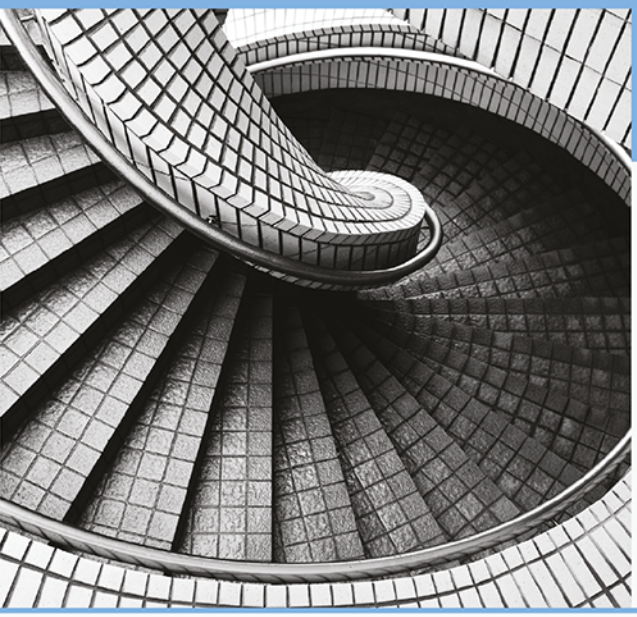
OKIEM EKSPERTA

Helion

Agenty AI w praktyce

Projektowanie, wdrażanie
i skalowanie autonomicznych systemów



Valentina Alto

‹packt›

Tytuł oryginału: AI Agents in Practice: Design, implement, and scale autonomous AI systems for production

Tłumaczenie: Piotr Rajca

ISBN: 978-83-289-3663-8

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/agaipr
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

CZĘŚĆ 2. Projektowanie, budowanie i skalowanie agentów AI

Potrzeba orkiestratora AI

Rozdział 3

Od czasu pojawienia się dużych modeli językowych i gwałtownego rozwoju aplikacji opartych na sztucznej inteligencji programiści stanęli przed rosnącym wyzwaniem: jak efektywnie zarządzać coraz bardziej złożonymi systemami AI. Wraz ze wzrostem poziomu zaawansowania oraz autonomii działania agentów AI ich zachowania muszą być ustrukturyzowane, monitorowane i optymalizowane, często w ramach wielu narzędzi, usług i źródeł danych. Ta rosnąca złożoność wytworzyła pilną potrzebę orkiestracji: sposobu na zapewnienie, że te inteligentne komponenty, dążąc do wspólnego celu, będą ze sobą współpracować bez zakłóceń.

W odpowiedzi na tę potrzebę powstały orkiestratory AI. Zamiast oferować jedynie gotowe komponenty, dostarczają one framework do strukturyzowania interakcji, zarządzania zależnościami i utrzymywania kontroli nad przepływami pracy obejmującymi wiele agentów lub modułów, jednocześnie przyspieszając rozwój i zmniejszając ryzyko operacyjne.

W tym rozdziale zajmiemy się następującymi zagadnieniami:

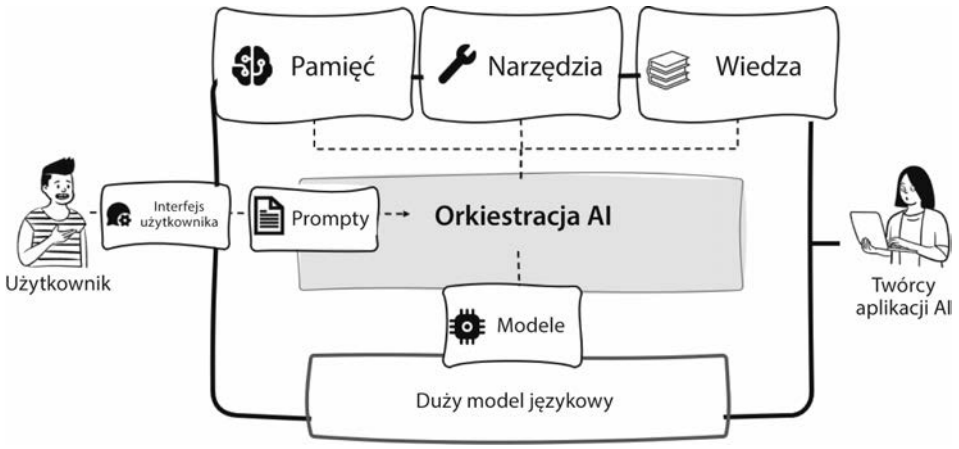
- wprowadzeniem do orkiestratorów AI;
- kluczowymi komponentami orkiestratorów AI;
- przeglądem najpopularniejszych orkiestratorów AI dostępnych na rynku;
- sposobem dobierania odpowiedniego orkiestratora dla Twojego agenta AI.

Po przeczytaniu tego rozdziału będziesz znał najpopularniejsze orkiestratory AI oraz sposoby ich wykorzystania w Twoim unikalnym przypadku użycia agentów.

Wprowadzenie do orkiestratorów AI

Oczywiste już jest, że wykorzystanie dużych modeli językowych (LLM) wykracza daleko poza proste wywołania API — wymaga ono zastosowania orkiestracji narzędzi, zarządzania pamięcią i koordynowania złożonych interakcji, niezbędnych do tworzenia prawdziwie inteligentnych systemów. Podczas gdy wczesne integracje AI opierały się na bezpośrednich interakcjach z modelami, nowoczesne agenty AI wymagają bardziej ustrukturyzowanego podejścia do zarządzania przepływami pracy, integracji zewnętrznych narzędzi i efektywnego zarządzania pamięcią. I właśnie w tym miejscu na scenę wkraczają **orkiestratory AI**.

Orkiestrator AI, jak pokazano na rysunku 3.1, służy jako centralny punkt koordynujący interakcje między modelem, narzędziami, magazynami pamięci, interfejsami API i innymi systemami zewnętrznymi. Zapewnia efektywne i kontrolowane działanie agentów AI.



Rysunek 3.1. Przykład warstwy orkiestratora AI w typowym środowisku programistycznym

Orkiestratory AI mogą pomóc w następujących obszarach:

- **Zarządzanie złożonością.** Procesy AI często obejmują wiele skoordynowanych kroków, takich jak pobieranie danych, wnioskowanie i wykonywanie działań. Orkiestratory automatyzują i strukturyzują te procesy, ułatwiając skalowanie i utrzymanie systemów.
- **Zwiększanie skalowalności.** Orkiestratory obsługują duże obciążenia poprzez dystrybucję zadań, buforowanie odpowiedzi i równoległe przetwarzanie, co jest kluczowe dla obsługi wielu użytkowników lub zadań wymagających dużej liczby tokenów.
- **Zapewnianie świadomości kontekstu.** Ponieważ modele językowe mają ograniczoną pamięć, orkiestratory integrują wektorowe bazy danych i systemy pamięci, by agenty mogły zachowywać informacje i dostarczać bardziej spójne, spersonalizowane doświadczenia.
- **Ułatwianie integracji narzędzi.** Poprzez zarządzanie wykonywaniem zadań i zapewnianie płynnej interakcji między modelami językowymi a zewnętrznymi narzędziami orkiestratory usprawniają korzystanie z interfejsów API, wyszukiwarek i baz danych.
- **Poprawa niezawodności i monitorowania.** Od rejestrowania zdarzeń po informacje zwrotne przekazywane przez człowieka orkiestratory oferują narzędzia do wykrywania błędów, zapobiegania halucynacjom i zapewniania bezpiecznego i niezawodnego działania systemów.

