

AI-lytics - Dokumentacja projektu

Pincipolo:

Mikołaj Gałkowski
Wiktor Jakubowski
Łukasz Tomaszewski

Spis treści

1 Opis projektu	1
2 Architektura rozwiązania	1
3 Aplikacja i jej właściwości	1
4 Testy bezpieczeństwa i jakości	2

1 Opis projektu

Projekt obejmuje rozwiązanie oparte o Generatywną Sztuczną Inteligencję, która ma na celu ułatwić proces eksploracji i analizy danych. Narzędzie udostępnia aplikację webową, która pozwala na interakcję z ChatBot'em, który odpowiada na zapytania użytkownika w oparciu o schemat bazy danych SQL. Przy pomocy różnych komponentów użytkownik może poprosić Sztuczną Inteligencję o opisanie bazy danych, wygenerowanie odpowiedniego zapytania SQL, a także wygenerowanie wykresu na podstawie wygenerowanych zapytań. Wszystkie te rzeczy generowane są na podstawie języka naturalnego, dzięki czemu użytkownik aplikacji nie musi posiadać technicznej wiedzy w celu analizy danych. Kod naszego rozwiązania znajduje się na [repozytorium GitHub](#).

2 Architektura rozwiązania

Na Figurze 1 przedstawiona została architektura naszego rozwiązania. wykorzystaliśmy Azure Container Instances do wdrożenia aplikacji Streamlit z możliwością konwersacji. W celu wykorzystania AI, połączylismy się za pomocą API Key do usług Azure OpenAI (GPT-3.5) oraz do PandasAI. Połączylismy naszą aplikację z bazą danych. Skonfigurowane zostały Network Security Groups i podsieci VNet do komunikacji pomiędzy aplikacją a bazą danych. Zastosowana została także filtracja ruchu (blokowanie zapytań do API modeli i bazy danych z innych adresów IP niż aplikacji).

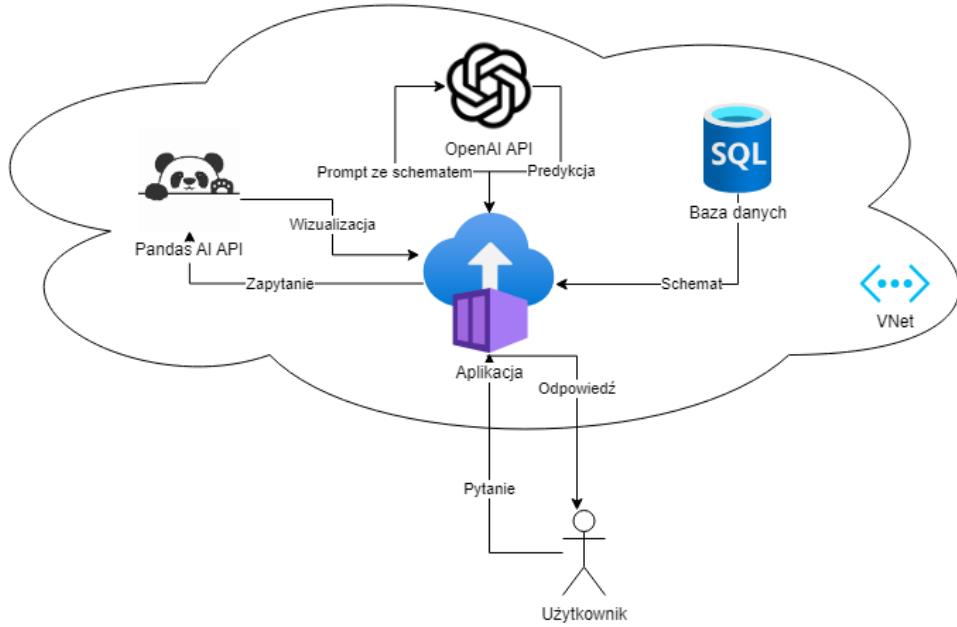
3 Aplikacja i jej właściwości

Aplikacja webowa oferuje 3 widoki: Chat z Pamięcią, Chat generujący zapytania SQL wraz z generowaniem wizualizacji oraz Chat z ReActem. Dodatkowo przy każdym zapytaniu wyświetlany jest jego koszt.

