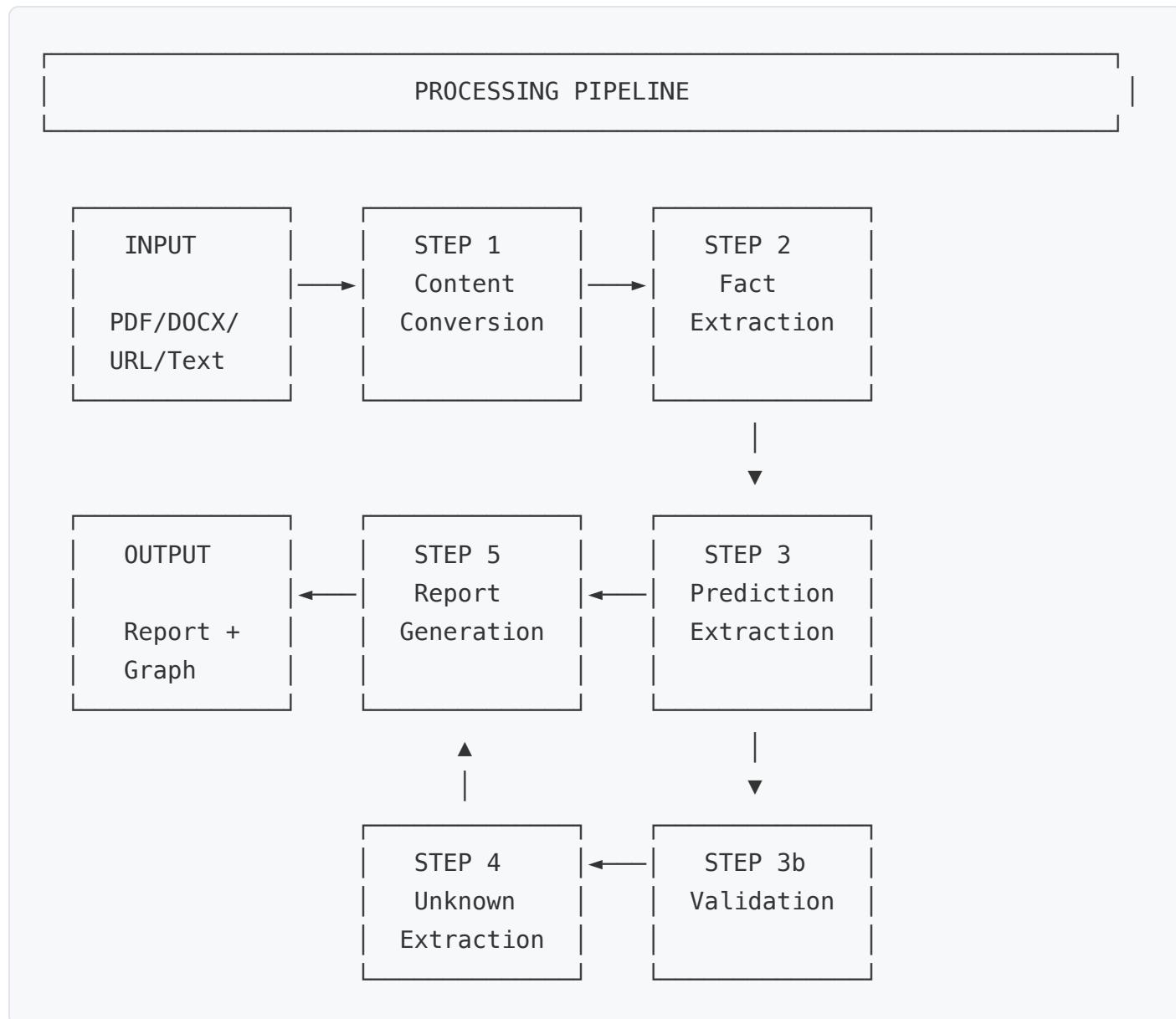
```
{
  "items": [
    {"type": "text", "content": "Treść do analizy", "wage": 100},
    {"type": "link", "content": "https://example.com/article", "wage": 80},
    {"type": "file", "content": "<base64>", "wage": 120}
  ],
  "processing": {
    "enable_scraping": true,
    "enable_fact_extraction": true,
    "language": "pl"
  }
}
```

Response:

```
{
  "job_uuid": "a1b2c3d4-e5f6-7890-abcd-ef1234567890"
}
```

8. Pipeline Przetwarzania

System wykonuje 5-etapowy pipeline przetwarzania:



8.1 Szczegóły Kroków

| Krok | Nazwa | Opis | LLM |
|------|-----------------------|---|-----|
| 1 | Content Conversion | Konwersja PDF/DOCX/URL → Markdown | ✗ |
| 2 | Fact Extraction | Ekstrakcja kluczowych faktów | ✓ |
| 3 | Prediction Extraction | Generowanie predykcji na podstawie faktów | ✓ |

| Krok | Nazwa | Opis | LLM |
|------|--------------------|---|-------------------------------------|
| 4 | Unknown Extraction | Identyfikacja brakujących informacji | <input checked="" type="checkbox"/> |
| 5 | Report Generation | Tworzenie raportu JSON ze scenariuszami | <input checked="" type="checkbox"/> |

9. Struktura Raportu

System generuje raport w formacie JSON:

```
{  
  "summary": "Streszczenie danych (max 150 słów) – przejrzyste, user-friendly",  
  "positive_scenario": "Scenariusz pozytywny dla Atlantis z korelacjami (200-300 słów)",  
  "negative_scenario": "Scenariusz negatywny dla Atlantis z korelacjami (200-300 słów)",  
  "recommendations": "Rekomendacje decyzyjne do uniknięcia scenariuszy negatywnych"  
}
```

10. Konfiguracja i Deployment

10.1 Zmienne Środowiskowe

```
# Database  
DB_HOST=database  
DB_PORT=5432  
DB_NAME=hacknation  
DB_USER=postgres  

```

```

LLM_PROVIDER=cloudflare

# Cloudflare Workers AI
CLOUDFLARE_ACCOUNT_ID=<your-account-id>
CLOUDFLARE_API_TOKEN=<your-api-token>
CLOUDFLARE_MODEL_EN=@cf/meta/llama-3.1-70b-instruct
CLOUDFLARE_MODEL_PL=@cf/meta/llama-3.1-70b-instruct

# Ollama (alternatywnie)
OLLAMA_HOST=host.docker.internal
OLLAMA_PORT=11434
OLLAMA_MODEL=qwen3:30b-a3b

```

10.2 Uruchomienie

```

# Produkcja (z Cloudflare AI)
docker-compose up -d

# Produkcja (CPU-only z Ollama)
docker-compose -f docker-compose.cpu.yml up -d

# Development
./start.sh

# Health check
./health-check.sh

```

10.3 Wymagania Sprzętowe

| Komponent | Minimum | Zalecane |
|-----------|-------------|---------------------|
| RAM | 8 GB | 16 GB |
| Dysk | 20 GB | 50 GB |
| CPU | 4 cores | 8 cores |
| GPU | Opcjonalnie | NVIDIA (dla Ollama) |

11. Bezpieczeństwo

| Aspekt | Status | Opis |
|---------------------|--------|------------------------------|
| CORS | ✓ | Skonfigurowane dla frontendu |
| Flask Debug | ✓ | Wyłączone w produkcji |
| Secrets | ✓ | Zmienne środowiskowe |
| Container Isolation | ✓ | Docker network isolation |
| Input Validation | ✓ | Walidacja requestów API |

12. Analiza Kosztów LLM

12.1 Cloudflare Workers AI

| Metryka | Wartość |
|--------------------|------------------------|
| Koszt 10k neuronów | ~\$0.11 |
| Model | Llama 3.1 70B Instruct |
| Latency | ~2-5s per request |

12.2 Self-Hosted (Ollama)

| Konfiguracja | Koszt miesięczny* |
|------------------|-------------------|
| GPU Cloud (A100) | ~\$1000-2000 |
| GPU Cloud (A10G) | ~\$300-600 |

| Konfiguracja | Koszt miesięczny* |
|-----------------------|---------------------|
| On-premise (RTX 4090) | Jednorazowo ~\$2000 |

*Szacunki dla intensywnego użycia

12.3 Rekomendacja

Dla małej skali (hackathon, MVP): **Cloudflare Workers AI** – brak upfront costs, pay-per-use.

Dla produkcji na większą skalę: **Self-hosted Ollama** – przewidywalny koszt, pełna kontrola.

13. Rozszerzalność

13.1 Łatwa Podmiana LLM

System obsługuje prostą podmianę providera LLM poprzez zmienną środowiskową
`LLM_PROVIDER : - cloudflare` – Cloudflare Workers AI (domyślnie) - `ollama` – Lokalny/self-hosted Ollama

13.2 Dodawanie Nowych Formatów

Dzięki bibliotece MarkItDown dodanie obsługi nowych formatów wymaga minimalnych zmian w kodzie.

13.3 Graf Wiedzy

System wykorzystuje strukturę grafową (nodes + relations) umożliwiającą: - Wizualizację powiązań między faktami - Śledzenie pochodzenia predykcji - Identyfikację konfliktów informacyjnych

14. Podsumowanie

S.I.K.O.R.K.A. (System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz) to w pełni funkcjonalny system analizy strategicznej wykorzystujący najnowsze osiągnięcia w dziedzinie LLM.

Kluczowe zalety:

| # | Zaleta | Opis |
|---|--------------------------------|---|
| 1 | 100% Open Source | Brak zależności od zamkniętych rozwiązań |
| 2 | Pełna konteneryzacja | Gotowy do wdrożenia w dowolnej infrastrukturze Docker |
| 3 | Elastyczny provider LLM | Łatwa migracja między Cloudflare a self-hosted |
| 4 | Niskie koszty | ~\$0.11 za intensywną sesję testową |
| 5 | Wieloformatowy input | PDF, DOCX, PPTX, strony dynamiczne (JS) |
| 6 | Kompleksowy output | Fakty, predykcje, brakujące info, raport ze scenariuszami |

Dokumentacja wygenerowana: Grudzień 2025 **S.I.K.O.R.K.A.** – Zespół "Wifi nie działa"

S.I.K.O.R.K.A.

System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz



Specyfikacja Techniczna

Wersja 1.0

Data Grudzień 2025

Zespół Wifi nie działa

Hackathon Hacknation/Scenariusze Jutra

„Mała, ale inteligentna – jak sikorka, która szybko przetwarza informacje z otoczenia”

Spis Treści

1. Wprowadzenie
 1. Nazwa Projektu
 2. Cel Projektu
 3. Potencjał Wdrożeniowy
2. Architektura Systemu
 1. Diagram C4 – Poziom Kontekstu
 2. Diagram C4 – Poziom Kontenerów
 3. Diagram C4 – Poziom Komponentów
3. Diagram Sekwencji – Submit Job Flow
4. Obsługiwane Formaty Danych
 1. Dokumenty
 2. Źródła Webowe
5. Stos Technologiczny
 1. Frontend
 2. Backend
 3. Baza Danych
 4. LLM & AI
6. Model Danych
 1. Diagram ERD
 2. Typy Węzłów
 3. Typy Relacji
7. API Endpoints
 1. Główne Endpointy
 2. Przykład Request/Response

8. Pipeline Przetwarzania

1. Szczegóły Kroków

9. Struktura Raportu

10. Konfiguracja i Deployment

1. Zmienne Środowiskowe

2. Uruchomienie

3. Wymagania Sprzętowe

11. Bezpieczeństwo

12. Analiza Kosztów LLM

1. Cloudflare Workers AI

2. Self-Hosted (Ollama)

3. Rekomendacja

13. Rozszerzalność

1. Łatwa Podmiana LLM

2. Dodawanie Nowych Formatów

3. Graf Wiedzy

14. Podsumowanie

1. Wprowadzenie

1.1 Nazwa Projektu

S.I.K.O.R.K.A. – System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz

Nazwa projektu nawiązuje do ptaka sikorki – małego, ale niezwykle inteligentnego i adaptacyjnego stworzenia, które potrafi szybko przetwarzać informacje z otoczenia i dostosowywać swoje zachowanie. Podobnie nasz system szybko przetwarza różnorodne źródła danych i adaptuje się do potrzeb analityka.

