Architektura zakázkového AI asistenta s podporou nástrojů: Hloubkový pohled na MCP, agentní frameworky a webovou integraci

Sekce 1: Úvod - Návrh akčně orientovaného AI agenta

Tato sekce stanovuje strategický kontext projektu, který přesahuje koncept jednoduchého chatbota a směřuje k plnohodnotnému, autonomnímu agentovi. Představuje čtyřvrstvý architektonický model, který bude sloužit jako vodítko pro celou zprávu.

1.1. Vize: Od konverzační AI k agentní automatizaci

Současný vývoj v oblasti umělé inteligence umožňuje přechod od tradičních chatbotů k pokročilejším systémům známým jako AI asistenti a skuteční AI agenti. Zatímco chatboty jsou typicky navrženy pro automatizaci jednoduchých, opakujících se konverzací, AI agenti představují významný posun. Jsou definováni svou schopností uvažování, plánování, paměti a autonomního používání nástrojů k dosažení cílů zadaných uživatelem.[1]

Cílem tohoto projektu je vytvořit takzvaného "cílově orientovaného agenta" (goal-based agent), který dokáže provádět vícekrokové úkoly k dosažení konkrétního výsledku.[2] Příkladem je schopnost naplánovat schůzku, což vyžaduje nejen konverzaci s uživatelem, ale i aktivní interakci s externími systémy, jako je kalendář.

1.2. Čtyřvrstvá architektura pro vlastního agenta

Pro vytvoření robustního, škálovatelného a udržovatelného systému je klíčové přijmout modulární architekturu. Požadavek na vlastní uživatelské rozhraní, které komunikuje s backendem prostřednictvím API, přirozeně vede k oddělení jednotlivých komponent. Tento princip oddělení (decoupling) není pouze konceptuálním rozdělením, ale základní architektonickou strategií, která zajišťuje, že každá část systému může být vyvíjena, nasazována a aktualizována nezávisle. To umožňuje v budoucnu například vyměnit orchestrační framework nebo změnit poskytovatele hostingu s minimálním dopadem na uživatelskou aplikaci.

Navrhovaná architektura se skládá ze čtyř hlavních vrstev:

- **Prezentační vrstva:** Jedná se o vlastní webové rozhraní ("tvář"), které bude vytvořeno. Je to vstupní bod pro uživatele, který komunikuje s backendem prostřednictvím dobře definovaného API.
- **Logická vrstva (Reasoning Layer):** "Mozek" celého systému, poháněný orchestračním frameworkem pro AI agenty (např. LangGraph, AutoGen). Tato vrstva řídí konverzaci, plánuje úkoly a rozhoduje, které nástroje použít k jejich splnění.[3]

- **Akční vrstva (Action Layer):** "Ruce" agenta. Tuto vrstvu tvoří nástroje, které může agent používat. Zpráva se zaměřuje na Model Context Protocol (MCP) jako standardizovaný způsob, jak tyto nástroje zpřístupnit.[5]
- **Deploymentová vrstva:** Infrastruktura, na které jsou hostovány logická a akční vrstva. Může se jednat o serverless funkce, kontejnery nebo řešení Platform-as-a-Service (PaaS).[7]

Sekce 2: Akční vrstva - Zvládnutí protokolu MCP (Model Context Protocol)

Tato sekce poskytuje požadovaný hloubkový pohled na MCP, vysvětluje jeho význam, dostupné nástroje a postup, jak vytvořit vlastní.

2.1. Dekonstrukce MCP: "USB-C pro AI"

Model Context Protocol (MCP) je otevřený standard vyvinutý společností Anthropic, který řeší problém vytváření zakázkových, křehkých a jednorázových integrací pro každý nástroj, který AI agent potřebuje.[6] Nahrazuje tyto ad-hoc přístupy jednotným, standardizovaným protokolem, což je často přirovnáváno k funkci USB-C portu pro AI aplikace – poskytuje standardizovaný způsob připojení AI modelů k různým zdrojům dat a nástrojům.[12]

Základní architektura
Architektura MCP se skládá ze tří klíčových komponent 6:

- **MCP Hostitel/Klient (Host/Client):** Aplikace poháněná AI (v tomto případě vytvořený agent), která využívá nástroje.
- **MCP Server:** "Chytrý adaptér", který zpřístupňuje nástroj nebo zdroj (např. souborový systém, databázi nebo API) hostiteli.
- **MCP Protokol:** Standardizovaný soubor pravidel pro komunikaci, obvykle využívající strukturované formáty jako JSON. Komunikace může probíhat lokálně nebo přes internet.

