여행플래너 아키텍처 설계

2025. 6. 22

목차

1. 시스템 구성

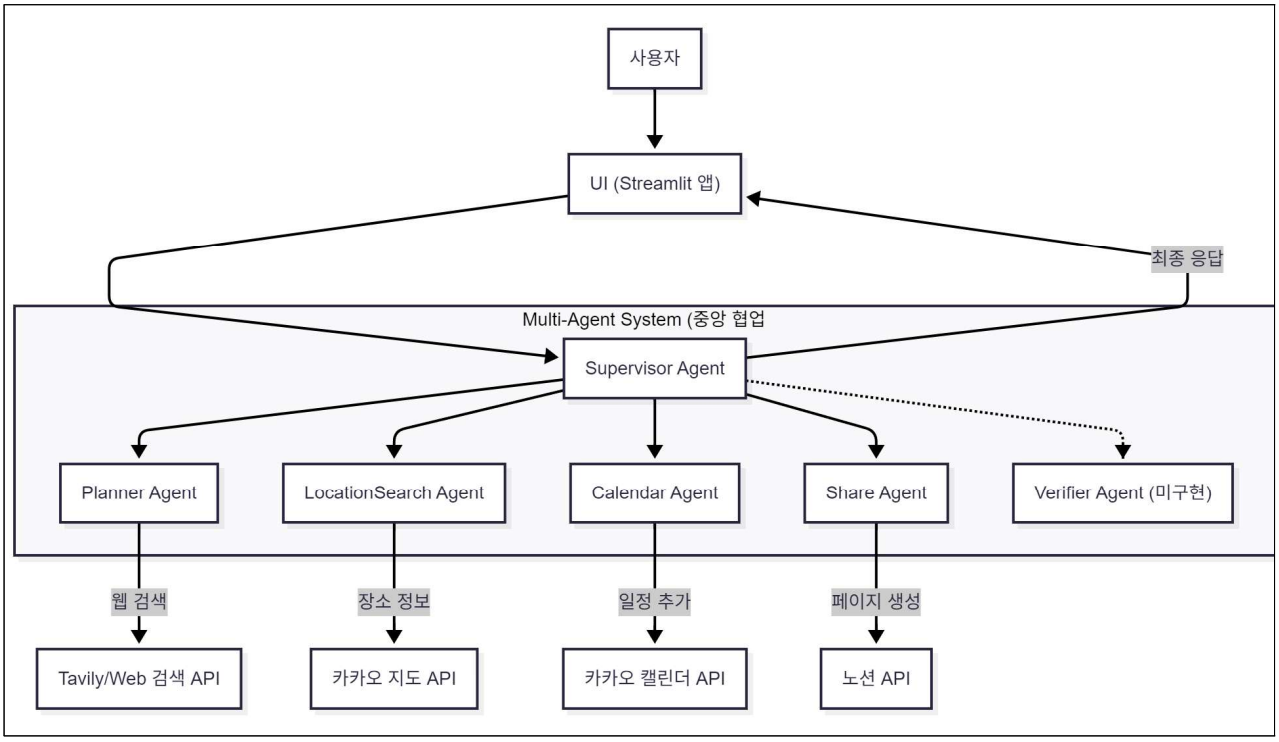
1-1. 개요

여행플래너 시스템은 멀티 에이전트 아키텍처로 구성하였으며, 총 5개의 에이전트들이 중앙 슈퍼바이저(Supervisor) 에이전트의 조율 아래 협력하여 동작한다. Supervisor는 사용자의 요청을 받아 가장 적합한 작업자 Worker 에이전트에게 작업을 할당하고, 작업 완료 후 결과를 다시 수집하는 중앙 집중형구조를 갖는다. 이러한 Supervisor-Worker 패턴을 통해 각 에이전트는 고유한 역할에 집중하면서도, 중앙 관리자를 통해 제어된다.