Chat z pamięcią pozwala na prowadzenie konwersacji z AI. Do zapytań przekazywany jest pobrany z bazy danych schemat, na podstawie którego AI generuje zapytania, wywołuje je oraz analizuje ich zawartość i prezentuje wyniki analizy w przystępny dla użytkownika sposób (Figura ??).

Kolejny chat generuje zapytania SQL na podstawie zapytań w języku naturalnym. Następnie wyniki tych zapytań przekazywane są do PandasAI, który generuje wykres na podstawie tych danych (Figura 3).

Ostatnim chatem jest Agent wykorzystujący framework do promptowania ReAct (Reasoning + Act). W wyniku iteracyjnych zapytań poprawia on sekwencyjnie swojego prompta, co pozwala na otrzymanie lepszych rezultatów.



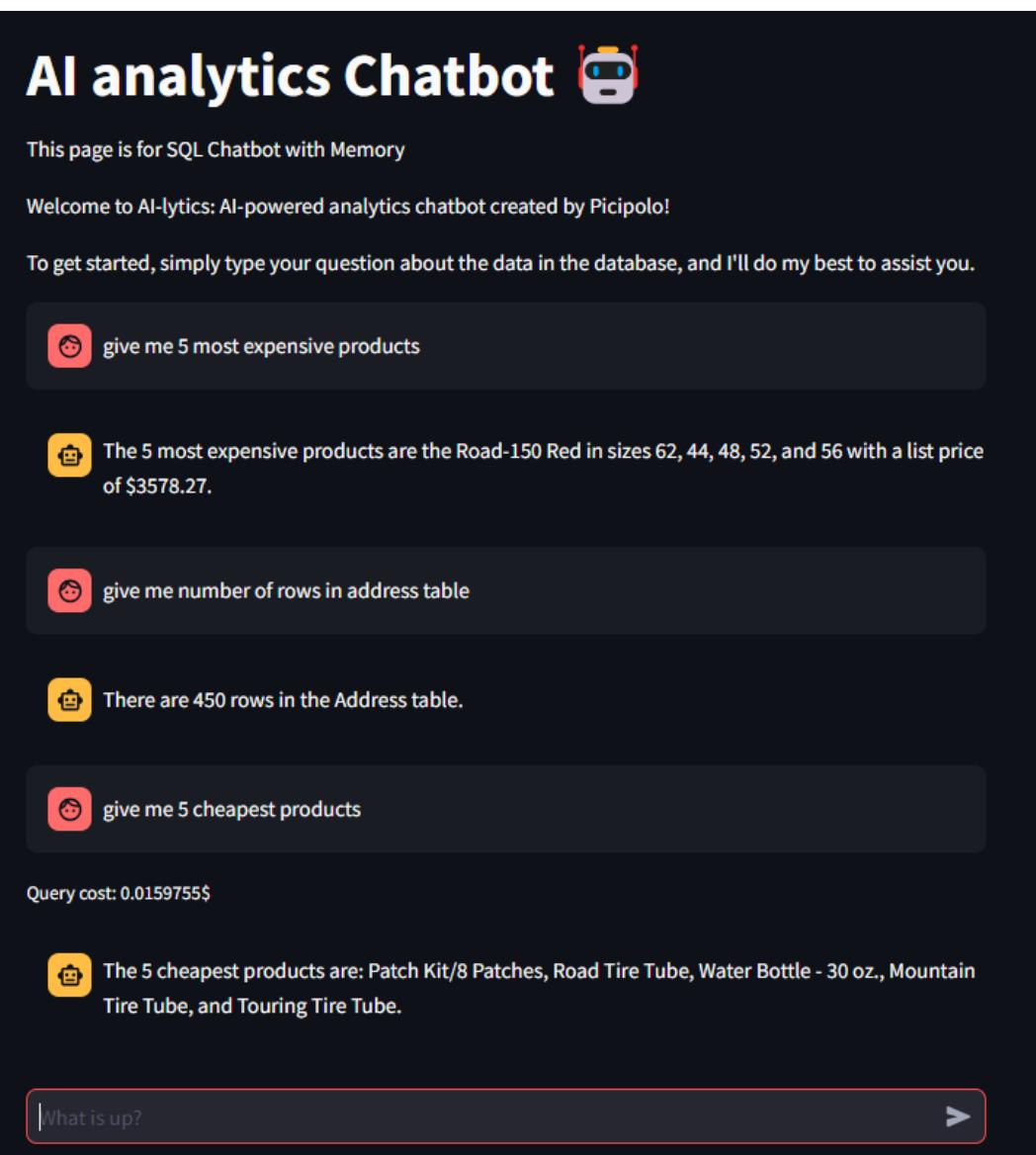
Rysunek 1: Diagram architektury rozwiązania