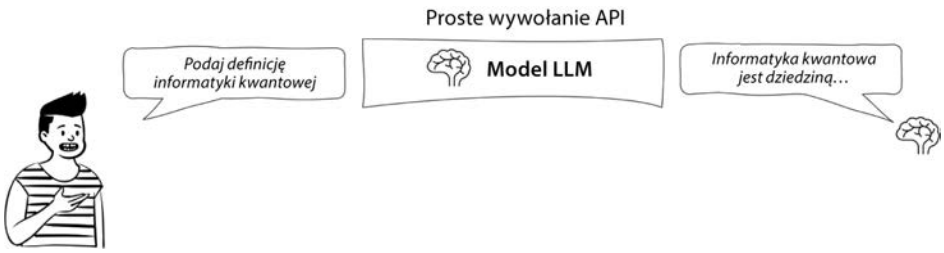
Aby lepiej zrozumieć potrzebę orkiestratora AI w kontekście agentów AI, musimy wprowadzić trzy istotne cechy tych ostatnich: autonomię, abstrakcję i modułowość.

Autonomia

Autonomia odnosi się do zdolności agenta AI do **samodzielnego** działania, podejmowania decyzji i wykonywania czynności bez interwencji człowieka. To zachowanie oparte na samostanowieniu umożliwia agentom AI wykonywanie zadań, dostosowywanie się do nowych sytuacji i dążenie do celów na podstawie zdobytych doświadczeń.

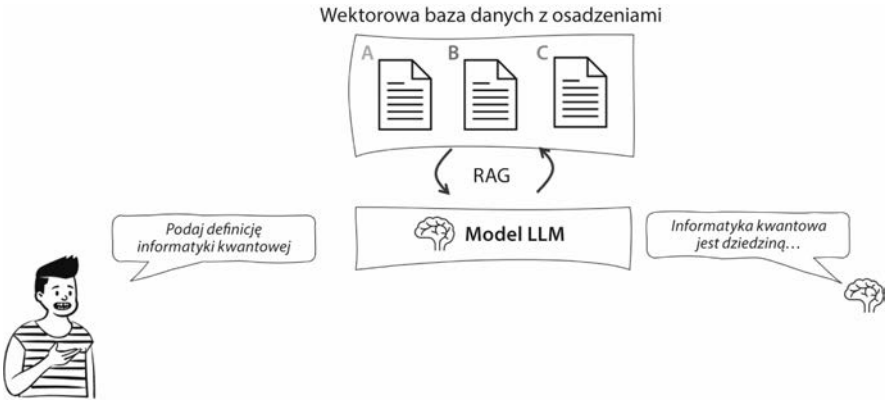
Autonomia agenta AI oznacza, że kroki, które będzie on wykonywać, niekoniecznie są znane z góry.

Na przykład rozważmy prosty, nieagentowy przepływ pracy przedstawiony na rysunku 3.2.



Rysunek 3.2. Przykład bezpośredniego wywołania API do modelu językowego

Gdy wysyłamy prompt do modelu LLM, wykonujemy wywołanie API, które stanowi jedyny krok w tym procesie. Nawet w przypadku **generowania wspomaganego wyszukiwaniem** (**RAG**) kroki są z góry określone, jak pokazano na rysunku 3.3.

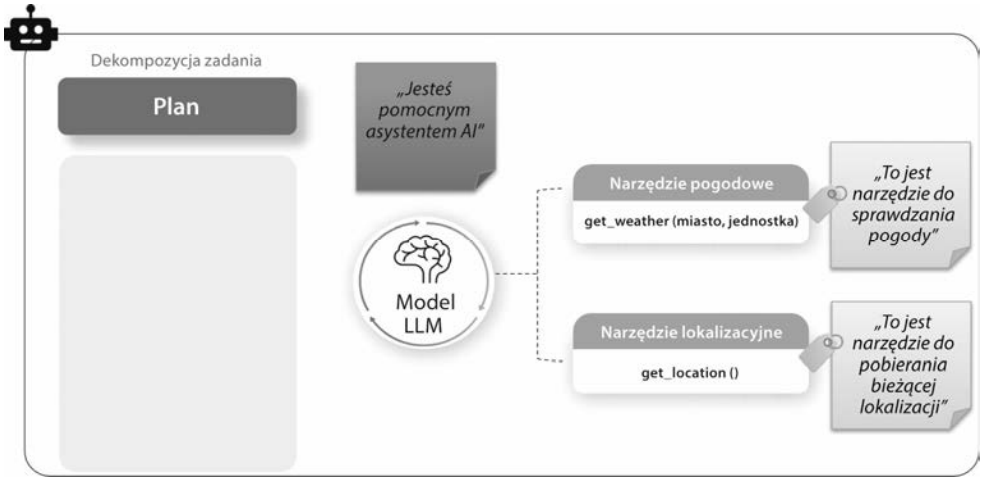


Rysunek 3.3. Przykład wzorca RAG

Rozważmy teraz podejście wykorzystujące autonomicznego agenta. Załóżmy, że nasz agent ma do dyspozycji dwa narzędzia:

- Narzędzie pogodowe — funkcję przyjmującą dwa parametry: miasto i jednostkę miary.
- Narzędzie lokalizacyjne — funkcję wykorzystującą bieżącą lokalizację użytkownika określaną na podstawie GPS-u. Ta funkcja nie pobiera żadnych parametrów.

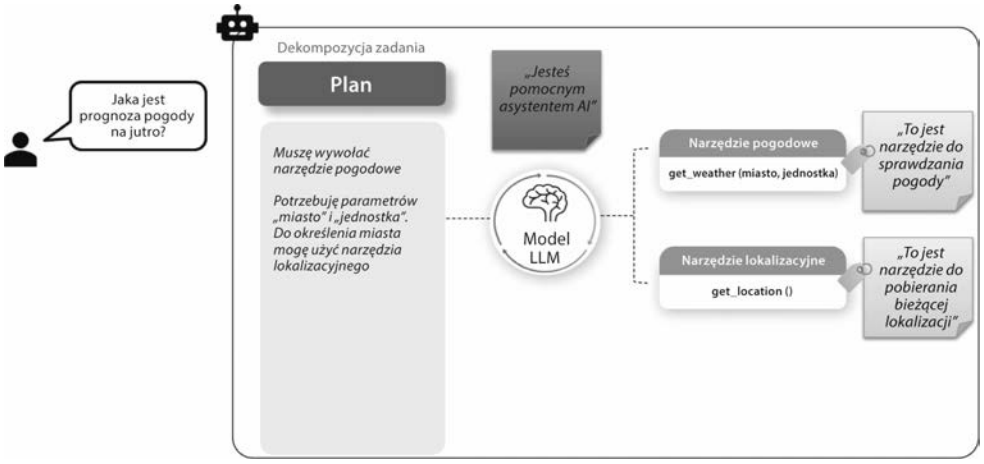
Obie funkcje są opatrzone opisem w języku naturalnym, zgodnie z anatomią agenta AI. Projekt naszego procesu, przedstawiony na rysunku 3.4, pozwoli agentowi zdecydować, którego narzędzia powinien użyć.