1.2 Cel Projektu

System **S.I.K.O.R.K.A.** to narzędzie analityczne wykorzystujące modele LLM (Large Language Models) do automatycznej ekstrakcji faktów, generowania predykcji i tworzenia raportów analitycznych dla fikcyjnego państwa **Atlantis**. System wspiera proces podejmowania decyzji strategicznych poprzez:

- Automatyczną ekstrakcję kluczowych faktów z różnorodnych źródeł
- Identyfikację potencjalnych scenariuszy (pozytywnych i negatywnych)
- Wykrywanie brakujących informacji (unknowns)
- Generowanie kompleksowych raportów z rekomendacjami

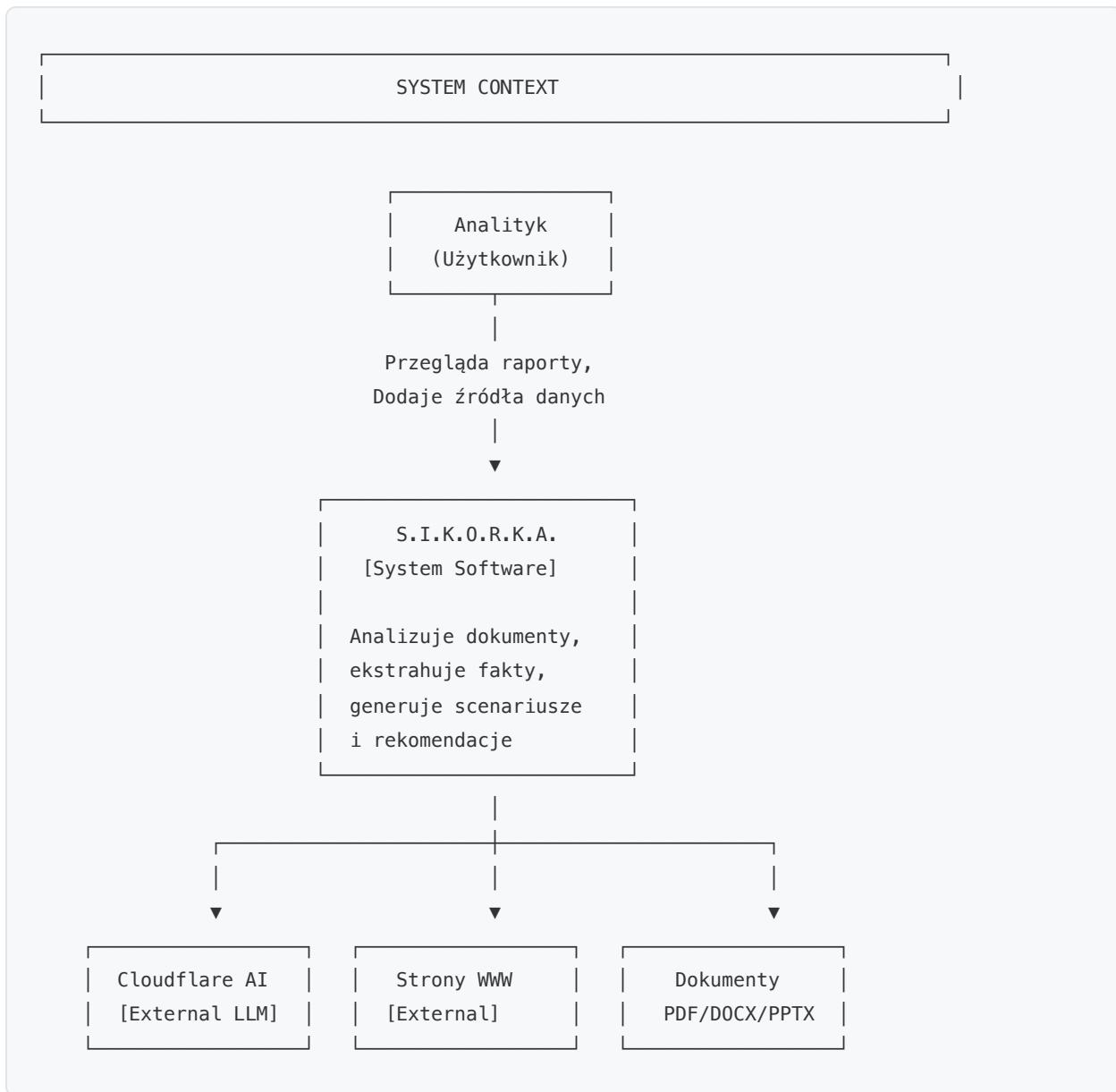
1.3 Potencjał Wdrożeniowy

| Aspekt | Opis |
|-----------------------|---|
| Konteneryzacja | Pełna konteneryzacja Docker – system działa na dowolnej infrastrukturze wspierającej Docker |

| Aspekt | Opis |
|---------------------------|---|
| Modele Open Source | Wykorzystanie wyłącznie modeli open-source (Llama 3.1, Qwen3) – brak vendor lock-in |
| Elastyczność LLM | Prosty mechanizm podmiany providera LLM (zmienna <code>LLM_PROVIDER</code>) |
| Koszty | Podczas intensywnych testów zużyto ~10k neuronów Cloudflare (~\$0.11) |
| Self-hosting | Możliwość hostowania własnego LLM (Ollama) z określonym, skalowalnym kosztem |
| Skalowalność | Architektura mikroserwisowa umożliwia niezależne skalowanie komponentów |