4 Testy bezpieczeństwa i jakości

Kod naszej aplikacji sprawdzony został przy użyciu GitGuardian. Narzędzie to sprawdza czy w kodzie nie występują wrażliwe informacje takie jak hasła czy endpointy. Przykładowe wywołanie przedstawione zostało na Figurze 4. Wykryło ono dwa zagrożenia, które zostały przez nas usunięte.

Nasz kod sprawdzany jest także na poziomie GitHuba. Każdy PR wymaga weryfikacji przez GitGuardian (Figura 5) oraz sprawdzana jest jakość kodu. Wykorzystywane do tego są narzędzia Black, Isort oraz Flake8, których działanie przedstawione zostało na Figurze 6.

Ostatnim etapem jest sprawdzenie podatności oraz nieścisłości infrastruktury chmurowej. W tym celu wykorzystywany jest Microsoft Defender for Cloud, który generuje rekomendacje dotyczące architektury oraz ich wagę. Działanie tego narzędzia przedstawione zostało na Figurze 7.



Rysunek 2: Chat z Pamięcią

AI analytics Chatbot



This page is for SQL Chatbot with Memory and Visualization

Welcome to AI-lytics: AI-powered analytics chatbot created by Picipolo!

To get started, simply type your question about the data in the database, and I'll do my best to assist you.

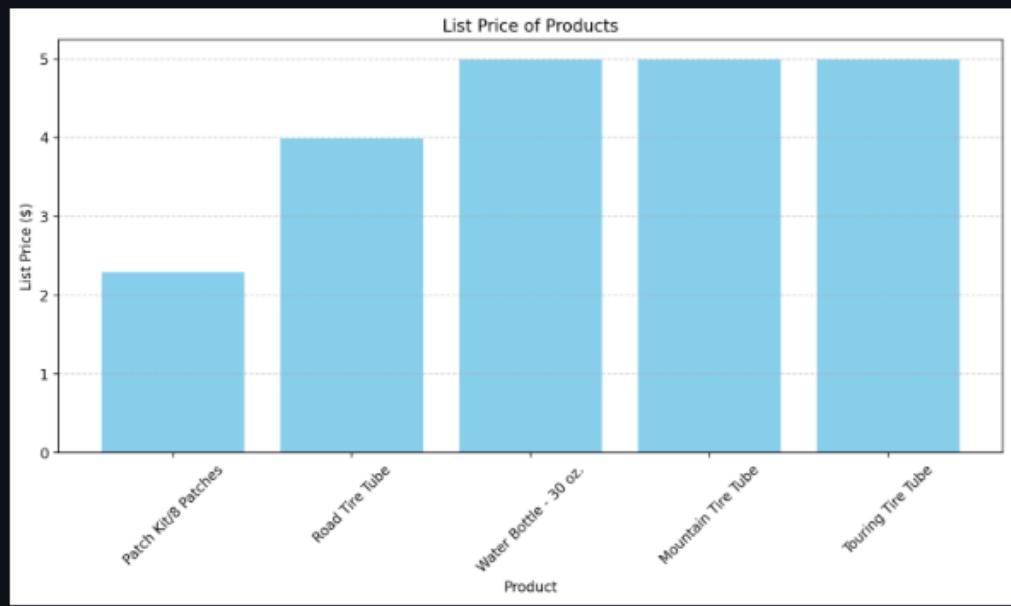


give me 5 cheapest products

Query cost: 0.0079175\$



SELECT TOP 5 [Name], [ListPrice] FROM [SalesLT].[Product] ORDER BY [ListPrice] ASC