Mezi klíčové vlastnosti patří dynamické objevování (agenti mohou automaticky detekovat dostupné servery a jejich schopnosti) a podpora bohatých, obousměrných interakcí, což MCP odlišuje od jednodušších, bezstavových plugin systémů.[11]

2.2. Ekosystém MCP: Využití předpřipravených nástrojů

Vznik platforem jako Zapier MCP představuje pro vývojáře klíčový strategický bod. Volba již není jen *jak* nástroj postavit, ale *zda* ho vůbec stavět. Pro běžné úkoly, jako je plánování schůzek (primární případ užití v dotazu), je použití Zapier MCP výrazně jednodušší a rychlejší než vývoj vlastního serveru. Pro proprietární interní nástroje nebo vysoce specializované funkce je však vlastní server nezbytný. To vede k hybridní strategii: využívat PaaS řešení pro běžné úkoly a soustředit vlastní vývojové úsilí na unikátní, vysoce hodnotné nástroje. Doporučeným postupem je začít se Zapierem pro nástroj kalendáře, aby se

dosáhlo rychlého výsledku, a zároveň se učit, jak postavit vlastní MCP server pro budoucí, specializovanější potřeby.

Krajina dostupných MCP serverů je různorodá a lze ji kategorizovat podle funkce a složitosti nastavení:

- **Open-Source a komunitní servery:** Přehled dostupných serverů, například z repozitáře awesome-mcp-servers, zahrnuje několik kategorií [5]:
  - **Automatizace prohlížeče:** Nástroje využívající Playwright a Puppeteer pro web scraping, extrakci dat a vyplňování formulářů (např. microsoft/playwright-mcp, agent-infra/mcp-server-browser). **Složitost:** Střední; vyžaduje spuštění serverového procesu, potenciálně v Dockeru.
  - **API Wrappery:** Servery, které poskytují MCP rozhraní pro populární API jako YouTube, Open Library nebo Bilibili (např. kimtaeyoon83/mcp-server-youtube-transcript, 8enSmith/mcp-open-library). **Složitost:** Nízká až střední; často vyžaduje pouze poskytnutí API klíčů.
  - **Interakce s lokálním systémem:** Servery pro interakci s lokálními aplikacemi jako Apple Reminders nebo Shortcuts (např. fradser/mcp-server-apple-reminders). **Složitost:** Nízká; typicky běží jako lokální proces.
- **Platform-as-a-Service (PaaS) MCP:** Služby, které poskytují spravovaný MCP endpoint.
  - **Zapier MCP:** Toto je velmi silná možnost, která zpřístupňuje ekosystém Zapieru s více než 7 000 aplikacemi a 30 000 akcemi prostřednictvím jediného, bezpečného MCP endpointu.[14] Je to ideální kandidát pro případ použití plánování schůzek.
    Složitost: Velmi nízká; zahrnuje vygenerování URL a konfiguraci akcí ve webovém rozhraní.
- **Meta-MCP Servery:** Pokročilé koncepty jako sitbon/magg, server, který umožňuje LLM objevovat a orchestrovat *jiné* MCP servery, což agentům umožňuje rozšiřovat své vlastní schopnosti za běhu.[5]

2.3. Vytvoření vlastního MCP serveru pro kalendář

Tato podsekce poskytuje praktický, krok za krokem návod k vytvoření nástroje pro integraci s kalendářem.

- **Předpoklady:** Nastavení Python prostředí a instalace MCP SDK (např. pomocí uv add "mcp[cli]").[15]
- **Základní logika:** Následující kroky popisují potřebný Python kód:
  1. Autentizace s Google Calendar API (nebo podobnou službou).
  2. Definování funkcí pro check_availability (kontrola dostupnosti) a schedule_meeting (naplánování schůzky).
  3. Zapouzdření těchto funkcí jako MCP nástrojů v rámci instance MCP serveru pomocí Python SDK.[15]
- **Testování:** Použití nástroje MCP Inspector pro vizualizaci a testování schopností vlastního serveru před jeho integrací s agentem.[17]
- **Příklad kódu:** Bude poskytnut minimální, kompletní příklad kódu založený na vzorech z tutoriálů.[18]

Sekce 3: Logická vrstva - Výběr orchestračního frameworku pro AI agenta

Tato sekce hodnotí "mozek" agenta a porovnává hlavní frameworky, aby pomohla při výběru toho správného pro dané potřeby.

3.1. Anatomie moderního AI agenta

Frameworky pro tvorbu agentů poskytují základní komponenty pro jejich konstrukci [4]:

- **Plánování:** Schopnost rozložit složitý úkol na sekvenci menších kroků (dekompozice úkolu).[2]
- **Paměť:** Ukládání minulých interakcí pro udržení kontextu v konverzaci.[19]
- **Použití nástrojů:** Mechanismus pro výběr a spouštění nástrojů pro interakci s vnějším světem.[1]

3.2. Srovnávací analýza předních frameworků

Tato část nabízí detailní, srovnávací analýzu hlavních frameworků.