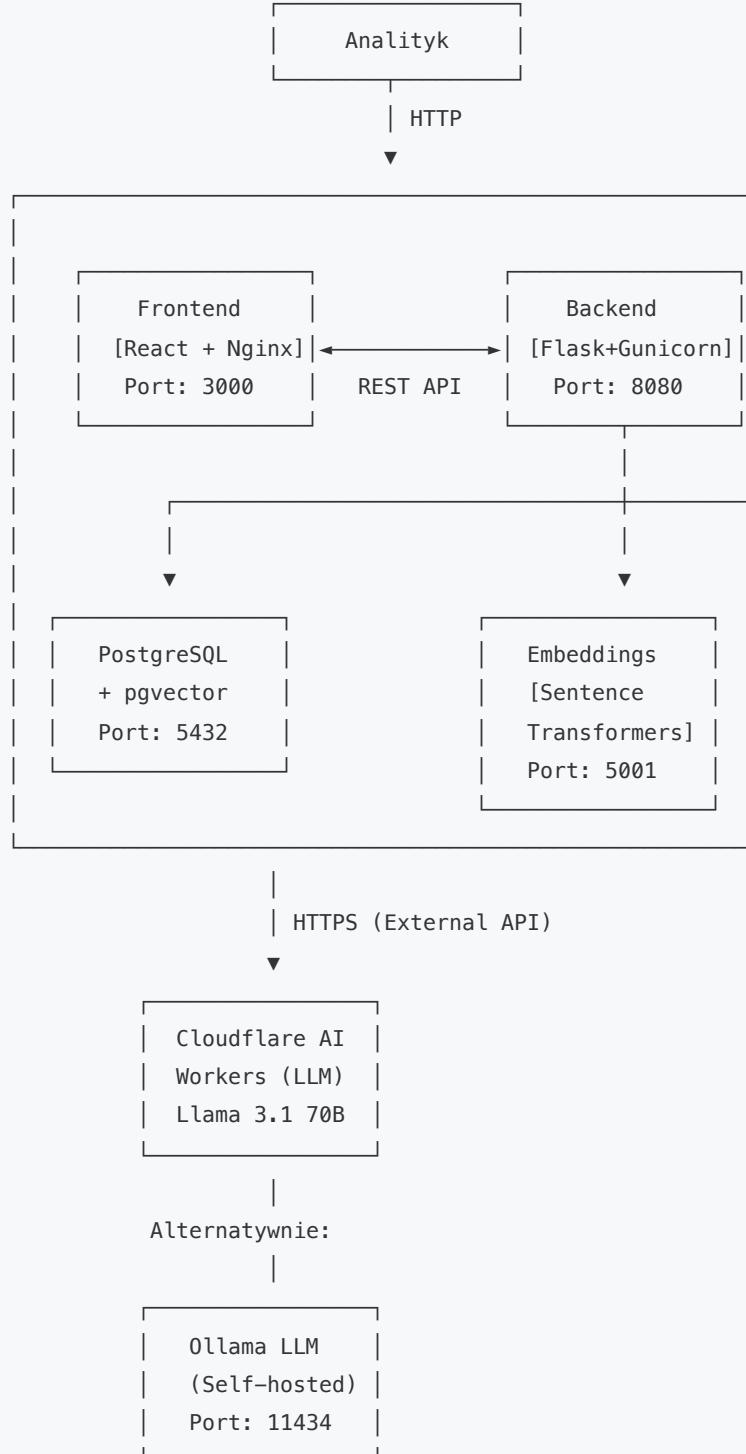
2. Architektura Systemu

2.1 Diagram C4 – Poziom Kontekstu

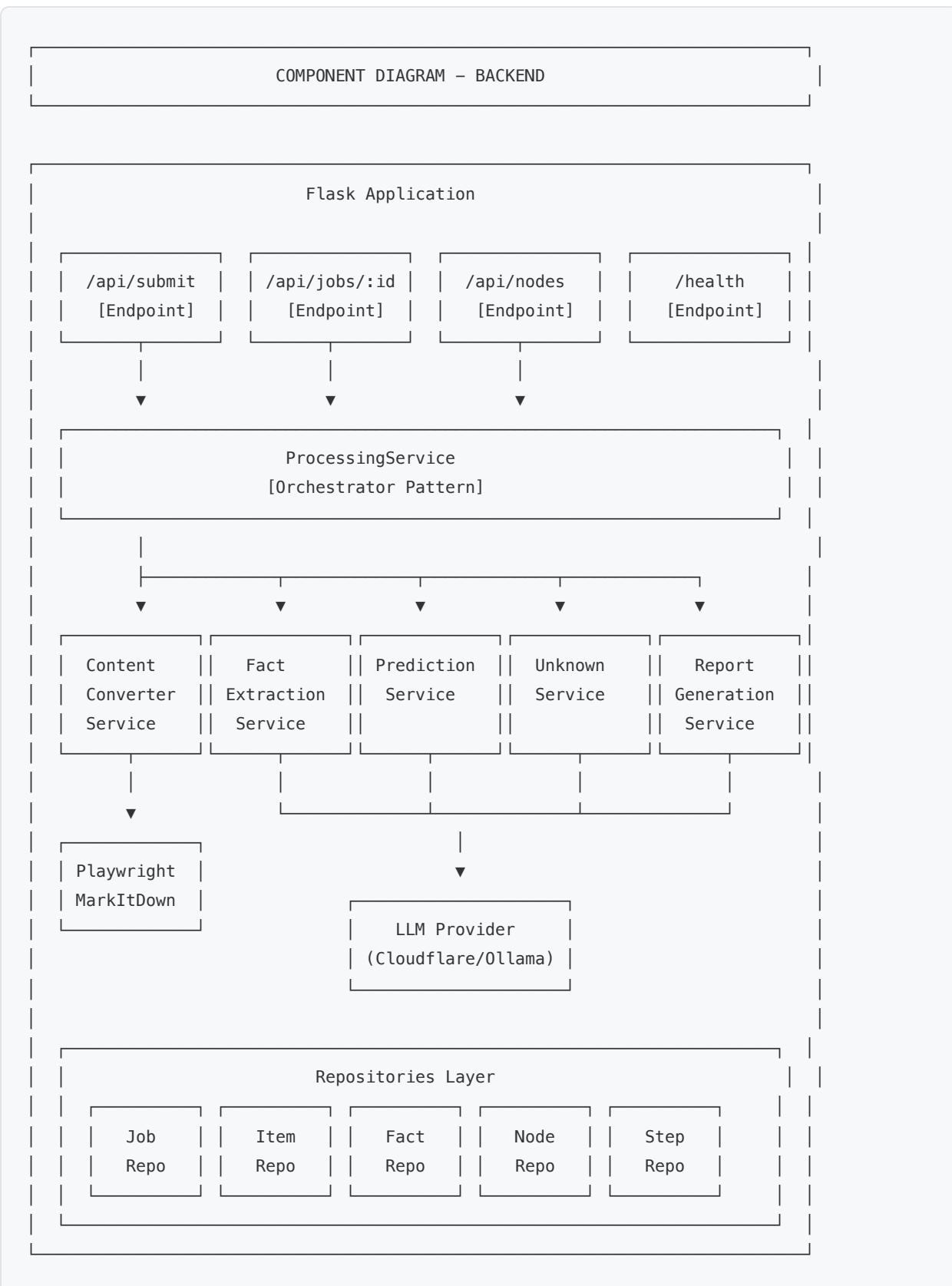


2.2 Diagram C4 – Poziom Kontenerów

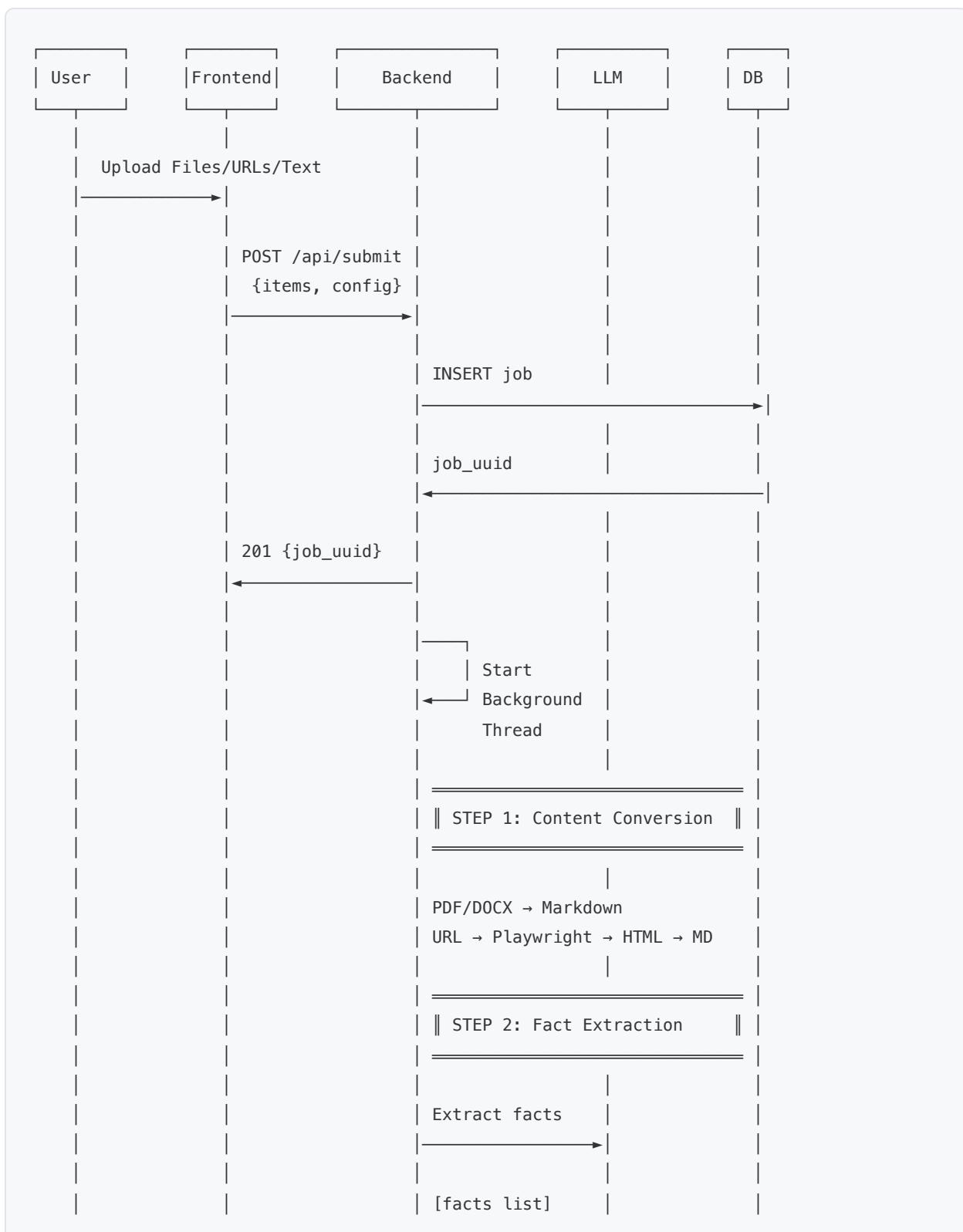
CONTAINER DIAGRAM
Docker Network: hacknation-network

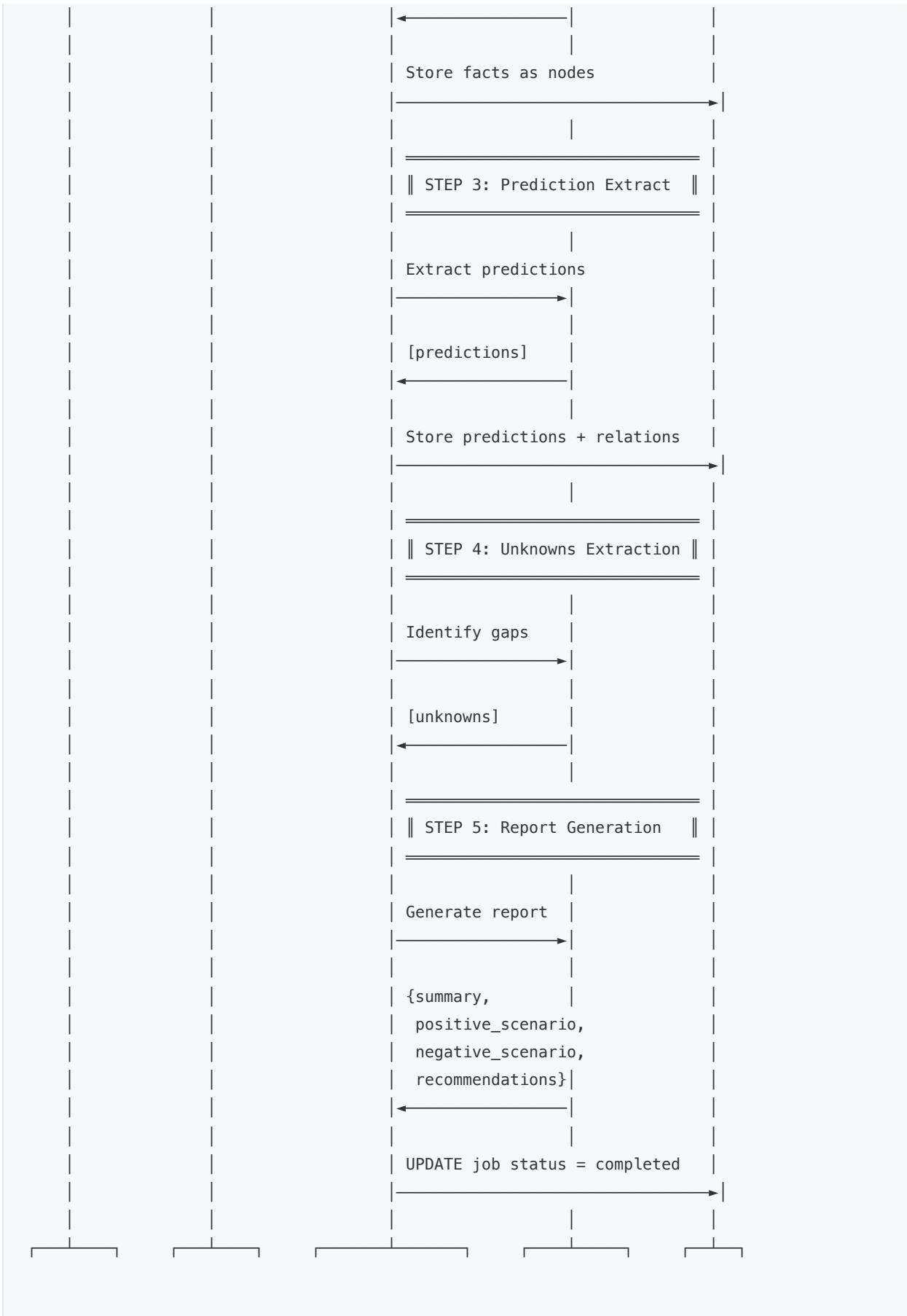


2.3 Diagram C4 – Poziom Komponentów (Backend)



3. Diagram Sekwencji – Submit Job Flow







4. Obsługiwane Formaty Danych

System akceptuje różnorodne formaty wejściowe:

4.1 Dokumenty

| Format | Rozszerzenie | Biblioteka konwersji |
|------------|--------------|----------------------|
| PDF | .pdf | MarkItDown |
| Word | .docx | MarkItDown |
| PowerPoint | .pptx | MarkItDown |
| Excel | .xlsx | MarkItDown |
| Tekst | plain text | Native |

4.2 Źródła Webowe

| Typ | Obsługa | Technologia |
|---------------------------|-------------------------------------|--|
| Strony statyczne | <input checked="" type="checkbox"/> | requests + BeautifulSoup |
| Strony dynamiczne (JS) | <input checked="" type="checkbox"/> | Playwright – czeka na pełne załadowanie JS |
| SPA (Single Page Apps) | <input checked="" type="checkbox"/> | Playwright z <code>wait_until='networkidle'</code> |

Kluczowa funkcjonalność: System wykorzystuje Playwright do renderowania stron dynamicznych. Po nawigacji do URL czeka na zakończenie aktywności sieciowej (`networkidle`), co gwarantuje pobranie treści generowanych przez JavaScript.

