**Dokumentace AI Agenta: Více-zdrojový Asistent (ReAct)**

Tento dokument popisuje architekturu a implementaci AI agenta navrženého pro efektivní práci s nástroji a pamětí. Projekt byl vytvořen na základě vlastní logiky (**Vlastní workflow**) namísto použití komplexních frameworků, čímž byla zajištěna maximální kontrola nad tokem dat a sníženo riziko nekompatibility knihoven.

**1. Architektura a Jádro Agenta**

Agent je postaven na principu **ReAct (Reasoning and Acting)**. Každá interakce probíhá v cyklu, který má maximálně 10 kroků, což zabraňuje zacyklení a neefektivnímu plýtvání API kredity.

1. **Vstup:** Uživatel zadá dotaz.
2. **Uvažování (Thought):** Agent (LLM) analyzuje dotaz a historii konverzace.
3. **Akce (Action):** Agent se rozhodne, zda potřebuje některý z nástrojů. Pokud ano, vygeneruje akci ve striktním formátu Action: název\_nástroje("parametr").
4. **Nástroj:** Vybraný nástroj je spuštěn (např. vyhledávání nebo výpočet).
5. **Výstup:** Agent dostane výsledek nástroje a buď syntetizuje finální odpověď, nebo se vrací ke kroku 2 pro další upřesnění.

Pokud LLM vyčerpá 10 pokusů bez finální odpovědi, aktivuje se **Final Solution Tool** (poslední pokus), který přinutí LLM k přímé a okamžité odpovědi.

**2. Nástroje (Tooly) a Robustnost**

Agent je vybaven sadou specializovaných nástrojů, které umožňují získávat aktuální data a provádět složité operace:

* **search\_tool (Primární vyhledávání):** Klíčový nástroj, který zajišťuje robustnost. **Primárně** volá Google Custom Search API. V případě selhání (např. chyby 400 Bad Request) **automaticky přechází na záložní Tavily API**. Vrácený kontext z obou nástrojů obsahuje plné URL adresy.
* **wikipedia\_search:** Získává faktické informace a shrnutí z české Wikipedie.
* **calculator\_tool:** Umožňuje složité matematické výpočty, protože je postaven na bezpečné knihovně SymPy, která zvládá trigonometrické a fyzikální výpočty (např. sin(pi/2)).
* **get\_current\_datetime:** Poskytuje aktuální datum a čas.

**3. Správa Paměti a Kontextu**

Agent implementuje paměť konverzace, která zajišťuje plynulost dialogu:

* **Historie:** Konverzační paměť (conversation\_history) ukládá **pouze finální dotazy uživatele a finální odpovědi agenta** (v roli user a model).
* **Izolace:** Mezikroky, jako jsou Thought a výstupy nástrojů, nejsou ukládány do trvalé historie, ale jsou předávány LLM pouze v aktuálním kroku. Tím se **zabraňuje zacyklení a zmatení agenta** jeho vlastními předchozími myšlenkami.
* **Formátování:** Finální odpovědi jsou vždy formátovány jako **očíslovaný seznam s plnými URL adresami**, což zajišťuje čitelnost a přímou použitelnost pro uživatele.

**4. Logování a Debugování**

Pro účely ladění a auditu je implementován pokročilý logovací systém:

* **Umístění:** Při každém spuštění se vytvoří nový, časově označený soubor (chat\_log\_YYYY-MM-DD\_HH-MM-SS.log) v podadresáři **Logs**.
* **Úplná historie:** Logovací soubor obsahuje veškerou aktivitu agenta: vstupy uživatele, volání nástrojů, kompletní výstupy LLM (Thought, Action), chyby API a všechny kroky v reálném čase.
* **Čistota terminálu:** Výstup do konzole je minimalizován a slouží pouze pro konverzaci s uživatelem. Všechny interní zprávy a chyby jsou přesměrovány do logu, což zajišťuje čistý a profesionální zážitek pro uživatele.