- **LangChain & LangGraph:**
  - **Filozofie:** Otevřený, kompozitní framework poskytující standardní rozhraní pro jakýkoli model, nástroj nebo databázi.[21] LangGraph je jeho rozšíření pro vytváření stavových, kontrolovatelných a cyklických agentních workflow, které modeluje jako uzly a hrany v grafu.[23]
  - **Silné stránky:** Bezkonkurenční flexibilita a obrovský ekosystém s více než 600 integracemi.[22] LangGraph nabízí jemnozrnnou kontrolu pro složité, vícekrokové uvažování a workflow s lidskou zpětnou vazbou (human-in-the-loop).[24] Je připraven pro produkční nasazení a používán velkými společnostmi.[22]
  - **Složitost:** Může mít strmou křivku učení kvůli své nízkoúrovňové, nevnucující povaze.[19] Ladění může být náročné.[27]
  - **Integrace nástrojů:** Integruje nástroje pomocí dekorátoru @tool nebo obalením jiných spustitelných komponent. Podporuje nativní API pro volání funkcí.[18]
- **LlamaIndex:**
  - **Filozofie:** Framework zaměřený na vytváření kontextově obohacených AI agentů, vynikající v propojování LLM s privátními daty.[29]
  - **Silné stránky:** Nejlepší ve své třídě pro RAG (Retrieval-Augmented Generation) pipeline. Jeho abstrakce AgentWorkflow zjednodušuje tvorbu jedno- i víceagentních systémů.[31] Silné zaměření na podnikové případy použití a parsování složitých dokumentů.[29]
  - **Složitost:** Vysokoúrovňové abstrakce jako AgentWorkflow usnadňují začátek.[31] Vytváření vlastních agentů od nuly je také možné, ale je složitější.[34]
  - **Integrace nástrojů:** Poskytuje FunctionTool pro obalení Python funkcí a QueryEngineTool pro přeměnu RAG pipelines na nástroje, které může agent použít. Podporuje také bohatou knihovnu předpřipravených nástrojů přes LlamaHub.[36]
- **Microsoft AutoGen:**

    - **Filozofie:** Framework zaměřený na vytváření víceagentních konverzačních aplikací. Vyniká při řešení složitých úkolů tím, že nechává spolupracovat více specializovaných agentů.[4]
    - **Silné stránky:** Silný pro úkoly, které lze rozdělit do odlišných rolí (např. Plánovač, Kodér, Kritik).[41] Jeho konverzační přístup je intuitivní pro modelování spolupráce.[43] Nabízí no-code UI, AutoGen Studio, pro rychlé prototypování.[4]
    - **Složitost:** Paradigma více agentů může být koncepčně složité na návrh a ladění, zejména sledování toku konverzace mezi mnoha agenty.[27]
    - **Integrace nástrojů:** Nástroje jsou registrovány jako funkce jak pro caller agenta (který navrhuje volání), tak pro executor agenta (který spouští kód).[45]
- **Microsoft Semantic Kernel:**
    - **Filozofie:** SDK navržené pro integraci LLM do stávajících podnikových aplikací, se silným zaměřením na orchestraci "pluginů" (jeho termín pro nástroje).[4]
    - **Silné stránky:** Podpora více jazyků (Python, C#, Java).[4] Jeho koncept Plugin se dobře hodí k podnikovým vývojovým vzorům a dependency injection.[47] Může importovat pluginy z OpenAPI specifikací a dokonce z MCP serverů.[47]
    - **Složitost:** Strukturovanější a názorovější než LangChain, což může být pro podnikové vývojáře jednodušší, ale méně flexibilní pro výzkumníky.[47]
    - **Integrace nástrojů:** Používá architekturu Plugin, kde jsou metody třídy dekorovány @kernel_function.[48]

3.3. Srovnávací tabulka vlastností agentních frameworků

Následující tabulka poskytuje přehledné shrnutí, které pomůže při rozhodování. Byla navržena tak, aby destilovala podrobnou analýzu do jasné, srovnávací matice, která přímo odpovídá potřebě přehledu technologií a posouzení jejich složitosti.

Vlastnost	LangChain / LangGraph	LlamaIndex	Microsoft AutoGen	Microsoft Semantic Kernel
Primární případ použití	Flexibilní, vlastní agentní workflow s jemnozrnnou kontrolou.	Kontextově citliví agenti, zejména s RAG nad privátními daty.	Kolaborativní víceagentní systémy pro komplexní dekompozici úkolů.	Integrace LLM do podnikových aplikací (vícejazyčné).
Integrace nástrojů	Dekorátor @tool; obrovská knihovna integrací.[18]	FunctionTool, QueryEngineTool; nástroje z LlamaHub.[36]	register_function pro caller a executor agenty.[45]	Architektura Plugin s dekorátorem @kernel_function.[47]

Podpora více agentů	Klíčová vlastnost LangGraph (orchestrace založená na grafu).[25]	Podporováno přes AgentWorkflow.[31]	Základní designový princip (konverzační orchestrace).[39]	Podporováno přes Agent Framework; agenti mohou spolupracovat.[47]
Křivka učení	Vysoká (zejména LangGraph), nabízí největší kontrolu.[27]	Střední; vysokoúrovňové abstrakce zjednodušují běžné případy použití.[31]	Střední-Vysoká; paradigma více agentů může být složité na ladění.[27]	Střední; strukturovanější a názorovější, známé podnikovým vývojářům.[47]
Zralost ekosystému	Velmi vysoká; největší komunita a nejvíce integrací.[22]	Vysoká; silné zaměření na datové konektory a RAG komponenty.[29]	Střední; rychle roste, podporováno Microsoft Research.[4]	Střední; silné podnikové zaměření, podporováno Microsoftem.[47]

Sekce 4: Integrační vrstva - Propojení logiky agenta s reálnými nástroji

Tato sekce překlenuje mezeru mezi "mozkem" a "rukama" agenta, vysvětluje základní technologii a strategickou volbu mezi MCP a přímou integrací nástrojů.