5. Stos Technologiczny

5.1 Frontend

| Technologia | Wersja | Rola |
|--------------|--------|-------------------|
| React | 18.x | Framework UI |
| Vite | 6.x | Build tool |
| React Router | 6.x | Routing |
| Tailwind CSS | 3.x | Stylowanie |
| Lucide React | - | Ikony |
| Nginx | latest | Production server |

5.2 Backend

| Technologia | Wersja | Rola |
|-------------|--------|-------------------------|
| Python | 3.11+ | Runtime |
| Flask | 3.0.0 | Framework API |
| Gunicorn | 22.0.0 | WSGI Server (4 workers) |
| psycopg2 | 2.9.9 | PostgreSQL driver |
| pgvector | 0.2.4 | Vector operations |
| Playwright | latest | Web scraping dynamiczny |

| Technologia | Wersja | Rola |
|----------------|--------|----------------------|
| MarkItDown | latest | Konwersja dokumentów |
| BeautifulSoup4 | 4.12.2 | HTML parsing |

5.3 Baza Danych

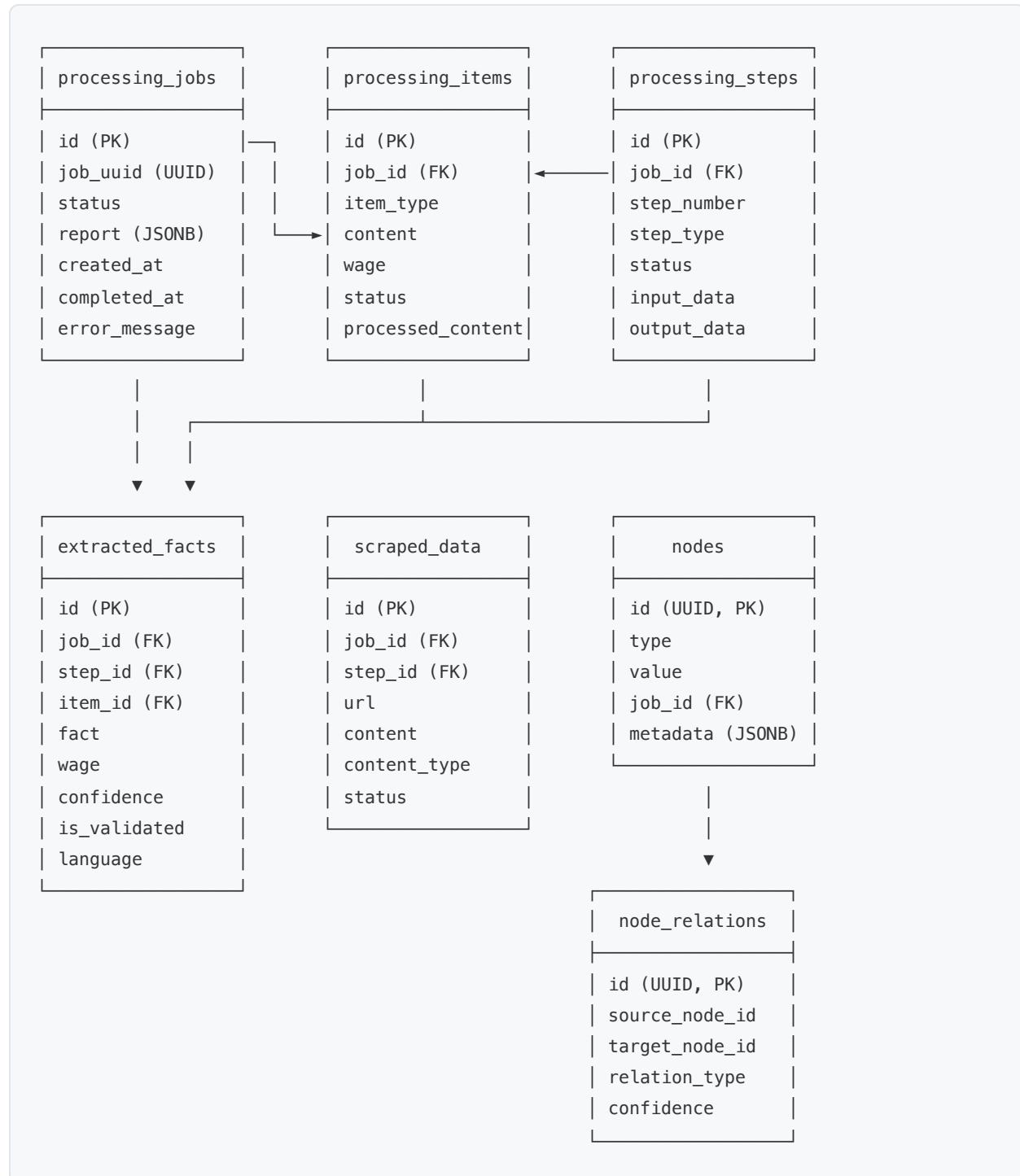
| Technologia | Wersja | Rola |
|-------------|-----------|------------------------|
| PostgreSQL | 16 | Główna baza danych |
| pgvector | extension | Wyszukiwanie wektorowe |

5.4 LLM & AI

| Provider | Model | Użycie |
|-----------------------|------------------------|-----------------------------|
| Cloudflare Workers AI | Llama 3.1 70B Instruct | Główny provider (hostowany) |
| Ollama | Qwen3:30b-a3b / Llama2 | Alternatywny (self-hosted) |
| Sentence Transformers | MiniLM-L12-v2 | Embeddingi multilingualne |

6. Model Danych

6.1 Diagram ERD



6.2 Typy Węzłów (Nodes)

| Typ | Opis |
|---------------------|--|
| fact | Wyekstrahowany fakt z dokumentu |
| prediction | Predykcja/prognoza wynikająca z faktów |
| missing_information | Zidentyfikowana luka informacyjna |
| report | Wygenerowany raport końcowy |

6.3 Typy Relacji

| Typ | Opis |
|--------------|--|
| derived_from | Węzeł wynika z innego węzła |
| supports | Węzeł wspiera/potwierdza inny węzeł |
| contradicts | Węzeł jest sprzeczny z innym węzłem |
| requires | Węzeł wymaga informacji z innego węzła |
| suggests | Węzeł sugeruje/implikuje inny węzeł |

7. API Endpoints

7.1 Główne Endpointy

| Metoda | Endpoint | Opis |
|--------|---------------------------|---|
| GET | /health | Health check |
| POST | /api/submit | Utworzenie nowego zadania analizy |
| GET | /api/jobs | Lista wszystkich zadań |
| GET | /api/jobs/{uuid} | Szczegóły zadania z faktami, predykcjami i raportem |
| GET | /api/jobs/{uuid}/nodes | Węzły powiązane z zadaniem |
| GET | /api/nodes/{id} | Szczegóły pojedynczego węzła |
| GET | /api/nodes/{id}/relations | Relacje węzła |

7.2 Przykład Request/Response

POST /api/submit

```
{
  "items": [
    {"type": "text", "content": "Treść do analizy", "wage": 100},
    {"type": "link", "content": "https://example.com/article", "wage": 80},
    {"type": "file", "content": "<base64>", "wage": 120}
  ],
  "processing": {
    "enable_scraping": true,
    "enable_fact_extraction": true,
    "language": "pl"
  }
}
```

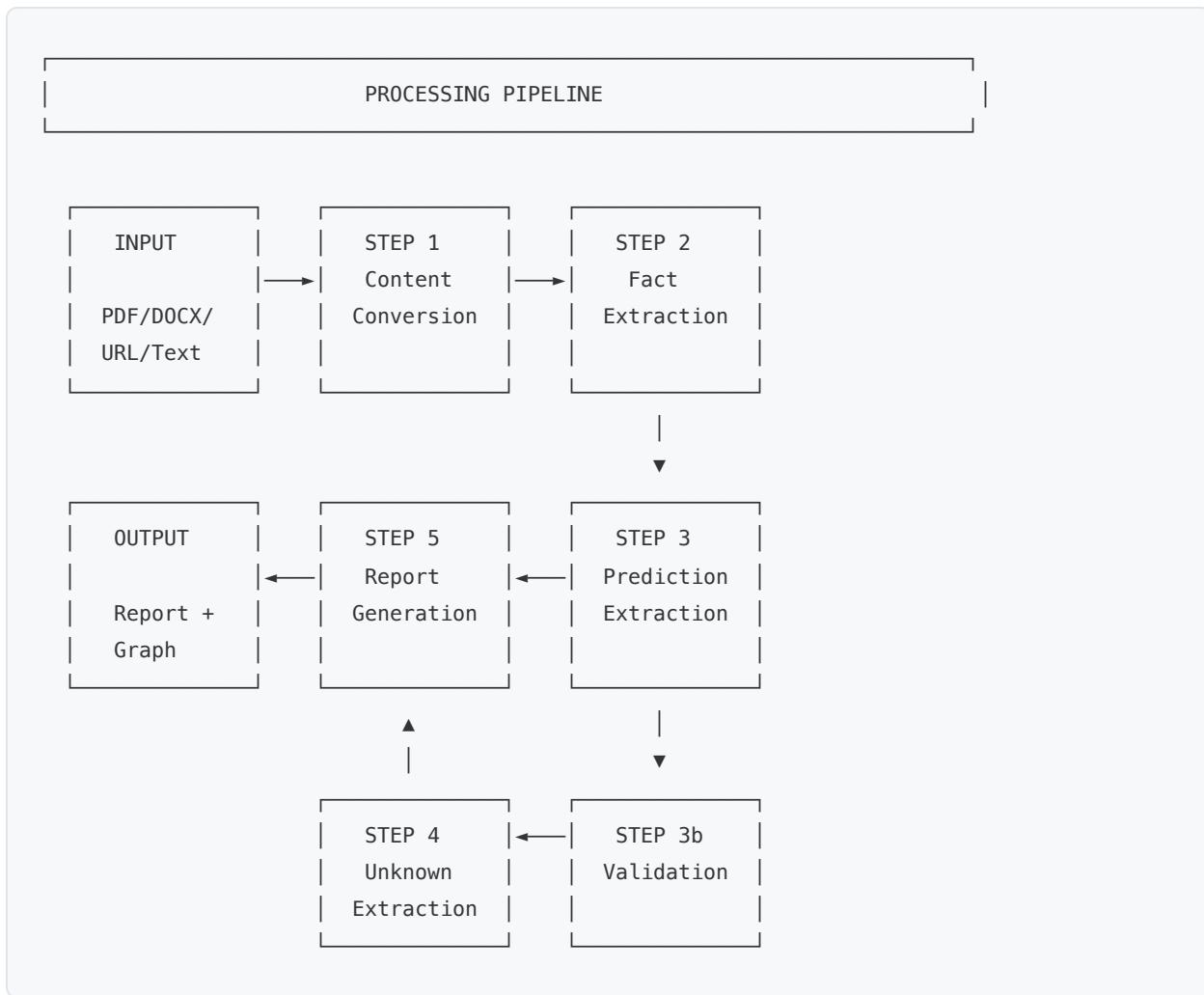
```
 }  
 }
```

Response:

```
{  
  "job_uuid": "a1b2c3d4-e5f6-7890-abcd-ef1234567890"  
}
```

8. Pipeline Przetwarzania

System wykonuje 5-etapowy pipeline przetwarzania:



8.1 Szczegóły Kroków

| Krok | Nazwa | Opis | LLM |
|------|--------------------|-----------------------------------|-----|
| 1 | Content Conversion | Konwersja PDF/DOCX/URL → Markdown | ✗ |
| 2 | Fact Extraction | Ekstrakcja kluczowych faktów | ✓ |

| Krok | Nazwa | Opis | LLM |
|------|-----------------------|---|-----|
| 3 | Prediction Extraction | Generowanie predykcji na podstawie faktów | ✓ |
| 4 | Unknown Extraction | Identyfikacja brakujących informacji | ✓ |
| 5 | Report Generation | Tworzenie raportu JSON ze scenariuszami | ✓ |

9. Struktura Raportu

System generuje raport w formacie JSON:

```
{  
  "summary": "Streszczenie danych (max 150 słów) – przejrzyste, user-friendly",  
  "positive_scenario": "Scenariusz pozytywny dla Atlantis z korelacjami (200–300 słów)",  
  "negative_scenario": "Scenariusz negatywny dla Atlantis z korelacjami (200–300 słów)",  
  "recommendations": "Rekomendacje decyzyjne do uniknięcia scenariuszy negatywnych"  
}
```

10. Konfiguracja i Deployment

10.1 Zmienne Środowiskowe

```
# Database
DB_HOST=database
DB_PORT=5432
DB_NAME=hacknation
DB_USER=postgres
DB_PASSWORD=postgres

# LLM Provider: 'cloudflare' lub 'ollama'
LLM_PROVIDER=cloudflare

# Cloudflare Workers AI
CLOUDFLARE_ACCOUNT_ID=<your-account-id>
CLOUDFLARE_API_TOKEN=<your-api-token>
CLOUDFLARE_MODEL_EN=@cf/meta/llama-3.1-70b-instruct
CLOUDFLARE_MODEL_PL=@cf/meta/llama-3.1-70b-instruct

# Ollama (alternatywnie)
OLLAMA_HOST=host.docker.internal
OLLAMA_PORT=11434
OLLAMA_MODEL=qwen3:30b-a3b
```

10.2 Uruchomienie

```
# Produkcja (z Cloudflare AI)
docker-compose up -d

# Produkcja (CPU-only z Ollama)
docker-compose -f docker-compose.cpu.yml up -d

# Development
./start.sh
```

```
# Health check  
./health-check.sh
```

10.3 Wymagania Sprzętowe

| Komponent | Minimum | Zalecane |
|-----------|-------------|---------------------|
| RAM | 8 GB | 16 GB |
| Dysk | 20 GB | 50 GB |
| CPU | 4 cores | 8 cores |
| GPU | Opcjonalnie | NVIDIA (dla Ollama) |

11. Bezpieczeństwo

| Aspekt | Status | Opis |
|---------------------|-------------------------------------|------------------------------|
| CORS | <input checked="" type="checkbox"/> | Skonfigurowane dla frontendu |
| Flask Debug | <input checked="" type="checkbox"/> | Wyłączone w produkcji |
| Secrets | <input checked="" type="checkbox"/> | Zmienne środowiskowe |
| Container Isolation | <input checked="" type="checkbox"/> | Docker network isolation |
| Input Validation | <input checked="" type="checkbox"/> | Walidacja requestów API |

12. Analiza Kosztów LLM

12.1 Cloudflare Workers AI

| Metryka | Wartość |
|--------------------|------------------------|
| Koszt 10k neuronów | ~\$0.11 |
| Model | Llama 3.1 70B Instruct |
| Latency | ~2-5s per request |

12.2 Self-Hosted (Ollama)

| Konfiguracja | Koszt miesięczny* |
|-----------------------|---------------------|
| GPU Cloud (A100) | ~\$1000-2000 |
| GPU Cloud (A10G) | ~\$300-600 |
| On-premise (RTX 4090) | Jednorazowo ~\$2000 |

*Szacunki dla intensywnego użycia

12.3 Rekomendacja

Dla małej skali (hackathon, MVP): **Cloudflare Workers AI** – brak upfront costs, pay-per-use.

Dla produkcji na większą skalę: **Self-hosted Ollama** – przewidywalny koszt, pełna kontrola.

13. Rozszerzalność

13.1 Łatwa Podmiana LLM

System obsługuje prostą podmianę providera LLM poprzez zmienną środowiskową
`LLM_PROVIDER` : - `cloudflare` – Cloudflare Workers AI (domyślnie) - `ollama` –
Lokalny/self-hosted Ollama

13.2 Dodawanie Nowych Formatów

Dzięki bibliotece MarkItDown dodanie obsługi nowych formatów wymaga minimalnych zmian w kodzie.

13.3 Graf Wiedzy

System wykorzystuje strukturę grafową (nodes + relations) umożliwiającą: - Wizualizację powiązań między faktami - Śledzenie pochodzenia predykcji - Identyfikację konfliktów informacyjnych

14. Podsumowanie

S.I.K.O.R.K.A. (System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz) to w pełni funkcjonalny system analizy strategicznej wykorzystujący najnowsze osiągnięcia w dziedzinie LLM.

Kluczowe zalety:

| # | Zaleta | Opis |
|---|--------------------------------|---|
| 1 | 100% Open Source | Brak zależności od zamkniętych rozwiązań |
| 2 | Pełna konteneryzacja | Gotowy do wdrożenia w dowolnej infrastrukturze Docker |
| 3 | Elastyczny provider LLM | Łatwa migracja między Cloudflare a self-hosted |
| 4 | Niskie koszty | ~\$0.11 za intensywną sesję testową |
| 5 | Wieloformatowy input | PDF, DOCX, PPTX, strony dynamiczne (JS) |
| 6 | Kompleksowy output | Fakty, predykcje, brakujące info, raport ze scenariuszami |

Dokumentacja wygenerowana: Grudzień 2025 **S.I.K.O.R.K.A.** – Zespół "Wifi nie działa"

1. Wprowadzenie

1.1 Nazwa Projektu

S.I.K.O.R.K.A. – System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz

Nazwa projektu nawiązuje do ptaka sikorki – małego, ale niezwykle inteligentnego i adaptacyjnego stworzenia, które potrafi szybko przetwarzać informacje z otoczenia i dostosowywać swoje zachowanie. Podobnie nasz system szybko przetwarza różnorodne źródła danych i adaptuje się do potrzeb analityka.

1.2 Cel Projektu

System **S.I.K.O.R.K.A.** to narzędzie analityczne wykorzystujące modele LLM (Large Language Models) do automatycznej ekstrakcji faktów, generowania predykcji i tworzenia raportów analitycznych dla fikcyjnego państwa **Atlantis**. System wspiera proces podejmowania decyzji strategicznych poprzez:

- Automatyczną ekstrakcję kluczowych faktów z różnorodnych źródeł
- Identyfikację potencjalnych scenariuszy (pozytywnych i negatywnych)
- Wykrywanie brakujących informacji (unknowns)
- Generowanie kompleksowych raportów z rekomendacjami

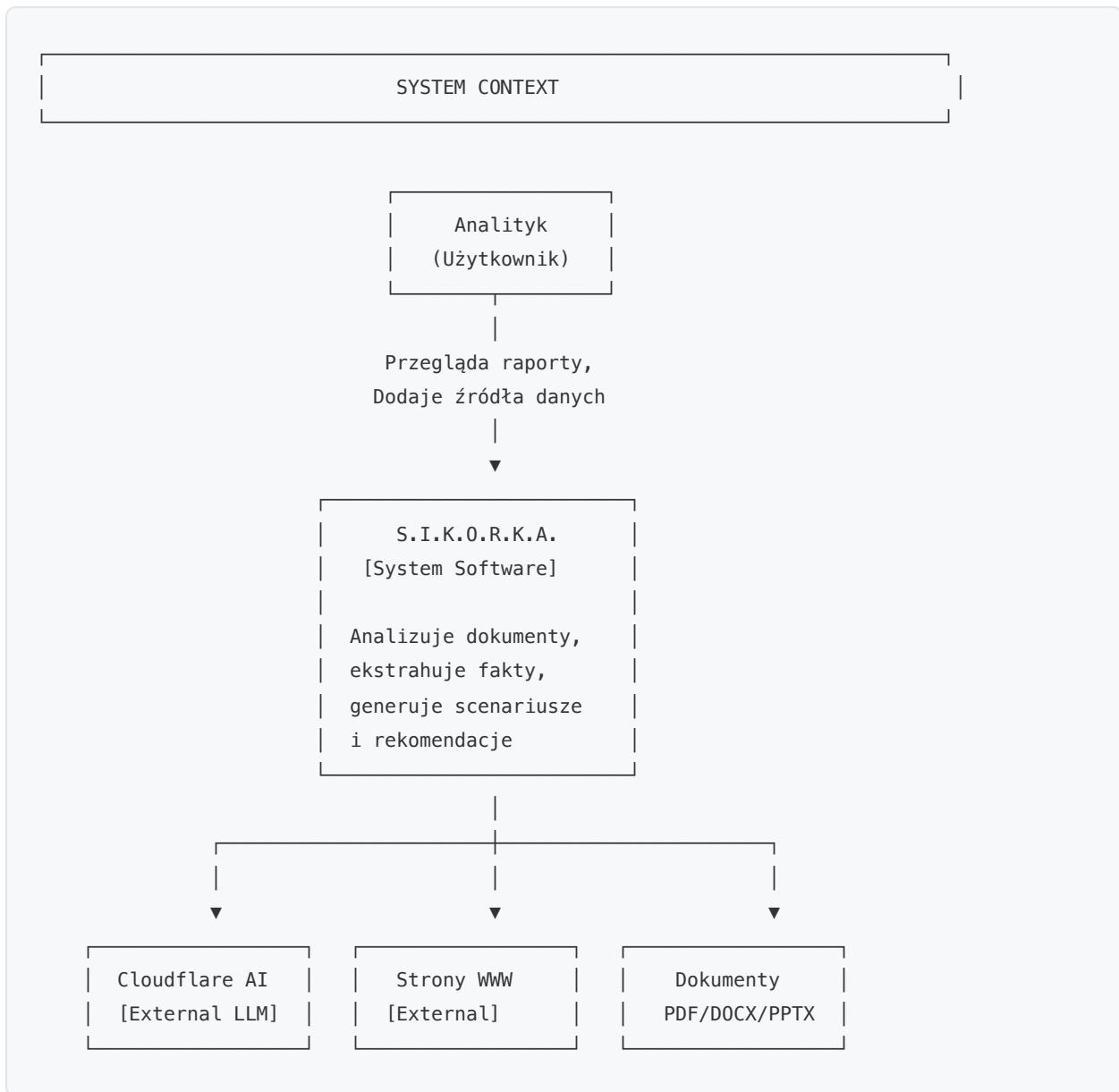
1.3 Potencjał Wdrożeniowy

| Aspekt | Opis |
|-----------------------|---|
| Konteneryzacja | Pełna konteneryzacja Docker – system działa na dowolnej infrastrukturze wspierającej Docker |

| Aspekt | Opis |
|---------------------------|---|
| Modele Open Source | Wykorzystanie wyłącznie modeli open-source (Llama 3.1, Qwen3) – brak vendor lock-in |
| Elastyczność LLM | Prosty mechanizm podmiany providera LLM (zmienna <code>LLM_PROVIDER</code>) |
| Koszty | Podczas intensywnych testów zużyto ~10k neuronów Cloudflare (~\$0.11) |
| Self-hosting | Możliwość hostowania własnego LLM (Ollama) z określonym, skalowalnym kosztem |
| Skalowalność | Architektura mikroserwisowa umożliwia niezależne skalowanie komponentów |