1-2. 주요 구성 요소

Agent	설명
Supervisor Agent	사용자 요청을 분석하여 적절한 워커 에이전트를 선택하고 전체 워크플로우를 조율
Planner Agent	여행 일정 초안을 생성하고 수정・제안하는 역할 수행
LocationSearch Agent	여행지의 특정 장소 정보나 주변 시설 정보를 검색
Calendar Agent	최종 확정된 여행 일정을 사용자 캘린더에 등록하고 관리
Share Agent	완성된 여행 계획을 정리하여 노션(Notion) 페이지로 생성
Verifier Agent	여행 계획에 활용된 정보의 신뢰성을 검증하여 LLM의 오류나 환각 정보를 최소화하는 역할 (추후 구현 예정)

1-3. 시스템 구성도



위 다이어그램은 위의 설명한 아키텍처를 개략적으로 나타낸 시스템 구성도이다. 사용자로부터 입력을 받는 프론트엔드(UI)부터 시작하여, 중앙 Supervisor와 각 도메인 에이전트들, 그리고 외부 서비스와의 연결 관계를 도식화하였다. 각 화살표는 상호 작용를 의미한다.

2. 에이전트 통신

2-1. 통신 메커니즘

에이전트들 간의 상호 작용은 명시적인 메시지 전달 방식을 따른다. 중앙 Supervisor가 각 워커 에이전트에 지시를 내려 작업을 요청하고, 워커가 결과를 산출하면 Supervisor에게 응답 형태로 결과를 반환한다. 이러한 통신은 일반적인 자연어 기반의 방식이 아니라, 시스템적으로 정의된 명령 객체를 통해 이뤄지므로 정확한 흐름 제어가 가능하다.

2-2. Handoff 기반 설계 및 구현

본 시스템에서는 LangGraph 프레임워크의 Command 객체를 활용한 핸드오프(Handoff) 방식을 통해 에이전트 간 통신을 구현한다. Supervisor 에이전트는 각 워커에게 작업을 위임하는 전용 명령 도구(create_handoff_tool)를 갖고 있으며, 이를 호출함으로써 특정 워커 에이전트를 실행한다. 워커 에이전트가 호출되면 Command 객체가 Supervisor의 상태 컨텍스트를 필요한 정보로 업데이트하고, 제어권을 해당 워커로 이동시킨다. 워커의 처리가 완료되면 다시 Supervisor에게 제어권이 반환되어 다음 단계로 넘어가거나 결과가 취합된다. 이러한 방식은 단순 키워드 매칭에 의존한 모호한 라우팅을 피하고, 명확한 작업 흐름과 안정적인 메시지 전달을 보장하는 장점이 있다.

에이전트 간 인터페이스는 LangGraph가 제공하는 그룹 채팅 형태의 환경에서 이루어지며, 각 에이전트는 자신에게 할당된 Tool(기능 모듈)을 통해 필요한 작업을 수행한다. 예를 들어 Supervisor는 transfer_to_* 형태의 도구들을 통해 특정 에이전트를 호출하고 PlannerAgent는 web_search_tool 등을 통해 웹 검색 기능을 사용하는 식이다.