4.1. Nativní volání funkcí: Motor pod kapotou

"Volání funkcí" (function calling) nebo "použití nástrojů" (tool use) je nativní schopnost moderních LLM od poskytovatelů jako OpenAI, Google (Gemini) a Anthropic (Claude).[50] Tento mechanismus je základním stavebním kamenem, který všechny agentní frameworky a MCP využívají k umožnění použití nástrojů.[56]

Proces probíhá následovně:

1. Vývojář poskytne LLM schéma (strukturovaný popis) dostupných funkcí.
2. Na základě uživatelského promptu LLM určí, zda by měla být nějaká funkce volána.
3. Pokud ano, LLM vygeneruje strukturovaný JSON objekt s názvem funkce a argumenty, které se mají použít.[50]
4. Kód aplikace obdrží tento JSON, spustí skutečnou funkci a výsledek pošle zpět LLM, aby zformuloval konečnou odpověď.[57]

4.2. Strategická volba: MCP vs. přímá integrace nástrojů ve frameworku

Existují dva hlavní přístupy, jak agentovi poskytnout nástroje, a volba mezi nimi má strategické důsledky.

- **Přímá integrace:** Použití vestavěných funkcí pro nástroje/pluginy ve zvoleném frameworku (např. @tool v LangChain, FunctionTool v LlamaIndex).
  - **Výhody:** Jednodušší pro jednoho agenta s několika málo nástroji. Těsně integrované s logikou frameworku.
  - **Nevýhody:** Nástroje jsou svázány s konkrétním frameworkem. Opětovné použití nástrojů v různých agentech nebo aplikacích vyžaduje duplikaci kódu nebo vlastní abstrakce. S rostoucím počtem nástrojů se stává systém křehkým a těžko spravovatelným – což je přesně problém, který MCP řeší.[6]
- **Integrace přes MCP:** Použití agentního frameworku jako MCP hostitele pro připojení k externím MCP serverům.
  - **Výhody:** Odděluje nástroje od logiky agenta. Nástroje se stávají znovupoužitelnými, jazykově agnostickými "agentními mikroslužbami". MCP server napsaný v Pythonu může být použit agentem vytvořeným v Semantic Kernel v C#. Podporuje škálovatelnou, standardizovanou a udržitelnou architekturu.[6]
  - **Nevýhody:** Přidává další vrstvu infrastruktury (samotný MCP server). Může být zbytečně složité pro velmi jednoduchý prototyp s jedním nástrojem.

Pro cíl vytvořit sofistikovaného, škálovatelného asistenta je přijetí MCP od samého začátku doporučeným strategickým přístupem. MCP je více než jen protokol pro nástroje; představuje vznik standardizované komunikační vrstvy pro budoucí "agentní web" nebo "společnost agentů".[11] Stejně jako API Gateway spravují a zabezpečují přístup k mikroslužbám pro tradiční aplikace, MCP servery budou spravovat a zabezpečovat přístup k nástrojům pro AI agenty. Tento pohled přerámovává MCP z pouhé technické výhody na základní prvek infrastruktury pro budování komplexních, víceagentních systémů, které mohou spolupracovat nejen v rámci jedné aplikace, ale i napříč organizačními hranicemi.

Sekce 5: Prezentační a deploymentová vrstva - Oživení agenta

Tato sekce se zaměřuje na praktické aspekty hostování agenta a jeho propojení s webovou stránkou.

5.1. Architektonický návrh pro webově integrovaného asistenta

Následující diagram a popis ilustrují kompletní tok dat:

1. Uživatel interaguje s vlastním rozhraním ("tváří") na webové stránce (Prezentační vrstva).
2. JavaScript na webu provede API volání (např. na /api/agent/chat) na backend.
3. API endpoint, hostovaný na zvolené infrastruktuře (Deploymentová vrstva), přijme požadavek.
4. Požadavek je předán agentovi (Logická vrstva).
5. Agent zpracuje požadavek, rozhodne se použít nástroj a komunikuje s příslušným MCP serverem (Akční vrstva).

6. MCP server provede akci (např. zavolá Google Calendar API).
7. Výsledek je vrácen agentovi, který zformuluje odpověď.
8. Konečná odpověď je odeslána zpět přes API na webovou stránku, která aktualizuje uživatelské rozhraní.