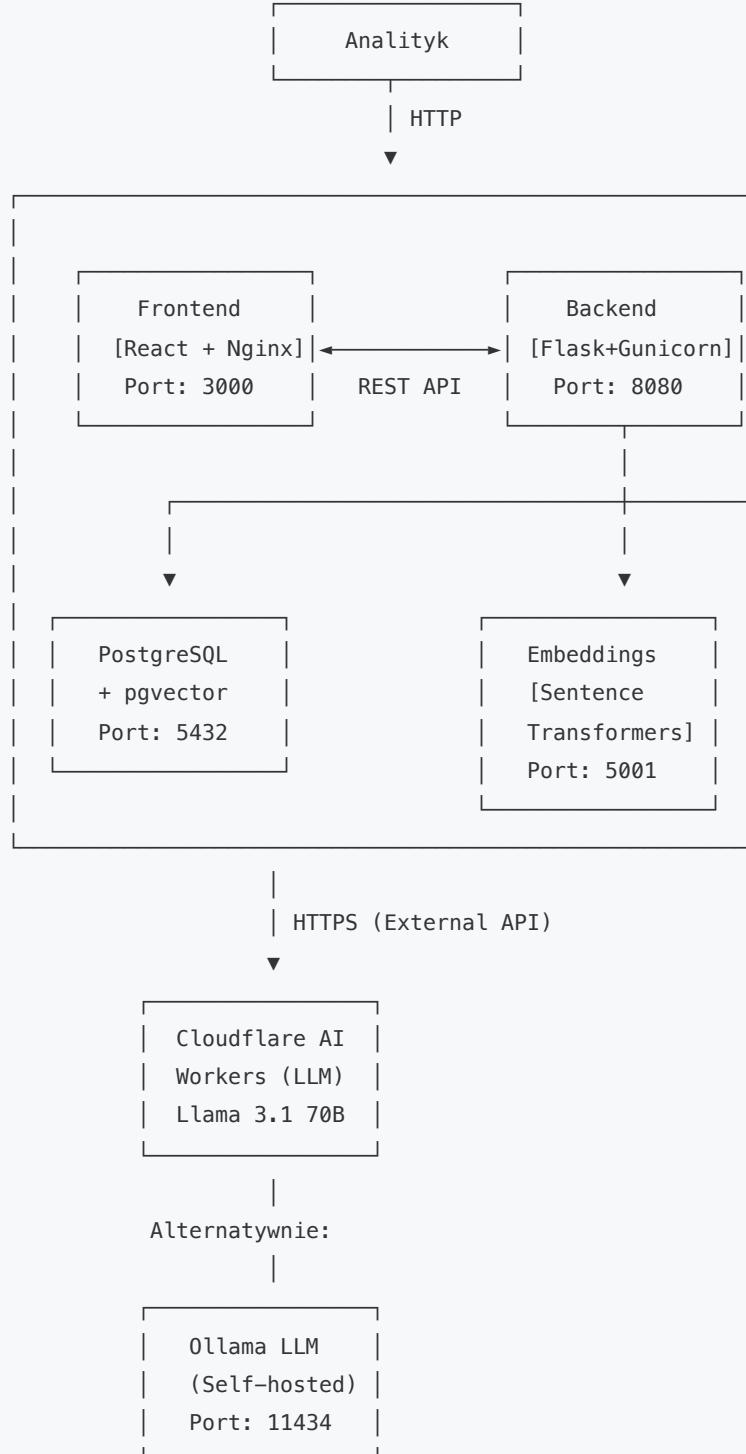
2. Architektura Systemu

2.1 Diagram C4 – Poziom Kontekstu

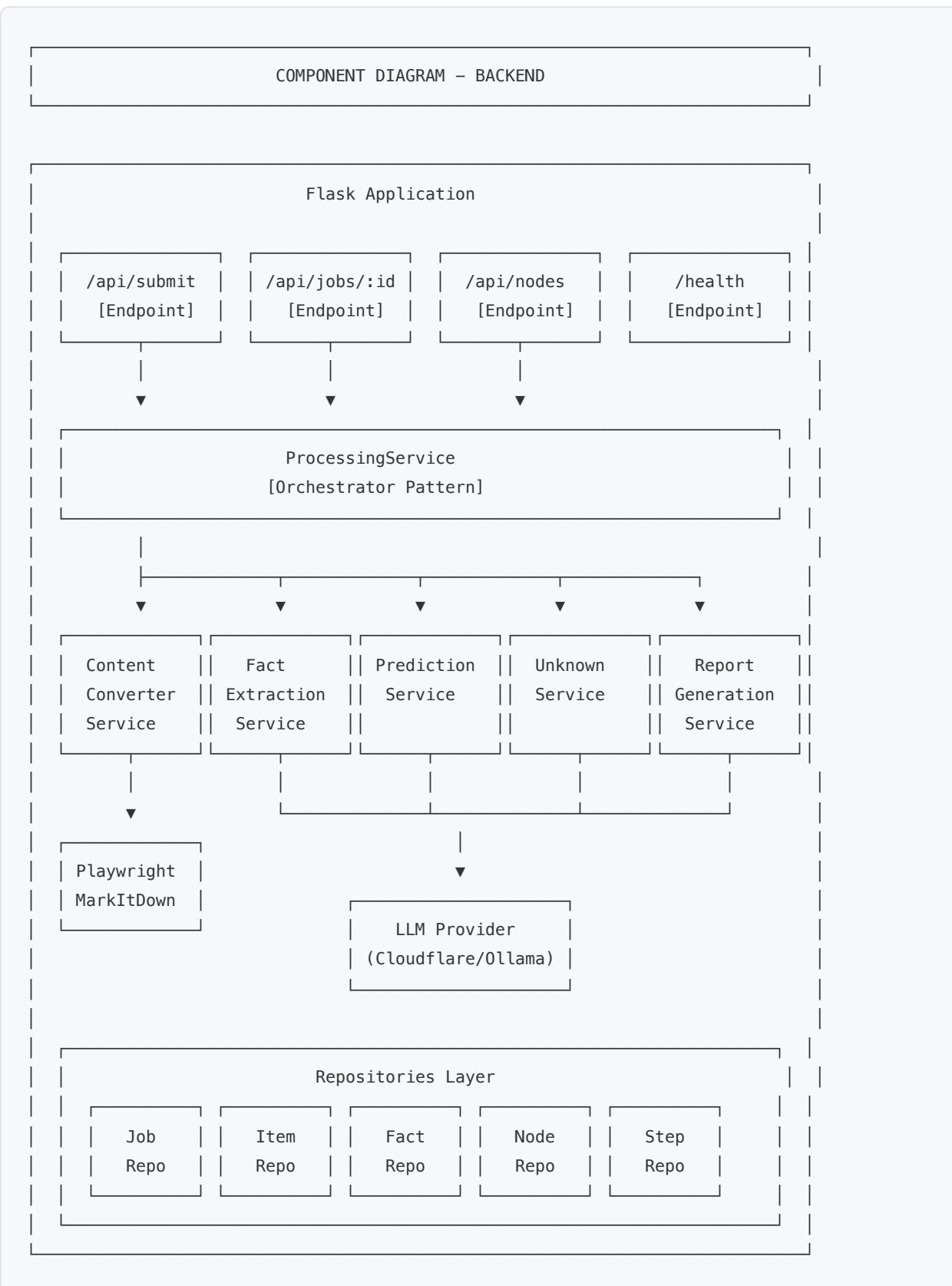


2.2 Diagram C4 – Poziom Kontenerów

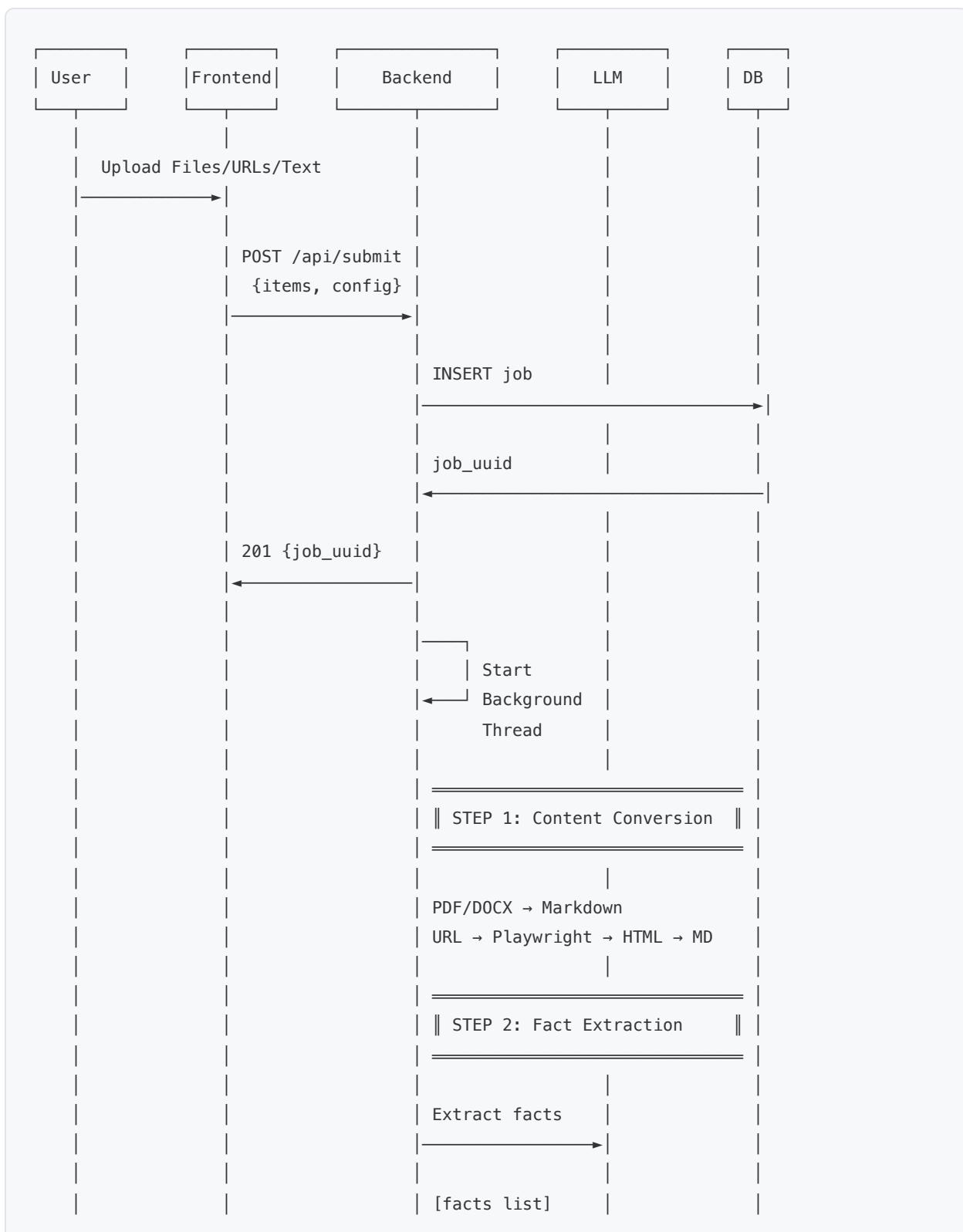
CONTAINER DIAGRAM
Docker Network: hacknation-network