Rysunek 3.4. Przykład wzorca agentowego

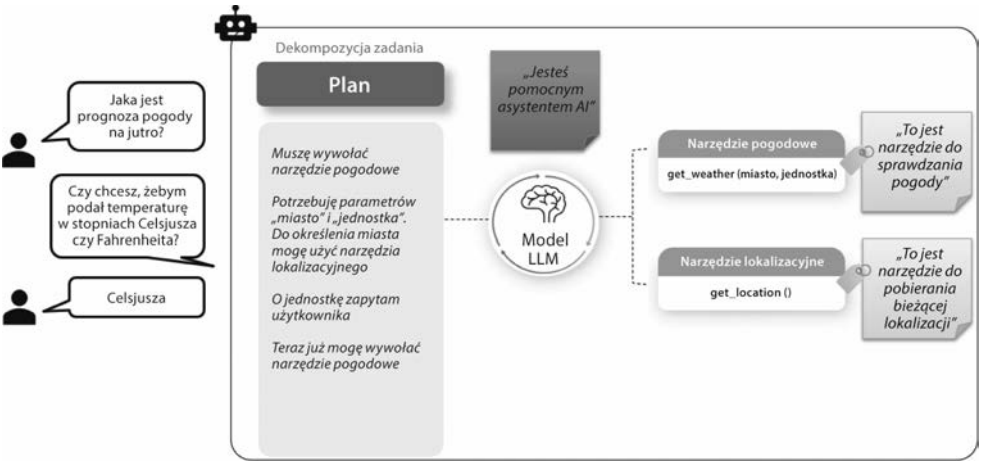
Załóżmy, że nowy użytkownik pyta: „Jaka jest prognoza pogody na jutro?". Spowoduje to następującą sekwencję zdarzeń:

1. Agent przeanalizuje opisy dostępnych narzędzi i zrozumie, że musi użyć narzędzia do sprawdzania pogody. Brakuje mu jednak dwóch parametrów, ale dzięki swojej autonomii może spróbować je pozyskać. Szybko orientuje się, że może wykorzystać narzędzie lokalizacyjne do uzyskania pierwszego parametru — wynik tej funkcji posłuży jako parametr dla narzędzia pogodowego (rysunek 3.5).



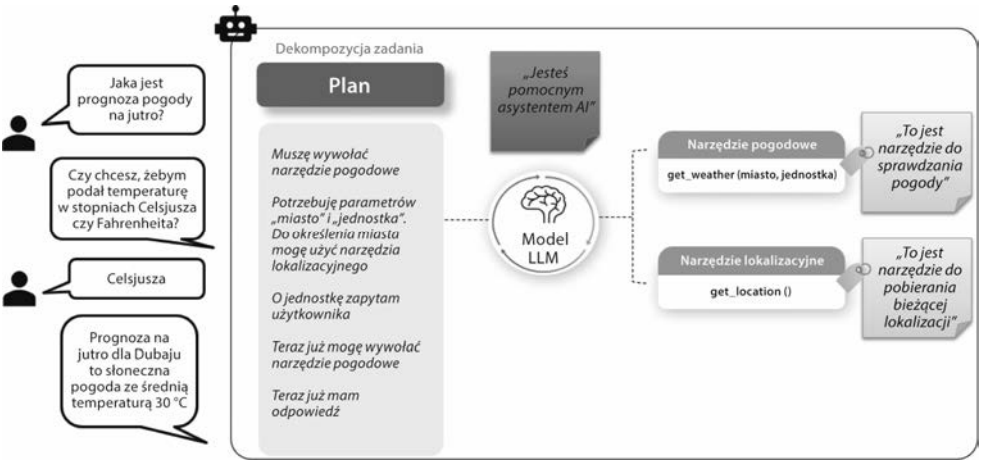
Rysunek 3.5. Przykład agenta AI wywołującego narzędzie w celu pobrania parametru

2. Drugiego parametru agent nie może ustalić samodzielnie — musi zapytać użytkownika. Dlatego pyta go, jakiej jednostki miary potrzebuje.
   Gdy użytkownik odpowie, agent będzie w stanie prawidłowo wywołać narzędzie pogodowe z obydwoma parametrami (jak pokazano na rysunku 3.6).



Rysunek 3.6. Przykład agenta AI pytającego użytkownika o brakujący parametr

3. Agent analizuje wynik zwrócony przez narzędzie pogodowe i ustala, że teraz już zna ostateczną odpowiedź, którą może przekazać użytkownikowi (rysunek 3.7).



Rysunek 3.7. Przykład wywołania narzędzia przez agenta AI z pobranymi parametrami

Czy wyobrażasz sobie, ile instrukcji warunkowych „if...else" byłoby potrzebnych, aby odtworzyć taki stopień autonomii w standardowym procesie opartym na **robotycznej automatyzacji procesów** (**RPA**)? A nawet gdybyśmy byli w stanie zarządzać podobnym scenariuszem, co by się stało, gdyby użytkownik zapytał o coś, co nie zostało wcześniej uwzględnione i zaimplementowane w procesie? Adaptacyjność, samoocena i umiejętność samodzielnego dostosowania się są kluczowymi cechami autonomii agentowej.

Możemy zapewnić naszym agentom różne stopnie autonomii; wszystko sprowadza się do ustalonego przepływu pracy i strategii planowania, którymi mają się kierować te agenty. Jak się przekonasz w dalszej części rozdziału, możemy zdefiniować plan, wedle którego agent będzie używał narzędzi w określonej kolejności, możemy zaplanować, aby agent powtarzał użycie narzędzia, aż osiągnie określony wynik, lub możemy dać mu całkowitą swobodę do korzystania ze wszystkich dostępnych narzędzi tyle razy, ile uzna za stosowne, zawsze gdy stwierdzi, że ich potrzebuje.

Zaprojektowanie odpowiedniego agentowego przepływu pracy jest kluczowym aspektem rozważań nad architekturą systemu agentowego podczas tworzenia jego stanu.