Rysunek 3: Chat generujący zapytania SQL

Sources is Wjakubowski/gen_ai_hack...						
2 results / 2		Display 10 results ▾				
□	DATE ▾	SECRET	SEVERITY	INFO	TAGS	STATUS
□	April 13th, 2024 19:24	Generic Password ↗ ⓘ #10305327 ⌚ 2	— Unknown	⌚ Wjakubowski/gen_ai... ⓘ src/app/ssl/deployment_ssl... ⌚ From h... ⓘ Triggered ⌚ Wiktor Jakubowski +1 72559388+Wjakubowski@use...		
□	April 13th, 2024 19:24	Microsoft Azure St... ↗ ⓘ #10305328 ⌚ 2	▲ High	⌚ Wjakubowski/gen_ai... ⓘ src/app/ssl/deployment_ssl... ⌚ From h... ⓘ Triggered ⌚ Wiktor Jakubowski +1 72559388+Wjakubowski@use...		

Rysunek 4: Przykładowe wywołanie GitGuardian

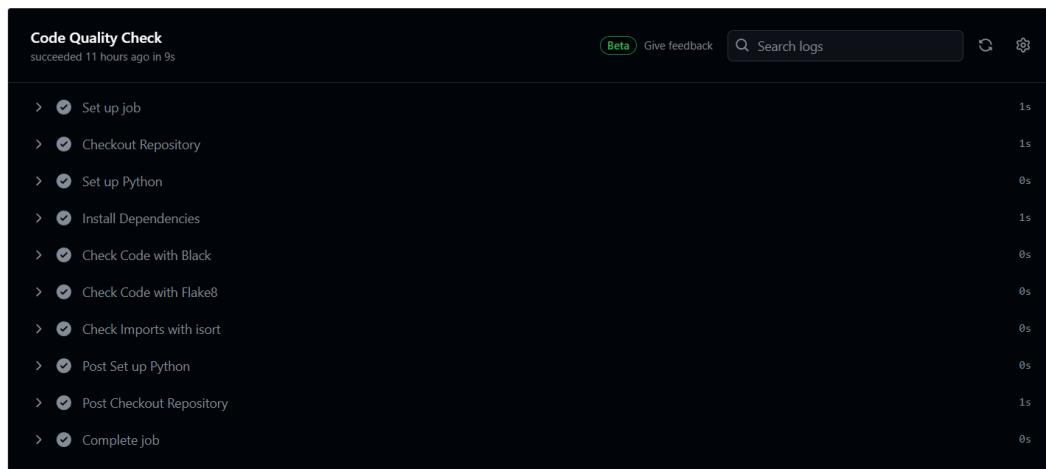
GitGuardian / GitGuardian Security Checks

succeeded 11 hours ago in 0s

No secrets detected 

41 commits were scanned without uncovering any secrets.

Rysunek 5: GitGuardian - wywołanie w GitHub



The screenshot shows the GitHub Actions interface for a 'Code Quality Check' job. The job succeeded 11 hours ago in 9s. The log output lists various steps: Set up job, Checkout Repository, Set up Python, Install Dependencies, Check Code with Black, Check Code with Flake8, Check Imports with isort, Post Set up Python, Post Checkout Repository, and Complete job. All steps are marked with a green checkmark and have a duration of 0s or 1s.

Rysunek 6: Weryfikacja jakości kodu

Severity ↑↓	Description	Status ↑↓
High	Microsoft Defender for SQL should be enabled for unprotected Azure SQL servers	● Unhealthy
High	SQL servers should have an Azure Active Directory administrator provisioned	● Unhealthy
High	SQL servers should have vulnerability assessment configured	● Unhealthy
Medium	Azure SQL Database should have Azure Active Directory Only Authentication enable	● Unhealthy
Medium	Azure SQL Database should be running TLS version 1.2 or newer	● Preview ● Healthy
Medium	Private endpoint connections on Azure SQL Database should be enabled	● Preview ● Unhealthy

Rysunek 7: Rekomendacje wygenerowane przez Microsoft Defender for Cloud