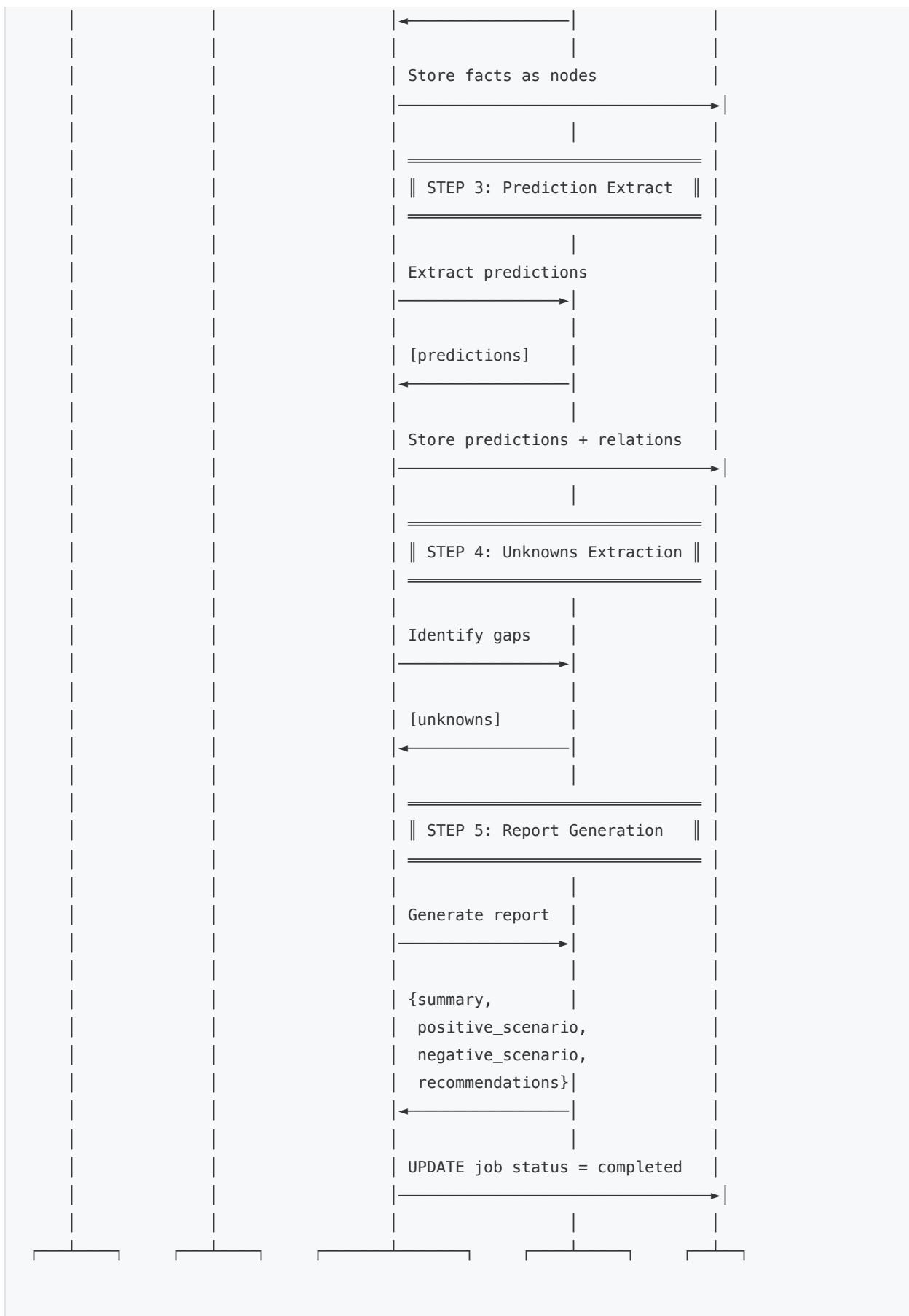


2.3 Diagram C4 – Poziom Komponentów (Backend)



3. Diagram Sekwencji – Submit Job Flow







4. Obsługiwane Formaty Danych

System akceptuje różnorodne formaty wejściowe:

4.1 Dokumenty

| Format | Rozszerzenie | Biblioteka konwersji |
|------------|--------------|----------------------|
| PDF | .pdf | MarkItDown |
| Word | .docx | MarkItDown |
| PowerPoint | .pptx | MarkItDown |
| Excel | .xlsx | MarkItDown |
| Tekst | plain text | Native |

4.2 Źródła Webowe

| Typ | Obsługa | Technologia |
|---------------------------|-------------------------------------|--|
| Strony statyczne | <input checked="" type="checkbox"/> | requests + BeautifulSoup |
| Strony dynamiczne (JS) | <input checked="" type="checkbox"/> | Playwright – czeka na pełne załadowanie JS |
| SPA (Single Page Apps) | <input checked="" type="checkbox"/> | Playwright z <code>wait_until='networkidle'</code> |

Kluczowa funkcjonalność: System wykorzystuje Playwright do renderowania stron dynamicznych. Po nawigacji do URL czeka na zakończenie aktywności sieciowej (`networkidle`), co gwarantuje pobranie treści generowanych przez JavaScript.

5. Stos Technologiczny

5.1 Frontend

| Technologia | Wersja | Rola |
|--------------|--------|-------------------|
| React | 18.x | Framework UI |
| Vite | 6.x | Build tool |
| React Router | 6.x | Routing |
| Tailwind CSS | 3.x | Stylowanie |
| Lucide React | - | Ikony |
| Nginx | latest | Production server |

5.2 Backend

| Technologia | Wersja | Rola |
|-------------|--------|-------------------------|
| Python | 3.11+ | Runtime |
| Flask | 3.0.0 | Framework API |
| Gunicorn | 22.0.0 | WSGI Server (4 workers) |
| psycopg2 | 2.9.9 | PostgreSQL driver |
| pgvector | 0.2.4 | Vector operations |
| Playwright | latest | Web scraping dynamiczny |

| Technologia | Wersja | Rola |
|----------------|--------|----------------------|
| MarkItDown | latest | Konwersja dokumentów |
| BeautifulSoup4 | 4.12.2 | HTML parsing |

5.3 Baza Danych

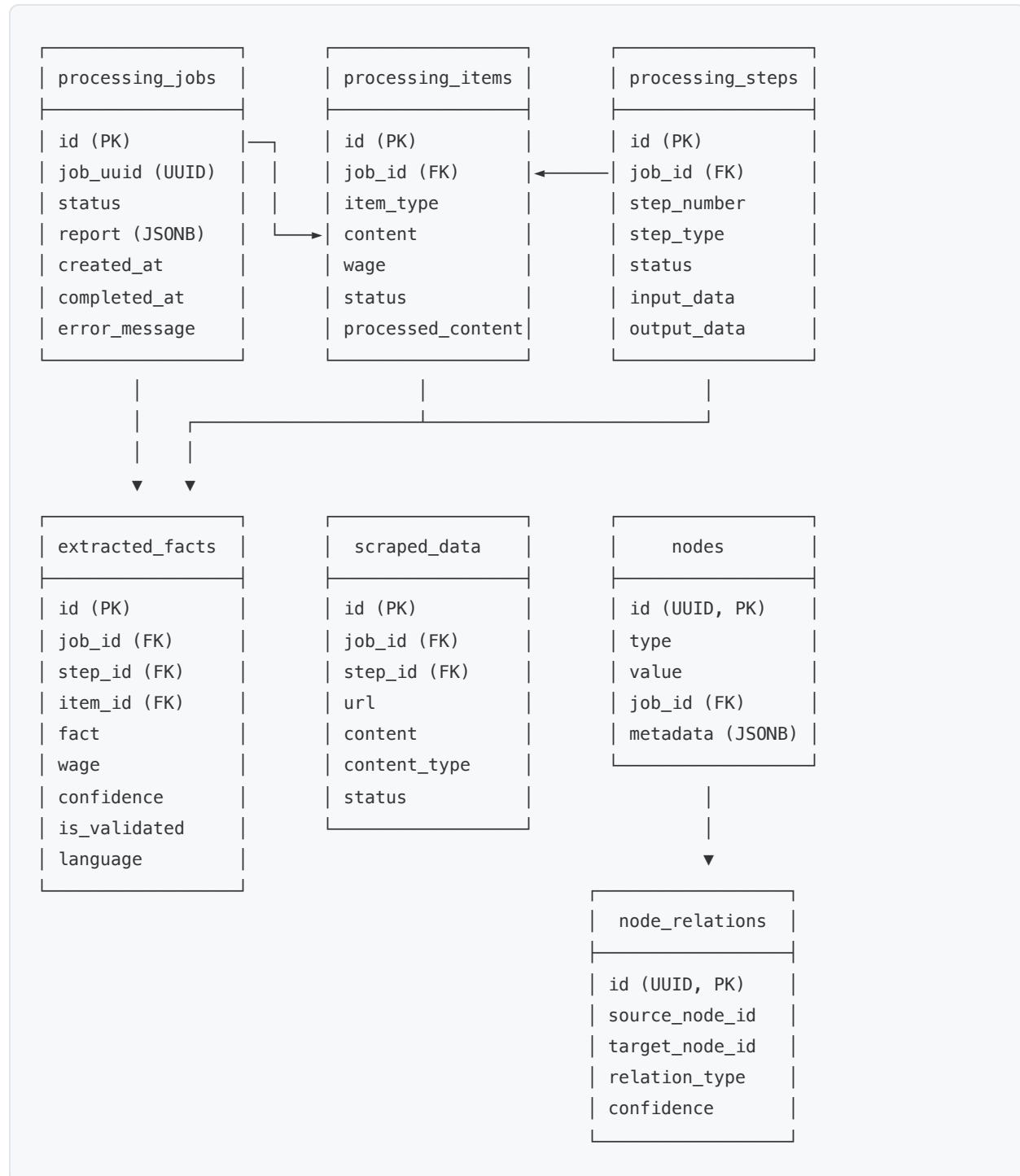
| Technologia | Wersja | Rola |
|-------------|-----------|------------------------|
| PostgreSQL | 16 | Główna baza danych |
| pgvector | extension | Wyszukiwanie wektorowe |

5.4 LLM & AI

| Provider | Model | Użycie |
|-----------------------|------------------------|-----------------------------|
| Cloudflare Workers AI | Llama 3.1 70B Instruct | Główny provider (hostowany) |
| Ollama | Qwen3:30b-a3b / Llama2 | Alternatywny (self-hosted) |
| Sentence Transformers | MiniLM-L12-v2 | Embeddingi multilingualne |

6. Model Danych

6.1 Diagram ERD



6.2 Typy Węzłów (Nodes)

| Typ | Opis |
|---------------------|--|
| fact | Wyekstrahowany fakt z dokumentu |
| prediction | Predykcja/prognoza wynikająca z faktów |
| missing_information | Zidentyfikowana luka informacyjna |
| report | Wygenerowany raport końcowy |