Abstrakcja i modularność

Abstrakcja polega na rozkładaniu i upraszczaniu złożoności. To ona sprawia, że systemy stają się zrozumiałe i skalowalne. Poza uproszczeniem umożliwia także projektowanie modularne, które jest fundamentalną zasadą w budowaniu inteligentnych systemów.

Modularność dzieli złożone problemy na mniejsze komponenty nadające się do wielokrotnego użytku, z których każdy zajmuje się konkretną częścią wyzwania. Takie podejście ma kilka zalet:

- **Wymienność.** Komponenty można zamieniać, ulepszać lub zastępować bez wpływu na cały system.
- **Ponowne wykorzystanie.** Dobrze zaprojektowane moduły można wykorzystać w różnych projektach, zwiększając efektywność.
- **Skalowalność.** Niezależne, ale płynnie zintegrowane komponenty ułatwiają rozbudowę rozwiązań.

W systemach wieloagentowych abstrakcja i modularność pozwalają na tworzenie współpracujących agentów, z których każdy specjalizuje się w wykonywaniu określonych zadań i które dynamicznie wchodzą ze sobą w interakcje. Odzwierciedla to ludzkie podejście do rozwiązywania problemów, polegające na ich dzieleniu, delegowaniu eliminowania poszczególnych podproblemów i współpracy, które pozwala skutecznie radzić sobie ze złożonością.

Doskonałym sposobem na zrozumienie abstrakcji i modularności w schematach agentowych jest przyjrzenie się wieloagentowemu systemowi zarządzania ruchem pojazdów w dużym mieście, gdzie różne poziomy agentów obsługują różne poziomy abstrakcji, zapewniając płynne działanie bez przeciążania żadnego pojedynczego elementu.

Uwaga

Systemy wieloagentowe zostaną bardziej szczegółowo opisane w rozdziale 7. Warto jednak zaznaczyć, że pojedynczy agent może zawsze zostać wykorzystany przez innego agenta jako „narzędzie", z zastosowaniem tego samego podejścia opartego na opisie jego możliwości w języku naturalnym. Na przykład „agent SQL" może być narzędziem dla „agenta zarządzającego projektem" w przypadku, gdy ten drugi będzie musiał wykonać zapytanie do bazy danych SQL.

Dlatego podczas omawiania systemów wieloagentowych — oraz w przykładach przedstawionych w dalszej części książki — warto myśleć o agentach jako o potencjalnych narzędziach dla innych agentów.

Na najbardziej szczegółowym poziomie mamy sterowniki skrzyżowań, które działają przy pojedynczych sygnalizacjach świetlnych lub skrzyżowaniach. Te agenty, działając w czasie rzeczywistym, wykorzystują dane z kamer i czujników, aby sterować sygnalizacją świetlną w zależności od natężenia ruchu pojazdów, ruchu pieszych i priorytetów pojazdów uprzywilejowanych.

Nie zajmują się one tym, co dzieje się w następnym kwartale miasta czy w szerszym krajobrazie miejskim; ich jedynym zadaniem jest optymalizacja przepływu ruchu w konkretnym miejscu. Jeśli nagle pojawi się duża liczba samochodów na skrzyżowaniu, mogą wydłużyć czas palenia się zielonego światła, aby zmniejszyć zator.

Patrząc w szerszej perspektywie, mamy koordynatorów ruchu nadzorujących poszczególne dzielnice. Te agenty nie zarządzają pojedynczymi sygnalizacjami, lecz analizują przepływ ruchu na wielu skrzyżowaniach w obrębie osiedla lub dzielnicy.

Wykorzystują one dane ze sterowników skrzyżowań, śledzenia GPS i systemów transportu publicznego do identyfikowania wzorców zatorów, przekierowywania pojazdów i równoważenia przepływu samochodów w danym obszarze. Jeśli wykryją nadmierne opóźnienia w jednej strefie, dostosowują czasy sygnalizacji dla wielu skrzyżowań, a nie tylko jednego.

Co ważniejsze, kierują one agentami na poziomie skrzyżowań, zapewniając, że ich działanie będzie zgodne z szerszymi celami ruchu w całej dzielnicy.

Na najwyższym poziomie mamy system zarządzania ruchem w całym mieście, odpowiedzialny za optymalizację przepływu milionów pojazdów w całym obszarze metropolitalnym. Ten agent nie skupia się na konkretnych sygnalizacjach świetlnych czy pojedynczych punktach zatorów. Zamiast tego alokuje zasoby, przewiduje długoterminowe wzorce i dokonuje strategicznych dostosowań.

Wykorzystując dane z prognoz pogody, harmonogramów dużych wydarzeń, wypadków i sieci transportu publicznego, ten agent może przekierowywać całe drogi, koordynować harmonogramy budów, by zminimalizować zakłócenia, lub, w przypadku poważnych incydentów, wdrażać plany reagowania kryzysowego obejmujące swym zasięgiem całe miasto.

Jeśli na głównej autostradzie dojdzie do wypadku, system miejski poinstruuje agenty na poziomie dzielnicy, nakazując im odpowiednie dostosowanie wzorców ruchu, a one z kolei poinstruują sterowniki skrzyżowań, aby efektywnie przekierowywały pojazdy.

Ta warstwowa struktura demonstruje siłę abstrakcji i modularności stanowiących kluczowe cechy systemów wieloagentowych:

- Agenty na skrzyżowaniach zajmują się lokalnymi decyzjami podejmowanymi w czasie rzeczywistym, dostosowując sygnalizację świetlną i nadając wyższy priorytet natychmiastowemu przepływowi ruchu.
- Agenty na poziomie dzielnicy analizują i koordynują grupy skrzyżowań, optymalizując ruch w szerszym obszarze.
- Agenty na poziomie miasta skupiają się na szerszej perspektywie — planują długoterminową wydajność, reagują na sytuacje awaryjne i optymalizują działanie całego systemu.

Odzwierciedla to sposób funkcjonowania architektur oprogramowania, systemów AI, a nawet struktur korporacyjnych w rzeczywistym świecie. Niezależnie od tego, czy mówimy o pracownikach pierwszej linii wykonujących zadania, menedżerach średniego szczebla koordynujących wysiłki, czy dyrektorach wyznaczających ogólną wizję, abstrakcja umożliwia złożonym systemom zachowanie skalowalności, wydajności i odporności.

Projektując wieloagentowe architektury AI z wykorzystaniem tego warstwowego podejścia, zapewniamy, że każdy agent koncentruje się tylko na tym, czym musi się zająć. Zapobiega to przeciążeniu systemu i umożliwia adaptacyjne podejmowanie decyzji w czasie rzeczywistym na dużą skalę, podobnie jak w inteligentnym systemie zarządzania ruchem w tętniącym życiem mieście.

Jeśli wydaje Ci się to nierealne, przyjrzyjmy się narzędziu OpenAI o nazwie Operator, które działa jako autonomiczny agent zdolny do wykonywania zadań w przeglądarce internetowej, takich jak rezerwacja biletów czy składanie zamówień online.

