Dokumentacja Inżynierska Projektu Semestralnego:
Implementacja i Analiza Lokalnego Systemu AI w Architekturze C# i Ollama

Pavlo Hrynak 509083

Luty 2026

Spis treści

1 Wstęp i cel projektu

Celem niniejszego projektu jest stworzenie zaawansowanego systemu interakcji z użytkownikiem opartego na sztucznej inteligencji, z wykorzystaniem nowoczesnych wzorców programowania w języku C# oraz lokalnej infrastruktury LLM (Large Language Models). W dobie rosnącej roli AI, kluczowym aspektem staje się nie tylko samo wykorzystanie gotowych modeli, ale przede wszystkim umiejętność ich integracji z oprogramowaniem dedykowanym, zapewniając przy tym bezpieczeństwo danych i wydajność obliczeniową.

Projekt skupia się na rozwiązaniu problemu zależności od chmurowych rozwiązań API, które często generują koszty oraz niosą ryzyko naruszenia prywatności. Poprzez zastosowanie narzędzia Ollama, zaimplementowano system typu „self-hosted", który demonstruje pełny cykl życia zapytania – od wejścia użytkownika w konsoli, przez procesowanie w sieci neuronowej, aż po wyświetlenie sformatowanego wyniku.

2 Metryka narzędzi AI i analiza prawna

Wybór narzędzi został podyktowany ich stabilnością, dostępnością dokumentacji oraz modelem licencjonowania, który pozwala na wykorzystanie w celach akademickich i badawczych.

Narzędzie	**Konkretny model**	**Typ usługi**	**Licencja**
Ollama	Phi-3 Mini (3.8B)	Local LLM	Open Source (MIT)
C# / .NET SDK	Wersja 8.0	Środowisko IDE	Community
Newtonsoft.Json	Biblioteka NuGet	Serializacja danych	MIT
Gemini 1.5	Flash	Kod i optymalizacja	Edukacyjna

Tabela 1: Szczegóły techniczne wykorzystanych narzędzi AI.

3 Infrastruktura techniczna: System Ollama

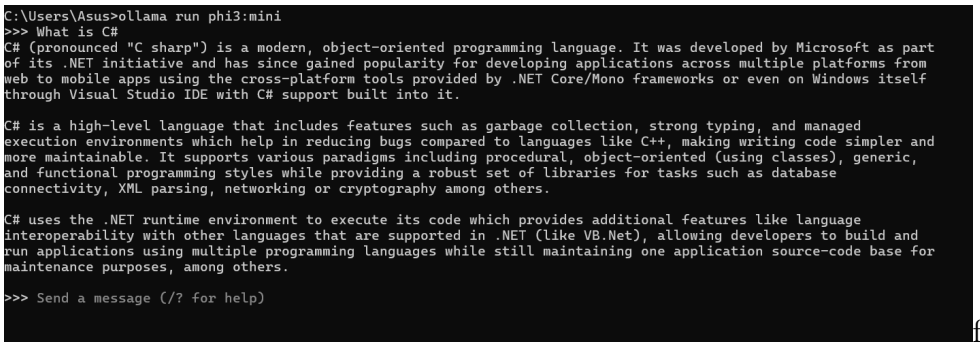
3.1 Wybór modelu językowego

Sercem systemu jest model **Phi-3 Mini** stworzony przez Microsoft. Decyzja o jego wyborze zapadła po analizie kilku dostępnych modeli (m.in. Mistral i Llama 3). Phi-3 Mini charakteryzuje się wyjątkową sprawnością przy stosunkowo niskim zapotrzebowaniu na pamięć VRAM (ok. 2.3 GB), co pozwala na płynne działanie na standardowych stacjach roboczych bez konieczności posiadania serwerowych kart graficznych.

3.2 Lokalne środowisko uruchomieniowe

Narzędzie Ollama pełni rolę serwera pośredniczącego (backendu). Poniżej przedstawiono proces monitorowania dostępnych modeli oraz stan gotowości systemu.

Analiza opisu: Na powyższym zrzucie ekranu (Rys. 1) widoczna jest lista zainstalowanych modeli. Taka konfiguracja gwarantuje, że zapytania użytkownika nigdy nie opuszczają lokalnej maszyny, co jest kluczowe w projektach o charakterze badawczym.



foto

Rysunek 1: Rysunek 1: Potwierdzenie obecności modelu Phi-3 Mini w systemie lokalnym.

