ChatGPT

Lộ Trình Học ADK, MCP, A2A và AP2 (Kèm Ý Tưởng Dự Án Thực Hành)

Giới Thiệu

Cùng với sự trỗi dậy của **AI agent**, hàng loạt công nghệ mới như **ADK**, **MCP**, **A2A** và **AP2** đã xuất hiện để giúp các tác nhân AI (agent) hoạt động thông minh và tương tác hiệu quả. Tuy nhiên, các thuật ngữ viết tắt này có thể gây nhầm lẫn vì chức năng khác nhau của chúng. Trong hướng dẫn này, chúng ta sẽ **vạch ra lộ trình học tập** cho từng công nghệ trên, đồng thời đề xuất một **dự án thực hành liên quan đến hệ thống thanh toán** (payment gateway) để áp dụng cả bốn công nghệ từ cơ bản đến nâng cao. Mục tiêu là giúp bạn nhanh chóng nắm vững **bức tranh toàn cảnh** và hiểu cách các công nghệ này **phối hợp với nhau** trong môi trường on-premise (tại chỗ), phù hợp với kinh nghiệm Python/Django sẵn có của bạn. Hãy bắt đầu bằng việc tìm hiểu khái quát từng công nghệ.

Tổng Quan Về ADK, MCP, A2A, AP2

- **ADK (Agent Development Kit)**: Bộ khung phát triển tác nhân do Google phát hành, cung cấp một toolkit mã nguồn mở, linh hoạt để xây dựng và triển khai các AI agent. ADK không phụ thuộc mô hình hay hạ tầng triển khai cụ thể, tương thích với nhiều framework khác, giúp việc phát triển tác nhân **giống phát triển phần mềm** hơn (dễ thử nghiệm, version control) [1] [2]. ADK tối ưu cho mô hình Gemini của Google nhưng cho phép dùng bất kỳ LLM nào, đồng thời hỗ trợ sẵn các công cụ tích hợp để mở rộng khả năng tác nhân (vd. tìm kiếm Google, hàm tùy chỉnh) [3].

- **MCP (Model Context Protocol)**: Giao thức **ngữ cảnh mô hình** do Anthropic đề xuất, là một tiêu chuẩn mở nhằm chuẩn hóa cách ứng dụng AI (như Claude, ChatGPT) **kết nối tới các nguồn dữ liệu, công cụ và hệ thống bên ngoài**. MCP đóng vai trò như cổng **USB-C cho AI**, cho phép agent truy cập tệp tin, cơ sở dữ liệu, API... một cách thống nhất [4] [5]. Nói cách khác, MCP định nghĩa chuẩn **agent-to-tool** – cách một tác nhân gọi các công cụ hoặc dịch vụ bên ngoài để lấy thông tin hoặc hành động.

- **A2A (Agent2Agent Protocol)**: Giao thức **tác nhân-tới-tác nhân** do Google đề xuất (đã đóng góp cho Linux Foundation) [6]. A2A là một tiêu chuẩn mở giúp các agent AI **giao tiếp và cộng tác trực tiếp với nhau** một cách liền mạch, bất kể chúng được xây dựng trên nền tảng hay ngôn ngữ gì [7] [8]. A2A định nghĩa cách thức để agent **khám phá, trao đổi nhiệm vụ, đàm phán và chia sẻ kết quả** thông qua thông điệp chuẩn. Điều này tạo ra một "ngôn ngữ chung" cho các agent, tương tự như Internet dành cho AI agent, **phá vỡ silo** giữa các hệ thống khác nhau [8] [9].

- **AP2 (Agent Payments Protocol)**: Giao thức **thanh toán tác nhân** do Google vừa giới thiệu (09/2025), là một khung thanh toán mở cho phép các agent AI **thực hiện giao dịch mua bán một cách tự động** [10]. AP2 mở ra phương thức thanh toán an toàn cho agent bằng cách sử dụng các chứng thư mật mã để xác nhận **ủy quyền và ý định** của người dùng trong suốt quá trình "intent →

cart → payment” [11]. Cụ thể, AP2 mở rộng các giao thức A2A và MCP, định nghĩa cách agent, merchant và cổng thanh toán trao đổi **bằng chứng xác thực** (như chữ ký số) để đảm bảo giao dịch do agent thực hiện là đúng ý người dùng và có thể audit sau này [11]. Điều này giúp **giải quyết khoảng trống niềm tin** khi agent tự động “bấm nút thanh toán” thay cho con người, ngăn chặn các tranh chấp về sau bằng chuỗi bằng chứng rõ ràng [12].