Operator OpenAI stosuje hierarchiczne podejście wieloagentowe podobne do systemu zarządzania ruchem. Każdy agent działa na innym poziomie abstrakcji, zapewniając wydajność i możliwości adaptacji bez przeciążania żadnego pojedynczego komponentu.

- **Kontrolery internetowe (agenty niskiego poziomu).** Te agenty zajmują się wykonywaniem poleceń: poruszaniem myszką, klikaniem przycisków i wprowadzaniem tekstu. Nie analizują ani nie planują — po prostu wykonują polecenia.
- **Agenty wizji i rozumowania (agenty średniego poziomu).** Te agenty interpretują interfejs internetowy. Agent wizji (ang. *Vision Agent*) przetwarza zrzuty ekranu, wykrywając istotne elementy, podczas gdy agent rozumujący (ang. *Reasoning Agent*) określa następną akcję (kliknięcie, wpisanie tekstu lub przewinięcie strony). Ta warstwa abstrahuje od szczegółów wykonania, skupiając się na zrozumieniu i podejmowaniu decyzji.
- **Planista/orkiestrator (agent wysokiego poziomu).** Agent najwyższego poziomu nadzoruje cały system, dbając o to, by interakcje internetowe były zgodne z szerszymi celami — czy to wyszukiwaniem informacji, czy wypełnianiem formularza. Deleguje zadania agentom średniego poziomu, zapewniając płynną i przemyślaną nawigację.

To ustrukturyzowane podejście podkreśla, że w projektowaniu systemów wieloagentowych abstrakcja ma kluczowe znaczenie:

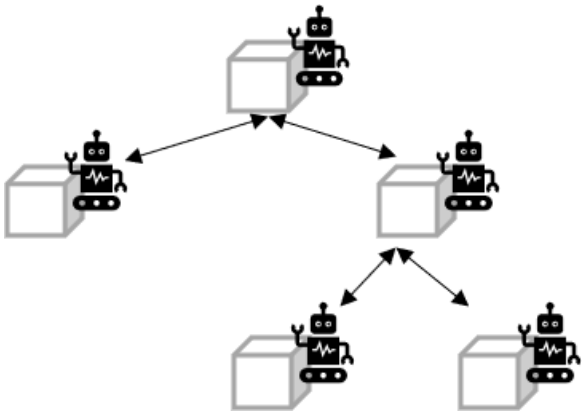
- agenty niższego poziomu wykonują zadania bez zastanawiania się nad decyzjami;
- agenty średniego poziomu koncentrują się na interpretacji i planowaniu;
- agenty wyższego poziomu zajmują się ogólną strategią, nie wchodząc w szczegóły techniczne.

Dzięki wykorzystaniu tej modułowej konstrukcji Operator OpenAI podczas obsługiwania różnych stron WWW dynamicznie dostosowuje się do nich bez konieczności ręcznego programowania. Ta skalowalna i uniwersalna architektura jest doskonałym przykładem tego, jak systemy wieloagentowe napędzają rzeczywiste aplikacje korzystające ze sztucznej inteligencji.

Z perspektywy architektonicznej wszystkie te komponenty (agenty, umiejętności, wtyczki) można postrzegać jako powtarzalne zasoby w organizacji. W tym kontekście orkiestratory AI zapewniają, że komponenty te współpracują ze sobą bez ścisłego powiązania, dzięki czemu zapobiegają przytłaczaniu systemów przez nadmierną złożoność.

Zgodnie z powyższym hierarchicznym przykładem (rysunek 3.8) przy użyciu orkiestratora AI można łatwo zdefiniować następujące elementy:

- **Agenty wykonawcze (niskiego poziomu).** Zajmują się podstawowymi zadaniami, takimi jak wywołania API, zapytania do baz danych lub pozyskiwanie treści ze stron WWW (ang. web scraping), wykonując polecenia bez podejmowania decyzji.
- **Agenty rozumujące (średniego poziomu).** Analizują dane, określają działania i wybierają odpowiednie narzędzia, abstrahując od szczegółów wykonania.
- **Orkiestracja i planowanie (wysoki poziom).** Orkiestrator nadzoruje przepływy pracy, rozbijając zadania na części, rozdzielając je między poszczególne agenty i dynamicznie dostosowując się do realizowanego celu.



Rysunek 3.8. Hierarchia agentów AI

Dzięki wykorzystaniu takiej struktury systemów AI orkiestratory umożliwiają uzyskanie adaptacyjnej i uniwersalnej inteligencji i zapewniają płynną interakcję między komponentami bez konieczności ręcznej interwencji podejmowanej przez człowieka.

Kluczowe elementy orkiestratora AI

Skoro wyjaśniliśmy już, dlaczego orkiestratory AI są tak skuteczne w zarządzaniu złożonością, skalowalnością, kontekstem i niezawodnością systemów agentowych, czas przyjrzeć się bliżej ich wewnętrznemu działaniu. Sercem każdego orkiestratora jest zestaw podstawowych komponentów, obejmujących wykonywanie przepływów pracy, zarządzanie pamięcią, integrację narzędzi, wykrywanie błędów i egzekwowanie zabezpieczeń. Każdy z tych elementów odgrywa kluczową rolę w zapewnieniu wydajnego i niezawodnego funkcjonowania agentów AI.

Zarządzanie przepływem pracy

Jedną z głównych funkcji orkiestratora AI jest zarządzanie ustrukturyzowanymi przepływami pracy. Przepływy pracy określają sposób wykonywania zadań — czy są one realizowane sekwencyjnie, równolegle, czy też z wykorzystaniem logiki warunkowej. Poniżej przedstawiamy najczęściej spotykane rodzaje przepływów pracy.

- **Przepływy sekwencyjne.** Zadania są wykonywane krok po kroku w z góry określonej kolejności. *Przykład*: agent AI do przetwarzania dokumentów najpierw wyodrębnia tekst z obrazów, następnie tworzy podsumowanie treści, a na końcu tłumaczy ją na inny język.
- **Przepływy równoległe.** Wiele zadań jest wykonywanych jednocześnie w celu optymalizacji wydajności. *Przykład*: agent AI do analizy finansowej może przetwarzać trendy wielu akcji jednocześnie, aby dostarczyć kompleksowy raport rynkowy.
- **Przepływy warunkowe.** Ścieżki wykonania zmieniają się w zależności od określonych warunków. *Przykład*: agent AI obsługi klienta może przekazać złożone zapytania do ludzkiego agenta, jeśli analiza nastrojów wykryje frustrację.
- **Przepływy hierarchiczne.** Zadania są zorganizowane w ustrukturyzowany, wielowarstwowy sposób, gdzie agenty AI wysokiego poziomu delegują podzadania do wyspecjalizowanych agentów. *Przykład*: agent AI zarządzania projektem nadzoruje przepływ prac inżynierskich, delegując zadania do agentów AI odpowiedzialnych za generowanie kodu, testowanie i wdrażanie, jednocześnie monitorując ogólny postęp.
- **Przepływy grupowego czatu.** Agenty AI współpracują w środowisku konwersacyjnym, wymieniając spostrzeżenia i dostosowując swoje działania na podstawie interakcji w czasie rzeczywistym. *Przykład*: zespół agentów AI (np.: asystent badawczy, bot sprawdzający fakty i model do tworzenia podsumowań) dynamicznie omawia temat, udoskonalając wyniki przed przedstawieniem końcowej odpowiedzi użytkownikowi.