5.2. Hodnocení možností nasazení

Pro backend agenta (logickou a akční vrstvu) existuje několik strategií hostování:

- **Kontejnerizované API (např. FastAPI s Dockerem):**
  - **Jak to funguje:** Agent (vytvořený v LangChain/LlamaIndex/AutoGen) je obalen ve webovém serveru FastAPI. Celá aplikace s jejími závislostmi je zabalena do Docker kontejneru.
  - **Výhody:** Vysoká přenositelnost, konzistentní prostředí, snadné nasazení na jakéhokoli cloudového poskytovatele nebo on-premises. LangServe je specializovaná knihovna pro snadné nasazení LangChain chains jako REST API.[59]
  - **Nevýhody:** Vyžaduje správu kontejneru a serverového procesu.
  - **Složitost:** Střední. Vyžaduje znalost FastAPI a Dockeru.[61]
- **Serverless funkce (např. AWS Lambda, Google Cloud Run):**
  - **Jak to funguje:** Logika agenta je umístěna do funkce, která je spouštěna HTTP požadavkem přes API Gateway.
  - **Výhody:** Automatické škálování, model plateb za skutečné využití, žádná správa serverů.[9]
  - **Nevýhody:** "Studené starty" (cold starts) mohou způsobovat latenci, což může být problém u chatových aplikací v reálném čase. Existují časové limity pro běh (např. 15 minut pro Lambda).[63] Správa závislostí může být složitější než u kontejnerů.[65]
  - **Složitost:** Střední. Vyžaduje porozumění serverless konceptům a IAM oprávněním.
- **Platform-as-a-Service (PaaS) pro AI (např. Vercel):**
  - **Jak to funguje:** Vercel poskytuje optimalizované prostředí pro nasazování AI aplikací s funkcemi jako AI SDK, serverless funkce a dokonce nativní podporu pro nasazení MCP serverů.[7]
  - **Výhody:** Vynikající vývojářská zkušenost, optimalizováno pro integraci s frontendem, spravovaná infrastruktura pro specifické AI zátěže.[68]
  - **Nevýhody:** Může být dražší a může vést k závislosti na ekosystému konkrétní platformy.
  - **Složitost:** Nízká až střední. Navrženo pro snadné použití.

Sekce 6: Zabezpečení agenta - Bezpečnostní a provozní osvědčené postupy

Tato sekce se zabývá kritickým nefunkčním požadavkem – bezpečností agenta, který může provádět akce v reálném světě.

6.1. Porozumění hrozbám: Prompt Injection

- **Definice hrozby:** Útok typu "prompt injection" nastává, když škodlivý uživatel poskytne vstup, který přiměje LLM ignorovat své původní instrukce a provést nezamýšlené akce.[69] Jedná se o nejvýznamnější zranitelnost LLM aplikací.[72]
- **Proč je to nebezpečné pro agenty s nástroji:** Na rozdíl od jednoduchého chatbota, kde je rizikem generování nevhodného textu, může být agent s nástroji zmanipulován k provádění škodlivých akcí, jako je mazání událostí v kalendáři, odesílání spamových e-mailů nebo únik citlivých dat získaných nástrojem.[73]
- **Typy injektáže:**
  - **Přímá injektáž:** Uživatel přímo přikáže agentovi "Ignoruj předchozí instrukce...".[69]
  - **Nepřímá injektáž:** Škodlivý prompt je skrytý ve zdroji dat, který agent zpracovává, například na webové stránce, kterou má shrnout, nebo v e-mailu, který čte.[70]

6.2. Rámec pro bezpečnost agenta

Následující kontrolní seznam obsahuje akční bezpečnostní postupy:

- **Princip nejmenších privilegií (Principle of Least Privilege):** Udělte nástrojům absolutní minimum oprávnění potřebných k jejich funkci. Nástroj pro kalendář by měl mít pouze oprávnění číst volné/obsazené časy a vytvářet nové události, nikoli mazat stávající nebo číst detaily událostí.[74]
- **Validace vstupů a sanitizace výstupů:** Přísně kontrolujte a validujte veškerá data předávaná nástrojům. Nikdy nedůvěřujte uživatelskému vstupu. Sanitizujte výstupy z nástrojů předtím, než jsou předány zpět LLM nebo zobrazeny uživateli.[74]
- **Potvrzení člověkem (Human-in-the-Loop):** Pro jakoukoli kritickou nebo nevratnou akci (jako je odeslání pozvánky do kalendáře nebo smazání souboru) musí být plán agenta předložen uživateli k explicitnímu potvrzení před provedením. Toto je klíčová funkce v frameworcích jako LangGraph.[24]
- **Sandboxing a izolace:** Spouštějte agenty, zejména ty, které provádějí kód nebo interagují se souborovým systémem, v izolovaných prostředích jako jsou Docker kontejnery nebo microVM, aby se omezily potenciální škody.[74]
- **Monitorování a omezování rychlosti (Rate Limiting):** Přistupujte k agentovi jako k jakékoli produkční službě. Monitorujte jeho chování, logujte všechna volání nástrojů a implementujte omezování rychlosti, abyste předešli zneužití.[74]

Sekce 7: Závěr a doporučený další postup

Tato závěrečná sekce shrnuje zjištění zprávy do konkrétního doporučení a fázového implementačního plánu.

7.1. Doporučený technologický stack

- **Prezentační vrstva:** Vlastní webový frontend (React, Vue atd.).
- **Logická vrstva: LangGraph**. Jeho explicitní, stavový a grafový přístup poskytuje kontrolu a spolehlivost potřebnou pro vícekrokové úkoly, jako je plánování. Jeho podpora pro human-in-the-loop je klíčová pro bezpečnost.
- **Akční vrstva:** Hybridní přístup k MCP. Začněte se **Zapier MCP** pro nástroj kalendáře, abyste dosáhli rychlého prototypu. Současně vytvořte vlastní **MCP server pomocí Python SDK** pro druhý, jednodušší nástroj, abyste si vybudovali zkušenosti pro budoucí proprietární integrace.
- **Deploymentová vrstva: FastAPI a Docker**. Toto poskytuje nejlepší rovnováhu mezi přenositelností, škálovatelností a kontrolou, což umožňuje nasadit backend agenta u jakéhokoli významného cloudového poskytovatele.