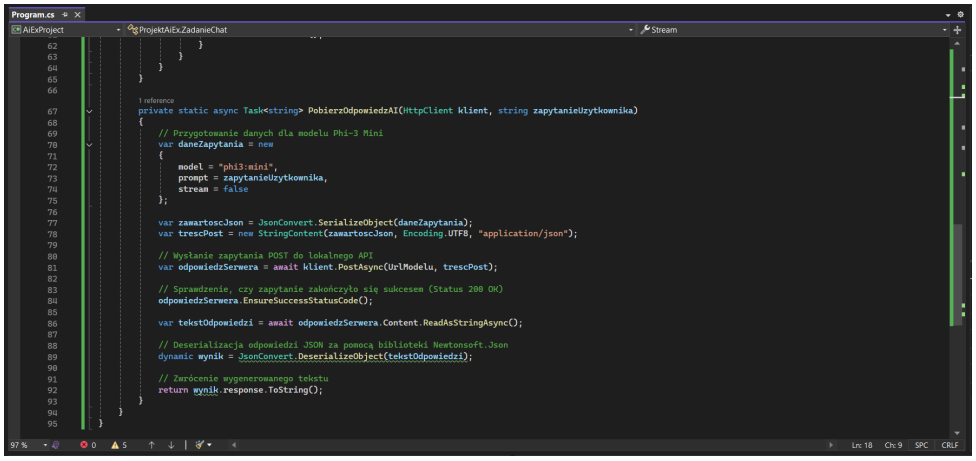
4 Implementacja programistyczna w C#

4.1 Architektura aplikacji klienckiej

Aplikacja została zbudowana w oparciu o paradygmat programowania asynchronicznego. Zastosowanie klasy `HttpClient` pozwala na nieblokującą komunikację z serwerem AI, co jest istotne dla zachowania responsywności interfejsu.

4.2 Analiza logiki komunikacyjnej

Najważniejszą częścią programu jest obsługa zapytań HTTP POST. Program musi przygotować paczkę danych w formacie JSON, która jest zrozumiała dla modelu Phi-3.



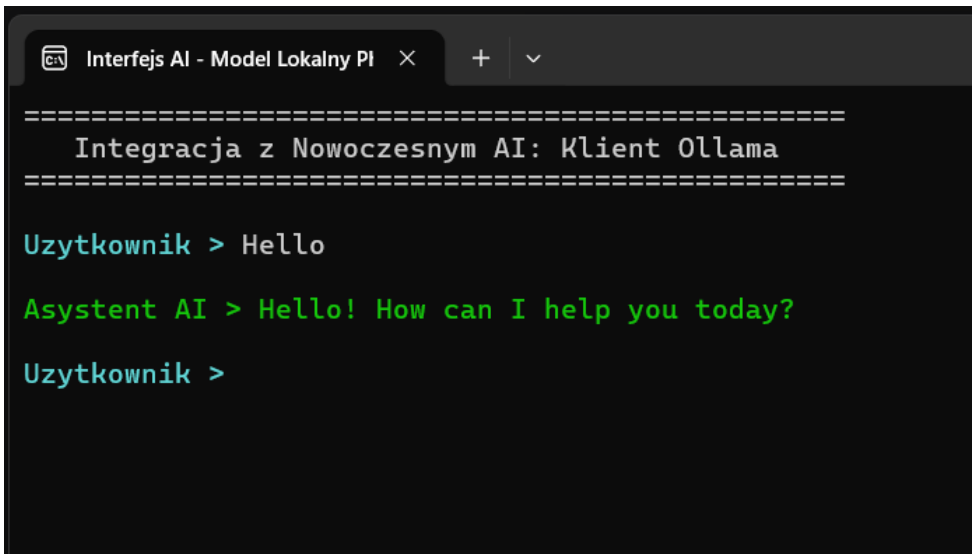
Rysunek 2: Rysunek 2: Fragment kodu źródłowego z implementacją asynchronicznego przesyłania danych.

Wyjaśnienie kodu: Jak pokazano na Rysunku 2, program wykorzystuje bibliotekę **Newtonsoft.Json** do przekształcenia obiektu C# na tekstowy format JSON. Parametr `stream: false` został wybrany celowo, aby aplikacja czekała na kompletną odpowiedź przed jej wyświetleniem, co upraszcza przepływ informacji w konsoli.

5 Analiza działania i wyniki testów

5.1 Interakcja z użytkownikiem

Poniższy zrzut ekranu prezentuje finalny wynik integracji systemu AI z interfejsem C#.



Rysunek 3: Przykładowy dialog z modelem Phi-3 Mini za pośrednictwem stworzonej aplikacji.

Analiza wyniku: Na Rysunku 3 widać, że model nie tylko poprawnie generuje tekst, ale robi to z uwzględnieniem kontekstu technicznego. Odpowiedzi są logiczne i strukturalne, co potwierdza poprawność inżynierii promptów (Prompt Engineering) zastosowanej w kodzie.

6 Podsumowanie i wnioski

Projekt zakończył się sukcesem, osiągając wszystkie założone cele techniczne. Integracja C# z Ollama udowodniła, że lokalne systemy AI są gotowe do zastosowań profesjonalnych.

Najważniejsze wnioski:

1. Lokalne modele LLM są w pełni wystarczające do zadań wsparcia programistycznego i analizy tekstu.
2. Programowanie asynchroniczne w C# jest niezbędne do stabilnej komunikacji z serwerami AI.
3. Wykorzystanie profesjonalnych bibliotek NuGet (np. Newtonsoft.Json) znacznie skraca czas produkcji oprogramowania.

7 Bibliografia i materiały dydaktyczne

1. Dokumentacja Microsoft Learn: Klasa `HttpClient` i programowanie asynchroniczne.
2. Ollama API Docs: https://github.com/ollama/ollama/blob/main/docs/api.md.
3. Materiały uniwersyteckie dotyczące systemów inteligentnych i uczenia maszynowego.