Uwaga

Zarządzanie przepływem pracy idzie w parze z koncepcją autonomii, którą wprowadziliśmy w poprzednim punkcie rozdziału. Na przykład jeśli chodzi o przepływ pracy typu czat grupowy, zapewniasz swojemu systemowi wieloagentowemu wysoki stopień autonomii. Z drugiej strony sekwencyjny przepływ pracy jest bardziej przewidywalny, ponieważ jasno określa kolejność wywoływania agentów.

Orkiestratory AI dostarczają programistom narzędzia do projektowania, modyfikowania i optymalizowania tych przepływów pracy w sposób dynamiczny, co czyni je niezbędnymi do tworzenia skalowalnych i adaptowalnych aplikacji AI.

Obsługa pamięci i kontekstu

Skuteczne agenty AI potrzebują dostępu do historycznych interakcji i zewnętrznych baz wiedzy, koniecznych do dostarczania trafnych odpowiedzi i zachowania ciągłości. Orkiestratory zapewniają te możliwości przy wykorzystaniu różnych technik zarządzania pamięcią:

- **Pamięć krótkotrwała.** Przechowuje kontekst sesji, dzięki czemu agenty AI mogą pamiętać szczegóły trwającej rozmowy. *Przykład*: wirtualny asystent pamięta poprzednie pytanie użytkownika podczas sesji czatu.
- **Pamięć długotrwała.** Zachowuje wiedzę w dłuższych okresach; często są do tego celu używane wektorowe bazy danych. *Przykład*: system AI do zastosowań medycznych pamięta historię choroby pacjenta w celu dostarczania spersonalizowanych rekomendacji; historia ta może zawierać transkrypcje poprzednich wizyt, raporty medyczne, informacje o alergiach oraz przyjmowanych lekach.
- **Semantyczne buforowanie pamięci.** Gdy orkiestratory AI zarządzają pamięcią, stosują strategie buforowania w celu optymalizacji szybkości i efektywności wyszukiwania. Semantyczne buforowanie pamięci polega na przechowywaniu często wyszukiwanych informacji w sposób umożliwiający agentom AI przypominanie sobie faktów, pojęć i relacji bez konieczności odwoływania się do historii sesji. *Przykład*: agent AI do obsługi klientów może przypomnieć sobie wcześniejsze skargi użytkownika i szybciej odnaleźć ich rozwiązanie.

Definicja

W kontekście informatyki buforowanie to technika tymczasowego przechowywania danych w celu przyspieszenia dostępu do nich w przyszłości. Tradycyjnie aplikacje wymagające niskich opóźnień i wysokiej przepustowości korzystają z buforowania w pamięci, które polega na przechowywaniu danych bezpośrednio w pamięci operacyjnej komputera, umożliwiając szybkie pobieranie danych dzięki wysokiej prędkości RAM. Jednak buforowanie danych w pamięci zazwyczaj bazuje na wykorzystaniu dokładnych par klucz-wartość, co oznacza, że zapytanie musi dokładnie pasować do zapisanego klucza, aby możliwe było pobranie danych.

Wraz z pojawieniem się aplikacji opartych na modelach językowych wprowadzono nowy system buforowania: buforowanie semantyczne. Koncentruje się ono na znaczeniu i kontekście danych (wykorzystując osadzenia) zamiast na dokładnych dopasowaniach. Podejście to wymaga przechowywania wyników zapytań wraz z ich kontekstem semantycznym, co umożliwia systemowi rozpoznawanie i pobieranie właściwych danych, nawet jeśli nowe zapytanie nie jest dokładnym odpowiednikiem poprzedniego.

Dzięki efektywnemu zarządzaniu pamięcią orkiestratory AI zapewniają, że agenty dostarczają spójne, przemyślane i kontekstowo trafne odpowiedzi.

Integracja narzędzi i API

Agenty AI często wymagają dostępu do zewnętrznych zasobów, takich jak bazy danych, interfejsy API i narzędzia obliczeniowe. Orkiestratory ułatwiają bezproblemową integrację, umożliwiając agentom:

- Pobieranie na bieżąco danych z API (np. pobieranie prognozy pogody przez asystenta AI wspierającego podróżowanie).
- Dostęp i przeszukiwanie baz danych (np. pobieranie szczegółów zamówienia przez asystenta AI w e-commerce).
- Wykorzystanie zewnętrznych narzędzi obliczeniowych (np. użycie API uczenia maszynowego do wykrywania oszustw w aplikacjach bankowych).

Orkiestratory pozwalają na efektywne zarządzanie tymi integracjami, zapewniając, że agenty AI będą dysponować aktualnymi i dokładnymi informacjami.

Obsługa błędów i monitorowanie

Aby zapewnić niezawodność aplikacji AI, systemy zarządzające wdrażają solidne mechanizmy obsługi błędów i monitorowania:

- **Rejestrowanie i analityka.** Rejestrują szczegółowe dzienniki interakcji AI do celów debugowania i optymalizacji.
- **Automatyczne wykrywanie błędów.** Identyfikują nieudane procesy i automatycznie je ponawiają lub eskalują.
- **Śledzenie wydajności.** Monitorują czasy odpowiedzi, dokładność i ogólny stan systemu.
- **Integracja człowieka w pętli.** Umożliwiają ludzką weryfikację krytycznych decyzji. *Przykład*: asystent AI w medycynie przed postawieniem diagnozy wymaga potwierdzenia jej przez człowieka.

Dzięki proaktywnemu zarządzaniu błędami i kompleksowemu monitorowaniu systemy zarządzające AI pomagają utrzymać wysoką niezawodność i wiarygodność całego rozwiązania.

Bezpieczeństwo i zgodność

Bezpieczeństwo jest kluczowym priorytetem w systemach sztucznej inteligencji, szczególnie w przypadku przetwarzania wrażliwych danych. Orkiestratory AI wykorzystują wiele mechanizmów zabezpieczających, w tym następujące:

- **Uwierzytelnianie i kontrola dostępu.** Zapewnienie, że tylko autoryzowani użytkownicy i systemy mogą wchodzić w interakcje z agentami AI.
- **Ograniczanie liczby żądań.** Zapobieganie nadużyciom poprzez kontrolowanie liczby zapytań, które agent AI może przetworzyć w określonym czasie.
- **Zgodność z przepisami o ochronie danych.** Przestrzeganie regulacji takich jak RODO czy HIPAA poprzez bezpieczne zarządzanie danymi użytkowników.