7.2. Fázový implementační plán

- **Fáze 1: Minimální životaschopný agent (MVA):**
  1. Vytvořte FastAPI backend s agentem v LangGraph.
  2. Nakonfigurujte Zapier MCP endpoint pro přístup ke Google Calendar.
  3. Integrujte agenta s nástrojem Zapier MCP.
  4. Vytvořte jednoduchý API endpoint na FastAPI serveru pro příjem promptu a vracení konečné textové odpovědi agenta.
  5. Otestujte celý tok bez UI (pomocí nástrojů jako curl nebo Postman).
- **Fáze 2: Webová integrace a vlastní UI:**
  1. Vyviňte vlastní "tvář" na webové stránce.
  2. Implementujte API volání z webu na FastAPI backend.
  3. Implementujte streamování odpovědí, aby se v reálném čase zobrazoval "myšlenkový proces" agenta pro lepší uživatelský zážitek.
- **Fáze 3: Zvýšení bezpečnosti a přidání vlastních nástrojů:**
  1. Implementujte krok potvrzení člověkem (human-in-the-loop) ve vašem LangGraph workflow předtím, než agent odešle finální pozvánku do kalendáře.
  2. Vytvořte a nasaďte svůj první vlastní MCP server pro nový nástroj.
  3. Integrujte tento nový vlastní nástroj do svého agenta.
- **Fáze 4: Škálování a monitorování:**
  1. Nasaďte svou Dockerizovanou aplikaci u cloudového poskytovatele (např. Google Cloud Run, AWS Fargate).
  2. Integrujte monitorování a logování (např. pomocí LangSmith) pro sledování výkonu agenta a řešení problémů v produkčním prostředí.

Citovaná díla

1. What are AI agents? Definition, examples, and types | Google Cloud, použito září 12, 2025, https://cloud.google.com/discover/what-are-ai-agents
2. What Are AI Agents? | IBM, použito září 12, 2025, https://www.ibm.com/think/topics/ai-agents
3. What is AI Agent Orchestration? - IBM, použito září 12, 2025, https://www.ibm.com/think/topics/ai-agent-orchestration
4. Best AI Agent Frameworks by Category in 2025 (Open-Source & Proprietary) - Bitcot, použito září 12, 2025, https://www.bitcot.com/best-ai-agent-frameworks-by-category/
5. punkpeye/awesome-mcp-servers: A collection of MCP servers. - GitHub, použito září 12, 2025, https://github.com/punkpeye/awesome-mcp-servers
6. MCP Explained: The New Standard Connecting AI to Everything | by Edwin Lisowski, použito září 12, 2025, https://medium.com/@elisowski/mcp-explained-the-new-standard-connecting-ai-to-everything-79c5a1c98