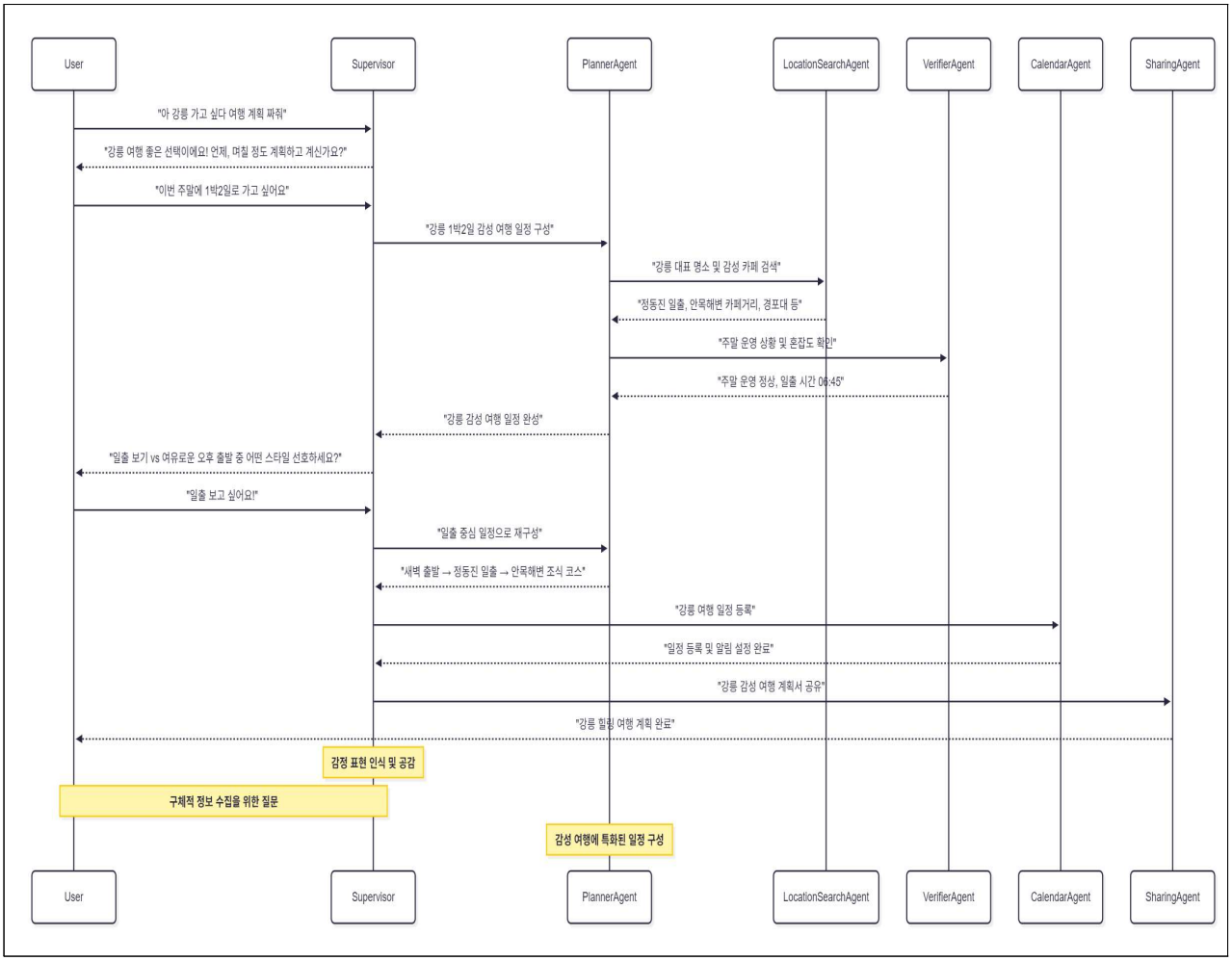
이러한 도구 기반 상호작용은 각 에이전트가 제한된 인터페이스로만 통신하도록 하여, 예측 불가능한 자유 대화 대신 통제된 형태의 메시지 교환을 가능하게 한다.

3. 데이터 흐름

3-1. 사용자 입력 및 처리

사용자가 여행 계획에 대한 요청을 UI를 통해 자연어로 입력하면, 해당 입력이 가드레일 필터링 후 Supervisor 에이전트에 전달된다. Supervisor 에이전트는 입력된 프롬프트를 분석하여 사용자의 의도와 요구사항을 파악한다. 필요한 경우 추가 정보를 얻기 위해 사용자와 추가 질의 응답을 수행할 수도 있다. (예: 여행 희망 지역, 기간, 선호 활동 등에 관한 질문)

이렇게 초기 질의 단계가 끝나면, Supervisor는 요청을 처리하기 위한 적절한 전략을 세우고 적절한 하위 에이전트에 작업을 분배하기 시작한다.



3-2. 멀티 에이전트 초기 처리

요청이 들어오면 여행 계획을 수립하기 위해 Supervisor가 일련의 에이전트를 순차적으로 호출한다. 먼저 Supervisor는 PlannerAgent를 통해 여행 일정의 초안을 작성한다. PlannerAgent는 웹 검색을 포함한 다양한 방법으로 기본 일정을 만들고, 이를 다시 Supervisor에게 전달한다. Supervisor는 전달받은 일정 초안을 평가하고, 더 구체적인 장소 정보 등 추가 정보가 필요하면 LocationSearchAgent를 호출한다. 이 과정에서 LocationSearchAgent는 카카오맵 API를 활용해 관광지나 맛집 등의 상세 정보를 수집하여 Supervisor에게 전달한다.