- **Filtry bezpieczeństwa i przeciwdziałania stronniczości.** Wdrażanie zabezpieczeń zapobiegających stronniczym lub szkodliwym wynikom AI.

Mechanizmy bezpieczeństwa i zgodności pomagają zapewnić, że agenty AI działają bezpiecznie, etycznie i zgodnie z ramami prawnymi.

Kluczowe komponenty orkiestratora AI — zarządzanie przepływem pracy, obsługa pamięci, integracja narzędzi, wykrywanie błędów i bezpieczeństwo — tworzą fundament do budowy solidnych i wydajnych aplikacji AI. Dzięki wykorzystaniu tych możliwości programiści mogą tworzyć agenty AI, które są nie tylko potężne, ale także niezawodne, skalowalne i bezpieczne. Zrozumienie tych komponentów pozwala na podejmowanie lepszych decyzji przy wyborze lub projektowaniu platformy do orkiestracji AI.

Przegląd najpopularniejszych narzędzi do orkiestracji sztucznej inteligencji dostępnych na rynku

Wśród dostępnych orkiestratorów AI można wskazać kilka rozwiązań, które oferują unikalne możliwości dostosowane do różnych przypadków użycia. Niektóre z nich koncentrują się na modułowości i elastyczności, zapewniając programistom możliwości dostosowywania przepływów pracy, inne priorytetowo traktują przyjazne dla użytkownika interfejsy do szybkiego prototypowania. Przyjrzyjmy się zatem najpopularniejszym orkiestratorom AI, które były dostępne na rynku w maju 2025 r., zwracając uwagę na ich kluczowe mocne strony i idealne zastosowania.

- **LangChain.** To modułowa platforma zaprojektowana do tworzenia aplikacji opartych na modelach LLM. Zapewnia niezbędne komponenty do integracji zewnętrznych narzędzi, zarządzania pamięcią w interakcjach i definiowania przepływów pracy opartych na agentach. Jako projekt open source LangChain może się pochwalić obszerną dokumentacją i silną społecznością, co czyni go popularnym wyborem dla programistów chcących budować solidne aplikacje oparte na AI.
- **LlamaIndex (dawniej GPT Index).** LlamaIndex specjalizuje się w optymalizacji wyszukiwania danych dla modeli LLM, zapewniając efektywny dostęp zarówno do ustrukturyzowanych, jak i nieustrukturyzowanych źródeł danych. Jest szczególnie skuteczny w połączeniu z LangChain do budowania agentów AI opartych na wiedzy, które wymagają zaawansowanych możliwości wyszukiwania i indeksowania. Jego zdolność do łączenia dużych zbiorów danych z generatywną AI czyni go nieocenionym narzędziem dla organizacji obsługujących ogromne repozytoria informacji.
- **AutoGen.** AutoGen jest dostosowany do tworzenia wieloagentowych przepływów pracy AI i zapewnia agentom opartym na modelach LLM możliwości komunikacji i współpracy przy złożonych zadaniach. Dzięki automatyzacji interakcji pomiędzy jednostkami AI AutoGen ułatwia badania,

rozumowanie i generowanie treści, pozwalając systemom AI podejmować bardziej świadome decyzje poprzez prowadzenie ustrukturyzowanego dialogu. Jest dobrze dostosowany do aplikacji wymagających współpracy wielu wyspecjalizowanych agentów dążących do wspólnego celu.

- **Langflow.** Langflow upraszcza proces projektowania przepływów pracy agentów AI poprzez intuicyjny interfejs wizualny. Dzięki bezproblemowej integracji z LangChain i innymi narzędziami do orkiestracji umożliwia szybkie prototypowanie i wizualizację interakcji agentów w czasie rzeczywistym. To sprawia, że jest szczególnie przydatny dla programistów i badaczy, którzy chcą eksperymentować z automatyzacją opartą na AI bez zagłębiania się w skomplikowane implementacje kodu.
- **Semantic Kernel (SK).** Opracowany przez Microsoft Semantic Kernel łączy możliwości uczenia maszynowego z tradycyjnymi praktykami tworzenia oprogramowania. Wspiera podejście oparte na wtyczkach, pozwalając programistom integrować przepływy pracy oparte na AI z istniejącymi systemami biznesowymi. Semantic Kernel jest zaprojektowany do zwiększania produktywności poprzez bezpośrednie osadzanie automatyzacji opartej na AI w korporacyjnych środowiskach programistycznych.
- **LangGraph.** LangGraph wprowadza ustrukturyzowane podejście do współpracy wielu agentów, wykorzystując przepływy pracy oparte na grafach. Zapewnia framework do projektowania zaawansowanych interakcji agent-agent, gwarantując, że systemy AI komunikują się w zorganizowany i skalowalny sposób. To sprawia, że jest szczególnie cenny przy orkiestracji aplikacji AI, gdzie różne agenty muszą dynamicznie współpracować, aby rozwiązywać złożone problemy.

Skoro znasz już dostępne orkiestratory, musimy odpowiedzieć na kolejne kluczowe pytanie: jak wybrać odpowiedni orkiestrator dla tworzonego agenta AI?

Jak wybrać odpowiedni orkiestrator dla swojego agenta AI

Wybór orkiestratora AI zależy od kilku czynników, w tym złożoności aplikacji, wymaganego poziomu dostosowania, dostępnego ekosystemu i łatwości wdrożenia. Oto kluczowe kryteria, które warto wziąć pod uwagę przy wyborze orkiestratora:

- **Łatwość użycia i modułowość.** Jeśli szukasz szybkiego i modułowego sposobu integracji modeli językowych w aplikacjach, to ze względu na swoją dobrze udokumentowaną, elastyczną architekturę świetnym wyborem będzie **LangChain**. *Przykład*: start-up tworzący chatbota AI do obsługi klienta może użyć LangChain do szybkiego prototypowania i integracji z istniejącą bazą danych i interfejsami API.
- **Aplikacje wymagające przetwarzania dużych ilości danych.** Jeśli Twój agent AI w dużym stopniu polega na pobieraniu ustrukturyzowanych lub

nieustrukturyzowanych danych, to najlepszym wyborem będzie framework **LlamaIndex**, który jest zoptymalizowany pod kątem efektywnej integracji z zewnętrznymi źródłami wiedzy. *Przykład*: asystent prawny AI pobierający i analizujący orzecznictwo z wielu repozytoriów dokumentów skorzystałby z możliwości wyszukiwania LlamaIndex.