288
7. How to build AI apps: an overview - Vercel, použito září 12, 2025, https://vercel.com/guides/how-to-build-ai-app
8. Deploying Multi-Agent Systems on AWS | by Hasnain Hakim | Aug, 2025 | Medium, použito září 12, 2025, https://medium.com/@hasnain.hakim25/deploying-multi-agent-systems-on-aws-05fd741b50a7
9. How to Build Serverless AI Agents Using Lambda and Bedrock - AWS in Plain English, použito září 12, 2025, https://aws.plainenglish.io/how-to-build-serverless-ai-agents-using-lambda-and-bedrock-19409e5f7621
10. How Do You Actually Deploy These Things??? A step by step friendly guide for newbs : r/AI_Agents - Reddit, použito září 12, 2025, https://www.reddit.com/r/AI_Agents/comments/1jmfw32/how_do_you_actually_deploy_these_things_a_step_by/
11. #14: What Is MCP, and Why Is Everyone – Suddenly!– Talking About ..., použito září 12, 2025, https://huggingface.co/blog/Kseniase/mcp
12. Model context protocol (MCP) - OpenAI Agents SDK, použito září 12, 2025, https://openai.github.io/openai-agents-python/mcp/
13. Model Context Protocol (MCP) - Anthropic API, použito září 12, 2025, https://docs.anthropic.com/en/docs/mcp
14. Zapier MCP—Connect your AI to any app instantly, použito září 12, 2025, https://zapier.com/mcp
15. modelcontextprotocol/python-sdk: The official Python SDK ... - GitHub, použito září 12, 2025, https://github.com/modelcontextprotocol/python-sdk
16. Model Context Protocol (MCP): A Guide With Demo Project - DataCamp, použito září 12, 2025, https://www.datacamp.com/tutorial/mcp-model-context-protocol
17. MCP - Model Context Protocol - SDK - Python - YouTube, použito září 12, 2025, https://www.youtube.com/watch?v=oq3dkNm51qc
18. How to use tools in a chain | 🦜 LangChain, použito září 12, 2025, https://python.langchain.com/docs/how_to/tools_chain/
19. Which AI Agent framework should i use? (CrewAI, Langgraph, Majestic-one and pure code), použito září 12, 2025, https://medium.com/@aydinKerem/which-ai-agent-framework-i-should-use-crewai-langgraph-majestic-one-and-pure-code-e16a6e4d9252
20. Writing effective tools for agents — with agents - Anthropic, použito září 12, 2025, https://www.anthropic.com/engineering/writing-tools-for-agents
21. What Is LangChain? | IBM, použito září 12, 2025, https://www.ibm.com/think/topics/langchain
22. One interface, integrate any LLM. - LangChain, použito září 12, 2025, https://www.langchain.com/langchain
23. Architecture | 🦜 LangChain, použito září 12, 2025, https://python.langchain.com/docs/concepts/architecture/
24. LangGraph - LangChain, použito září 12, 2025, https://www.langchain.com/langgraph
25. How to Build the Ultimate AI Automation with Multi-Agent Collaboration - LangChain Blog, použito září 12, 2025, https://blog.langchain.com/how-to-build-the-ultimate-ai-automation-with-multi-agent-collaboration/
26. LangChain, použito září 12, 2025, https://www.langchain.com/
27. Top 5 Open-Source Agentic Frameworks - Research AIMultiple, použito září 12, 2025, https://research.aimultiple.com/agentic-frameworks/
28. How to create tools | 🦜 LangChain, použito září 12, 2025, https://python.langchain.com/docs/how_to/custom_tools/
29. LlamaIndex - Build Knowledge Assistants over your Enterprise Data, použito září 12, 2025, https://www.llamaindex.ai/
30. LlamaIndex Integration - AgentQL Documentation, použito září 12, 2025, https://docs.agentql.com/integrations/llamaindex
31. Using Agents in LlamaIndex - Hugging Face Agents Course, použito září 12, 2025, https://huggingface.co/learn/agents-course/unit2/llama-index/agents
32. Diving into LlamaIndex AgentWorkflow: A Nearly Perfect Multi-Agent Orchestration Solution, použito září 12, 2025, https://www.dataleadsfuture.com/diving-into-llamaindex-agentworkflow-a-nearly-perfect-multi-agent-orchestration-solution/
33. Introducing AgentWorkflow: A Powerful System for Building AI Agent Systems - LlamaIndex, použito září 12, 2025, https://www.llamaindex.ai/blog/introducing-agentworkflow-a-powerful-system-for-building-ai-agent-systems
34. Agents - LlamaIndex, použito září 12, 2025, https://docs.llamaindex.ai/en/stable/module_guides/deploying/agents/
35. Building a Custom Agent - LlamaIndex 0.9.48, použito září 12, 2025, https://docs.llamaindex.ai/en/v0.9.48/examples/agent/custom_agent.html