이처럼 Supervisor는 여러 에이전트를 단계적으로 실행하면서 필요한 데이터를 점진적으로 축적한다. 만약 사용자의 요구에 맞춰 일정 수정이나 보완이 필요하다면 PlannerAgent를 다시 호출하기도 한다. 충분한 일정과 세부 정보가 모이면 Supervisor는 최종 일정을 확정한다. 만약 정보의 정확성이나 현실성을 추가적으로 검증할 필요가 있다면, VerifierAgent가 구현되어 있는 경우 최종 확정 전에 검증 과정을 수행하여 오류나 비현실적인 일정이 없는지 다시 한번 확인하게 된다.

3-3. 결과 통합 및 응답

여행 계획의 모든 구성이 완료되면 Supervisor는 CalendarAgent를 호출하여 최종 일정을 사용자 캘린더에 등록한다. CalendarAgent는 확정된 일정을 카카오 캘린더 API를 통해 사용자의 캘린더에 이벤트로 추가하고, 처리 결과를 Supervisor에게 보고한다. 이어서 Supervisor는 ShareAgent를 실행하여 완성된 여행 일정을 보기 좋은 형태의 노션 페이지로 생성한다. ShareAgent는 Notion API를 이용해 여행 계획 페이지를 작성하고, 사용자가 공유할 수 있도록 링크를 만들어 Supervisor에게 전달한다.

마지막으로 Supervisor는 여행 일정의 요약 정보와 추천 장소, 상세 일정, 예약 사항(있을 경우) 및 노션 공유 링크를 포함하여 최종 답변을 작성한다. 이 최종 결과는 UI를 통해 사용자에게 자연어 형태로 제공되며, 캘린더에 등록된 일정 정보와 여행 계획을 담은 노션 링크와 같은 추가 리소스도 함께 전달된다.

예시) 서울 2박 3일 여행 계획을 세워주세요
(1) Supervisor가 요청을 분석하여 PlannerAgent 호출
(2) PlannerAgent가 기본 일정을 생성 후 보고
(3) Supervisor가 추가 정보 위해 LocationSearchAgent 호출
(4) LocationSearchAgent가 세부 장소 정보를 수집 후 보고
(5) Supervisor가 일정 확정 후 CalendarAgent 호출
(6) CalendarAgent가 일정을 캘린더에 등록 후 보고
(7) Supervisor가 마지막으로 ShareAgent 호출
(8) ShareAgent가 노션 페이지를 생성하고 링크를 반환하면 완료. 최종적으로 사용자에게 완성된 일정과 공유 링크가 응답

4. 사용 기술 스택

이 시스템 구현에 사용된 주요 기술 스택은 다음과 같다. 각 구성 요소는 해당 역할에 최적화된 기술과 서비스로 구성되어 있다.

기술스택	설명
오케스트레이션 프레임워크	시스템 내 다중 에이전트 간의 제어 흐름은 LangGraph를 통해 관리한다. LangGraph는 에이전트 간의 Handoff 라우팅과 Command 기반의 명확한 통신 방식을 지원하여 안정적이고 효율적인 에이전트 협력을 가능하게 한다. 또한 각 에이전트가 사용하는 LLM 인터페이스는 LangChain을 활용해 추상화하여, 다양한 모델을 간편하게 연동할 수 있도록 설계하였다.
LLM	시스템의 주된 자연어 처리는 OpenAI의 GPT-4o-mini 모델로 이루어진다. GPT-4o-mini는 GPT-4 계열의 경량 모델로, 빠르고 효과적으로 여행 계획을 생성하는 데 적합하다. 특히 Supervisor와 같은 중요한 에이전트에는 보다 강력한 성능을 가진 GPT-4o 모델을 별도로 적용하는 등, 각 에이전트의 요구사항에 따라 모델을 유연하게 선택하여 활용할 수 있는 구조이다.
웹 검색 및 지도	사용자에게 최신 여행 정보를 제공하기 위해 Tavily API와 카카오맵 API를 사용한다. Tavily API는 다양한 데이터를 검색해 제공하며, 카카오맵 API는 관광지, 맛집 등 구체적이고 상세한 지역 정보를 제공한다. PlannerAgent와 LocationSearchAgent는 이러한 API를 활용하여 사용자 맞춤형 여행 일정을 구성한다.
캘린더	최종 확정된 일정은 카카오 캘린더 API를 통해 사용자의 캘린더에 이벤트로 등록된다. CalendarAgent는 여행 계획을 사용자의 카카오톡 캘린더에 추가하고, 일정 관리와 수정 기능을 제공한다.
콘텐츠 공유	여행 일정은 Notion API를 통해 노션 페이지로 생성 및 공유된다. ShareAgent는 완성된 여행 계획을 보기 좋게 정리한 후, 공유 가능한 링크를 생성하여 사용자가 쉽게 접근할 수 있도록 지원한다.
사용자 인터페이스	사용자와의 상호작용을 위한 웹 UI는 Streamlit을 사용해 구현하였다. 현재는 프로토타입 및 데모용으로 간편하고 직관적인 대화형 인터페이스를 제공하며, 사용자가 쉽게 질문을 입력하고 여행 플랜을 받을 수 있다. 향후 제품 출시 단계에서는 전문적인 프론트엔드 또는 모바일 앱으로 확장될 수 있다.