- **Przepływy pracy z wieloma agentami.** Jeśli Twoja aplikacja wymaga dynamicznej interakcji wielu agentów, to idealnymi rozwiązaniami do orkiestracji złożonych interakcji AI będą **AutoGen** lub **LangGraph**. *Przykład*: asystent AI do wspomagania prac badawczych, w ramach którego współpracuje ze sobą wielu agentów — jeden streszczający dokumenty, drugi sprawdzający fakty, a trzeci generujący raporty — mógłby z powodzeniem skorzystać z każdego z tych dwóch orkiestratorów.
- **Aplikacje AI klasy korporacyjnej.** Jeśli potrzebujesz silnej integracji i bezpieczeństwa na poziomie przedsiębiorstwa, **Semantic Kernel** jest dobrze dostosowany do środowisk opartych na produktach i rozwiązaniach firmy Microsoft i do ustrukturyzowanych przepływów pracy AI. *Przykład*: korporacyjne narzędzie analityczne oparte na AI, integrujące się z Microsoft Teams i SharePoint, mogłoby z powodzeniem korzystać z Semantic Kernel.
- **Wizualne projektowanie przepływów pracy.** Jeśli preferujesz tworzenie przepływów pracy AI w sposób niewymagający pisania żadnego kodu (ang. *no-code*) lub bardzo ograniczający jego ilość (ang. *low-code*), to skorzystaj z frameworka **Langflow**, który zapewnia intuicyjny interfejs użytkownika, świetnie nadający się do szybkiego prototypowania i debugowania interakcji prowadzonych przez agentów AI. *Przykład*: tworzący generator treści oparty na AI zespół marketingowy, który nie dysponuje głęboką wiedzą programistyczną, mógłby wykorzystać wizualny interfejs Langflow do szybkiego projektowania przepływów pracy.

Wybór orkiestratora AI powinien być zgodny z celami i wymaganiami technicznymi Twojego systemu AI. Różne orkiestratory mają różne mocne strony: jedne ułatwiają tworzenie rozwiązań modułowych, inne koncentrują się na skalowalności, współdziałaniu wielu agentów lub integracjach korporacyjnych. Zrozumienie tych różnic pomoże Ci wybrać najlepsze narzędzie dla Twojego konkretnego przypadku użycia.

Podsumowanie

Orkiestratory AI, poprzez zapewnianie niezbędnych ram do zarządzania przepływami pracy, integracji narzędzi i utrzymania wydajności, odgrywają kluczową rolę w rozwoju i wdrażaniu inteligentnych systemów. W miarę ewolucji aplikacji AI orkiestratory dbają o to, by agenty AI działały autonomicznie, radziły sobie ze złożonymi zadaniami i dostosowywały się do dynamicznych wymagań.

W tym rozdziale omówiliśmy podstawowe komponenty orkiestratorów AI, w tym zarządzanie przepływami pracy, obsługę pamięci i bezpieczeństwo. Przyjrzeliśmy się również niektórym najpopularniejszym obecnie orkiestratorom, z których każdy oferuje unikalne mocne strony dostosowane do konkretnych przypadków użycia.

Wybór odpowiedniego orkiestratora AI zależy od różnych czynników, takich jak potrzeby integracyjne, skalowalność i złożoność przepływu pracy. Zrozumienie ich kluczowych funkcji pozwala programistom i firmom podejmować świadome decyzje przy wyborze narzędzia do orkiestracji, które będzie najlepiej odpowiadało ich celom.

Od następnego rozdziału zaczniemy szczegółowo analizować niektóre najbardziej interesujące komponenty agentów AI, zaczynając od zarządzania pamięcią i kontekstem.

Bibliografia

- *AutoGen*, Microsoft, *https://www.microsoft.com/en-us/research/project/autogen/* [dostęp: 27.02.2026 r.].
- *Balance agent control with agency*, LangGraph, *https://www.langchain.com/langgraph* [dostęp: 27.02.2026 r.].
- Langflow, *https://www.langflow.org/* [dostęp: 27.02.2026 r.].
- *Przedstawiamy agenta Operator*, OpenAI, *https://openai.com/index/introducing-operator/* [dostęp: 27.02.2026 r.].
- *Semantic Kernel* (SK), GitHub, *https://github.com/microsoft/semantic-kernel* [dostęp: 27.02.2026 r.].
- *Ship reliable agents*, LangChain, *https://www.langchain.com/* [dostęp: 27.02.2026 r.].
- *The new standard for complex document processing*, LlamaIndex (dawniej GPT Index), [dostęp: 27.02.2026 r.].

Od teorii do praktyki — buduj inteligentne agenty AI

Agenty AI rewolucjonizują sposób, w jaki wchodzimy w interakcje z technologią, przekształcając pasywne modele językowe w autonomiczne systemy zdolne do planowania, rozumowania i wykonywania złożonych zadań. W dobie gwałtownego rozwoju sztucznej inteligencji agenty AI stają się kluczowym elementem cyfrowej transformacji przedsiębiorstw, oferują bowiem bezprecedensowe możliwości automatyzacji procesów biznesowych i personalizacji usług. Ta technologia, która jeszcze niedawno była domeną laboratoriów badawczych, dziś znajduje praktyczne zastosowanie w każdej branży — od obsługi klienta po zarządzanie finansami.

Książka stanowi przewodnik po projektowaniu, implementacji i wdrażaniu systemów agentowych AI w środowiskach produkcyjnych. Zawiera szczegółowe omówienie architektury agentów, ich kluczowych komponentów: pamięci, narzędzi, orkiestracji, a także praktycznych aspektów budowania aplikacji wieloagentowych. Prezentuje nowoczesne frameworki, jak LangChain i LangGraph, protokoły komunikacji między agentami (MCP, A2A, ACP), kluczowe zagadnienia etyczne i bezpieczeństwa. Nie brakuje tu również konkretnych przykładów implementacji, od prostych asystentów po złożone systemy współpracujących agentów, wraz z metodami oceny ich wydajności i niezawodności.

- Praktyczne budowanie agentów AI z wykorzystaniem LangChain i LangGraph — od prostych asystentów po złożone systemy wieloagentowe
- Architektura i komponenty agentów: pamięć, narzędzia, orkiestracja, bazy wiedzy i strategie zarządzania kontekstem
- Protokoły komunikacji nowej generacji (MCP, A2A, ACP) umożliwiające współpracę między agentami różnych platform
- Systemy wieloagentowe: wzorce projektowe, hierarchie, przepływy pracy i koordynacja działań
- Odpowiedzialna AI: etyka, bezpieczeństwo, filtry treści, zabezpieczenia i zgodność z regulacjami
- Monitorowanie i ocena agentów z wykorzystaniem LangSmith i techniki optymalizacji wydajności
- Przyszłość sieci agentowej (NLWeb) i transformacja internetu w kierunku współpracy człowiek – agent

Valentina Alto jest architektką techniczną specjalizującą się w sztucznej inteligencji w Microsoft Innovation Hub w Dubaju. Jako specjalistka do spraw rozwiązań koncentruje się na obciążeniach związanych z danymi i AI w branżach: produkcyjnej, farmaceutycznej i detalicznej. Jest twórczynią tekstów technicznych, prelegentką, a także autorką dwóch książek o generatywnej AI i dużych modelach językowych.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-3663-8
9 788328 936638
Cena: 89,00 zł
packt