36. Tools - LlamaIndex, použito září 12, 2025, https://docs.llamaindex.ai/en/stable/module_guides/deploying/agents/tools/
37. Using existing tools - LlamaIndex, použito září 12, 2025, https://docs.llamaindex.ai/en/stable/understanding/agent/tools/
38. AI Agent Frameworks: Choosing the Right Foundation for Your Business | IBM, použito září 12, 2025, https://www.ibm.com/think/insights/top-ai-agent-frameworks
39. Introduction to AutoGen - Microsoft Open Source, použito září 12, 2025, https://microsoft.github.io/autogen/0.2/docs/tutorial/introduction/
40. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations, použito září 12, 2025, https://openreview.net/forum?id=BAakY1hNKS
41. How to Use Microsoft AutoGen Framework to Build AI Agents | Charter Global, použito září 12, 2025, https://www.charterglobal.com/how-to-use-the-microsoft-autogen-framework-to-build-ai-agents/
42. Building Multi Agent Framework with AutoGen - Analytics Vidhya, použito září 12, 2025, https://www.analyticsvidhya.com/blog/2023/11/launching-into-autogen-exploring-the-basics-of-a-multi-agent-framework/
43. LangGraph: Multi-Agent Workflows - LangChain Blog, použito září 12, 2025, https://blog.langchain.com/langgraph-multi-agent-workflows/
44. AutoGen Tutorial: Build Multi-Agent AI Applications - DataCamp, použito září 12, 2025, https://www.datacamp.com/tutorial/autogen-tutorial
45. Tool Use | AutoGen 0.2 - Microsoft Open Source, použito září 12, 2025, https://microsoft.github.io/autogen/0.2/docs/tutorial/tool-use/
46. How to create a GenAI agent using Semantic Kernel | Nearform, použito září 12, 2025, https://nearform.com/digital-community/how-to-create-a-genai-agent-using-semantic-kernel/
47. Semantic Kernel documentation | Microsoft Learn, použito září 12, 2025, https://learn.microsoft.com/en-us/semantic-kernel/
48. Provide native code to your agents | Microsoft Learn, použito září 12, 2025, https://learn.microsoft.com/en-us/semantic-kernel/concepts/plugins/adding-native-plugins
49. Multi-agent Conversation Framework | AutoGen 0.2, použito září 12, 2025, https://microsoft.github.io/autogen/0.2/docs/Use-Cases/agent_chat/
50. What is Function Calling with LLMs? - Hopsworks, použito září 12, 2025, https://www.hopsworks.ai/dictionary/function-calling-with-llms
51. Function Calling with LLMs - Prompt Engineering Guide, použito září 12, 2025, https://www.promptingguide.ai/applications/function_calling
52. An introduction to function calling and tool use - Apideck, použito září 12, 2025, https://www.apideck.com/blog/llm-tool-use-and-function-calling
53. Function Calling in the OpenAI API, použito září 12, 2025, https://help.openai.com/en/articles/8555517-function-calling-in-the-openai-api
54. Function calling using the Gemini API | Firebase AI Logic - Google, použito září 12, 2025, https://firebase.google.com/docs/ai-logic/function-calling
55. Tool use with Claude - Anthropic, použito září 12, 2025, https://docs.anthropic.com/en/docs/agents-and-tools/tool-use/overview
56. Introducing Function Calling Abilities via Multi-task Learning of Granular Tasks - arXiv, použito září 12, 2025, https://arxiv.org/html/2407.00121v1
57. How to use function calling with Azure OpenAI in Azure AI Foundry Models - Microsoft Learn, použito září 12, 2025, https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/function-calling
58. Function calling with the Gemini API | Google AI for Developers, použito září 12, 2025, https://ai.google.dev/gemini-api/docs/function-calling
59. LangServe | 🦜 LangChain, použito září 12, 2025, https://python.langchain.com/docs/langserve/
60. Building and Deploying a MinIO-Powered LangChain Agent API with LangServe, použito září 12, 2025, https://blog.min.io/minio-powered-langchain-agent-with-langserve/
61. From Prompt to Production: Dockerizing a LangChain Agent with FastAPI - DEV Community, použito září 12, 2025, https://dev.to/moni121189/from-prompt-to-production-dockerizing-a-langchain-agent-with-fastapi-1pe3
62. Hosting Langchain With FastAPI — Vacation Planner Project | by Alex - Level Up Coding, použito září 12, 2025, https://levelup.gitconnected.com/hosting-langchain-with-fastapi-vacation-planner-project-ba706ff37c3e
63. AWS Lambda vs. Google Cloud Functions: Top 10 Differences - Lumigo, použito září 12, 2025, https://lumigo.io/learn/aws-lambda-vs-google-cloud-functions-top-10-differences/
64. Comparison of AWS Lambda and Google Cloud Functions - Saigon Technology, použito září 12, 2025, https://saigontechnology.com/blog/aws-lambda-vs-google-cloud-functions/
65. Is it just me, or is Google Cloud (Run) Function way easier to handle than AWS Lambda? | by Sacha Storz | Medium, použito září 12, 2025,

https://medium.com/@scmstorz/is-it-just-me-or-is-google-cloud-run-function-way-easier-to-handle-than-aws-lambda-16df97b84a03
66. Deploying to Vercel, použito září 12, 2025, https://vercel.com/docs/deployments
67. AI Agents on Vercel, použito září 12, 2025, https://vercel.com/docs/agents
68. Deploy AI at the speed of frontend - Vercel, použito září 12, 2025, https://vercel.com/ai
69. What Is a Prompt Injection Attack? - IBM, použito září 12, 2025, https://www.ibm.com/think/topics/prompt-injection
70. What Is a Prompt Injection Attack? [Examples & Prevention] - Palo Alto Networks, použito září 12, 2025, https://www.paloaltonetworks.com/cyberpedia/what-is-a-prompt-injection-attack
71. Prompt Injection Attacks on Applications That Use LLMs: eBook - Invicti, použito září 12, 2025, https://www.invicti.com/white-papers/prompt-injection-attacks-on-llm-applications-ebook/
72. Prompt Injection 101 for Large Language Models | Keysight Blogs, použito září 12, 2025, https://www.keysight.com/blogs/en/inds/ai/prompt-injection-101-for-llm
73. Guide to Ethical Red Teaming: Prompt Injection Attacks on Multi-Modal LLM Agents, použito září 12, 2025, https://testsavant.ai/guide-to-ethical-red-teaming-prompt-injection-attacks-on-multi-modal-llm-agents/
74. 7 Proven Tips to Secure AI Agents from Cyber Attacks - Jit.io, použito září 12, 2025, https://www.jit.io/resources/devsecops/7-proven-tips-to-secure-ai-agents-from-cyber-attacks
75. Best Practices for Mitigating the Security Risks of Agentic AI - Prey Project, použito září 12, 2025, https://preyproject.com/blog/mitigating-agentic-ai-security-risks
76. AI Agent Best Practices and Ethical Considerations | Writesonic, použito září 12, 2025, https://writesonic.com/blog/ai-agents-best-practices
77. What are your go-to strategies for securing autonomous AI agents? : r/cybersecurity - Reddit, použito září 12, 2025, https://www.reddit.com/r/cybersecurity/comments/1n3l3dr/what_are_your_goto_strategies_for_securing/