6.3 Typy Relacji

| Typ | Opis |
|--------------|--|
| derived_from | Węzeł wynika z innego węzła |
| supports | Węzeł wspiera/potwierdza inny węzeł |
| contradicts | Węzeł jest sprzeczny z innym węzłem |
| requires | Węzeł wymaga informacji z innego węzła |
| suggests | Węzeł sugeruje/implikuje inny węzeł |

7. API Endpoints

7.1 Główne Endpointy

| Metoda | Endpoint | Opis |
|--------|---------------------------|---|
| GET | /health | Health check |
| POST | /api/submit | Utworzenie nowego zadania analizy |
| GET | /api/jobs | Lista wszystkich zadań |
| GET | /api/jobs/{uuid} | Szczegóły zadania z faktami, predykcjami i raportem |
| GET | /api/jobs/{uuid}/nodes | Węzły powiązane z zadaniem |
| GET | /api/nodes/{id} | Szczegóły pojedynczego węzła |
| GET | /api/nodes/{id}/relations | Relacje węzła |

7.2 Przykład Request/Response

POST /api/submit

```
{  
  "items": [  
    {"type": "text", "content": "Treść do analizy", "wage": 100},  
    {"type": "link", "content": "https://example.com/article", "wage": 80},  
    {"type": "file", "content": "<base64>", "wage": 120}  
],  
  "processing": {  
    "enable_scraping": true,  
    "enable_fact_extraction": true,  
    "language": "pl"  

```

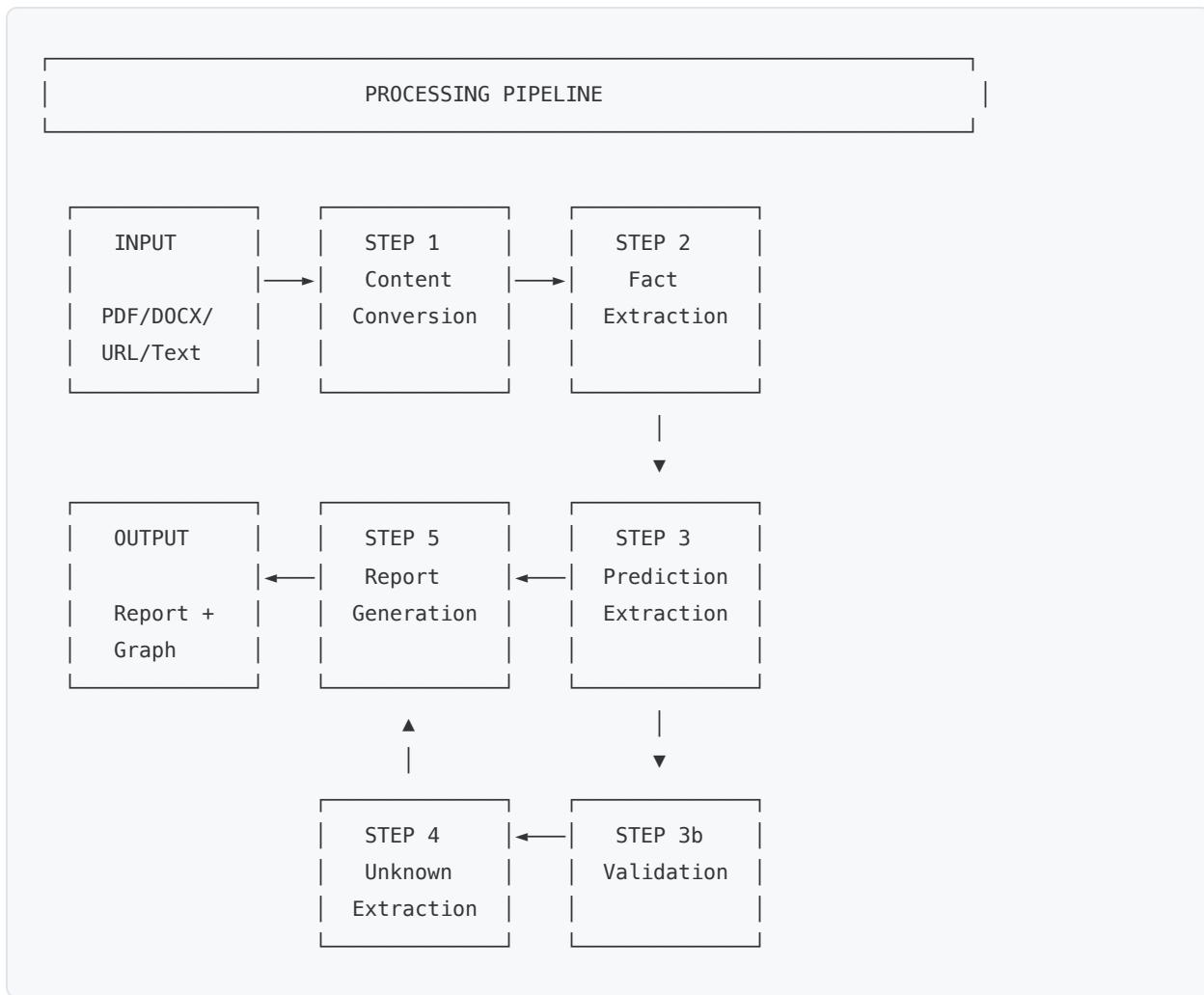
```
 }  
 }
```

Response:

```
{  
  "job_uuid": "a1b2c3d4-e5f6-7890-abcd-ef1234567890"  
}
```

8. Pipeline Przetwarzania

System wykonuje 5-etapowy pipeline przetwarzania:



8.1 Szczegóły Kroków

| Krok | Nazwa | Opis | LLM |
|------|--------------------|-----------------------------------|-----|
| 1 | Content Conversion | Konwersja PDF/DOCX/URL → Markdown | ✗ |
| 2 | Fact Extraction | Ekstrakcja kluczowych faktów | ✓ |

| Krok | Nazwa | Opis | LLM |
|------|-----------------------|---|-------------------------------------|
| 3 | Prediction Extraction | Generowanie predykcji na podstawie faktów | <input checked="" type="checkbox"/> |
| 4 | Unknown Extraction | Identyfikacja brakujących informacji | <input checked="" type="checkbox"/> |
| 5 | Report Generation | Tworzenie raportu JSON ze scenariuszami | <input checked="" type="checkbox"/> |

9. Struktura Raportu

System generuje raport w formacie JSON:

```
{  
  "summary": "Streszczenie danych (max 150 słów) – przejrzyste, user-friendly",  
  "positive_scenario": "Scenariusz pozytywny dla Atlantis z korelacjami (200–300 słów)",  
  "negative_scenario": "Scenariusz negatywny dla Atlantis z korelacjami (200–300 słów)",  
  "recommendations": "Rekomendacje decyzyjne do uniknięcia scenariuszy negatywnych"  
}
```

10. Konfiguracja i Deployment

10.1 Zmienne Środowiskowe

```
# Database
DB_HOST=database
DB_PORT=5432
DB_NAME=hacknation
DB_USER=postgres
DB_PASSWORD=postgres

# LLM Provider: 'cloudflare' lub 'ollama'
LLM_PROVIDER=cloudflare

# Cloudflare Workers AI
CLOUDFLARE_ACCOUNT_ID=<your-account-id>
CLOUDFLARE_API_TOKEN=<your-api-token>
CLOUDFLARE_MODEL_EN=@cf/meta/llama-3.1-70b-instruct
CLOUDFLARE_MODEL_PL=@cf/meta/llama-3.1-70b-instruct

# Ollama (alternatywnie)
OLLAMA_HOST=host.docker.internal
OLLAMA_PORT=11434
OLLAMA_MODEL=qwen3:30b-a3b
```

10.2 Uruchomienie

```
# Produkcja (z Cloudflare AI)
docker-compose up -d

# Produkcja (CPU-only z Ollama)
docker-compose -f docker-compose.cpu.yml up -d

# Development
./start.sh
```

```
# Health check  
./health-check.sh
```

10.3 Wymagania Sprzętowe

| Komponent | Minimum | Zalecane |
|-----------|-------------|---------------------|
| RAM | 8 GB | 16 GB |
| Dysk | 20 GB | 50 GB |
| CPU | 4 cores | 8 cores |
| GPU | Opcjonalnie | NVIDIA (dla Ollama) |

11. Bezpieczeństwo

| Aspekt | Status | Opis |
|---------------------|-------------------------------------|------------------------------|
| CORS | <input checked="" type="checkbox"/> | Skonfigurowane dla frontendu |
| Flask Debug | <input checked="" type="checkbox"/> | Wyłączone w produkcji |
| Secrets | <input checked="" type="checkbox"/> | Zmienne środowiskowe |
| Container Isolation | <input checked="" type="checkbox"/> | Docker network isolation |
| Input Validation | <input checked="" type="checkbox"/> | Walidacja requestów API |

12. Analiza Kosztów LLM

12.1 Cloudflare Workers AI

| Metryka | Wartość |
|--------------------|------------------------|
| Koszt 10k neuronów | ~\$0.11 |
| Model | Llama 3.1 70B Instruct |
| Latency | ~2-5s per request |

12.2 Self-Hosted (Ollama)

| Konfiguracja | Koszt miesięczny* |
|-----------------------|---------------------|
| GPU Cloud (A100) | ~\$1000-2000 |
| GPU Cloud (A10G) | ~\$300-600 |
| On-premise (RTX 4090) | Jednorazowo ~\$2000 |

*Szacunki dla intensywnego użycia

12.3 Rekomendacja

Dla małej skali (hackathon, MVP): **Cloudflare Workers AI** – brak upfront costs, pay-per-use.

Dla produkcji na większą skalę: **Self-hosted Ollama** – przewidywalny koszt, pełna kontrola.

13. Rozszerzalność

13.1 Łatwa Podmiana LLM

System obsługuje prostą podmianę providera LLM poprzez zmienną środowiskową
`LLM_PROVIDER` : - `cloudflare` – Cloudflare Workers AI (domyślnie) - `ollama` –
Lokalny/self-hosted Ollama

13.2 Dodawanie Nowych Formatów

Dzięki bibliotece MarkItDown dodanie obsługi nowych formatów wymaga minimalnych zmian w kodzie.

13.3 Graf Wiedzy

System wykorzystuje strukturę grafową (nodes + relations) umożliwiającą: - Wizualizację powiązań między faktami - Śledzenie pochodzenia predykcji - Identyfikację konfliktów informacyjnych

14. Podsumowanie

S.I.K.O.R.K.A. (System Inteligentnej Kompilacji Opinii, Raportów, Konfiguracji i Analiz) to w pełni funkcjonalny system analizy strategicznej wykorzystujący najnowsze osiągnięcia w dziedzinie LLM.

Kluczowe zalety:

| # | Zaleta | Opis |
|---|--------------------------------|---|
| 1 | 100% Open Source | Brak zależności od zamkniętych rozwiązań |
| 2 | Pełna konteneryzacja | Gotowy do wdrożenia w dowolnej infrastrukturze Docker |
| 3 | Elastyczny provider LLM | Łatwa migracja między Cloudflare a self-hosted |
| 4 | Niskie koszty | ~\$0.11 za intensywną sesję testową |
| 5 | Wieloformatowy input | PDF, DOCX, PPTX, strony dynamiczne (JS) |
| 6 | Kompleksowy output | Fakty, predykcje, brakujące info, raport ze scenariuszami |

Dokumentacja wygenerowana: Grudzień 2025 **S.I.K.O.R.K.A.** – Zespół "Wifi nie działa"