5. 향후 확장 및 보안

5-1. 확장 가능성

멀티 에이전트 구조의 장점은 새로운 기능이나 요구사항에 유연하게 대응할 수 있다는 점이다. 예를 들어, 정보의 신뢰성을 검증하는 VerifierAgent를 도입하거나 예산 관리 기능을 제공하는 BudgetingAgent를 신설하는 등, 시스템의 기능을 확장하는 것이 용이하다. 각 에이전트는 표준화된 인터페이스로 통신하기 때문에 에이전트를 추가하거나 제거할 때 시스템 전체의 혼란을 최소화할 수 있다.

또한, LLM 모델을 구성 파일로 관리하기 때문에 더 성능이 좋은 모델로의 전환이나 에이전트별로 다른 모델을 적용하는 것도 간편하다. 예를 들어, 현재 사용하는 GPT-4o-mini 모델 대신 GPT-4 등 보다 고성능 모델이 등장할 경우, 구성 파일 변경만으로 쉽게 전환 가능하며, 이를 통해 시스템 성능과 응답 품질을 개선할 수 있다.

시스템의 수평 확장 또한 가능하다. 현재는 하나의 프로세스 내에서 Supervisor와 에이전트들이 동작하지만, 필요에 따라 각 에이전트를 별도 서비스로 분리하여 마이크로서비스 아키텍처 형태로 확장할 수 있다. 이 경우 각 에이전트를 독립된 서버나 컨테이너에서 운영하며, 메시지 큐 또는 gRPC와 같은 방식으로 Supervisor와 통신함으로써 부하 분산과 병렬 처리를 강화할 수 있다. 이를 통해 사용자의 요청이 증가해도 시스템이 효율적으로 대응할 수 있게 된다.

5-2. 보안

해당 시스템은 사용자의 개인정보와 API 인증 정보의 보안을 철저히 관리한다. API 키와 같은 민감한 정보는 코드에 직접 기록하지 않고 환경 변수(.env 파일)로 별도 관리하여 외부 노출을 방지한다. 예를 들어 카카오 API나 노션 API 키는 배포 단계에서만 주입되고 소스코드 저장소에는 저장하지 않는다.

마지막으로, LLM 기반의 서비스 특성상 AI 윤리 및 응답 안정성을 유지하기 위해 OpenAI 등에서 제공하는 콘텐츠 필터 정책을 준수하며 부적절한 응답이 발생하지 않도록 프롬프트 가드레일을 설계했다. 현재는 입력 데이터를 먼저 가드레일 프롬프트를 통해 필터링해 sql 인젝션과 같은 부적절한 입력은 막는다. 향후 VerifierAgent가 추가되면 응답 내용의 사실 여부뿐 아니라 부적절한 콘텐츠까지 필터링하여 서비스의 신뢰성을 더욱 높일 계획이다.

본 아키텍처는 멀티 에이전트의 명확한 협업 구조와 최신 외부 서비스 활용을 통해 사용자에게 실질적이고 가치 있는 서비스를 제공하도록 설계되었다. 개발자와 관리자는 이를 통해 시스템의 구조와 특성을 명확히 이해하고, 요구사항 변화에도 유연하게 대응할 수 있다.