Sơ đồ mô tả cách các công nghệ kết hợp trong hệ sinh thái agent: Mỗi tác nhân AI (có thể được xây dựng bằng ADK) kết nối với công cụ/API của mình thông qua MCP, và giao tiếp với các tác nhân khác (hoặc người dùng) thông qua giao thức A2A [13]. AP2 được xây dựng trên nền A2A/MCP để thêm lớp giao dịch thanh toán an toàn giữa agent với hệ thống merchant và mạng lưới thanh toán.

Giải thích sự kết hợp: Các công nghệ trên không thay thế mà **bổ trợ lẫn nhau** trong một hệ thống agent hoàn chỉnh. MCP làm cho từng agent **có năng lực hành động** (truy cập được công cụ, dữ liệu cần thiết) [14], còn A2A làm cho các agent **có thể hợp tác** (trao đổi nhiệm vụ và kết quả một cách tiêu chuẩn) [9]. ADK cung cấp bộ khung triển khai thực tế, giúp lập trình viên dễ dàng xây dựng các agent tuân thủ các chuẩn giao thức như A2A và tích hợp công cụ (MCP) một cách nhất quán [15] [16]. Trên nền tảng đó, AP2 bổ sung **lớp thanh toán**: Đảm bảo bất kỳ giao dịch nào do agent khởi xướng đều được ràng buộc bằng chứng ủy quyền của người dùng, có chữ ký của các bên liên quan và tuân theo quy trình chuẩn nhằm **đảm bảo tính an toàn, hợp lệ** cho mọi phía [11] [12]. Nói ngắn gọn, nếu MCP giúp agent **kết nối được công cụ**, A2A giúp nhiều agent **kết nối được với nhau**, thì AP2 đảm bảo agent **mua bán được an toàn**.

Lộ Trình Học Tập và Phát Triển Dự Án

Dưới đây là lộ trình từng bước để bạn vừa học các công nghệ, vừa xây dựng một **dự án mẫu về hệ thống thanh toán** tích hợp AI agent. Dự án đề xuất xoay quanh việc tạo ra một **“trợ lý mua sắm tự động”**: agent có thể tìm sản phẩm, đàm phán với một agent merchant, sau đó tự động **thực hiện thanh toán** (ví dụ mua hàng khi giá phù hợp) thông qua AP2. Mỗi giai đoạn sẽ giới thiệu thêm một công nghệ mới (ADK, MCP, A2A, AP2) để bạn dần hoàn thiện hệ thống:

1. **Bước 1 – Xây dựng agent cơ bản với ADK:** Bắt đầu bằng cách làm quen với ADK. Hãy cài đặt thư viện ADK Python (`pip install google-adk`) và đọc qua tài liệu Quickstart của ADK [17]. Trong giai đoạn này, bạn sẽ tạo một **agent đơn giản** chạy cục bộ, ví dụ: một trợ lý trả lời câu hỏi hoặc một agent đổi ngoại tệ cơ bản. Tập trung vào việc hiểu các **thành phần chính của ADK**: cách định nghĩa một `Agent` với mô hình ngôn ngữ (LLM), thiết lập lời nhắc (instruction) và tích hợp **tools**. Ví dụ, bạn có thể tích hợp sẵn một tool như `google_search` để agent có thể tìm kiếm thông tin [18], hoặc một hàm Python tự chế (như tính toán tỷ giá, truy vấn file JSON sản phẩm). Mục tiêu của bước này là nắm vững cách **khởi tạo và vận hành một agent** dùng ADK, cũng như cách thức agent tương tác với các công cụ nội bộ. Hãy thử cho agent thực hiện vài tác vụ đơn giản và quan sát log đầu ra để hiểu vòng lặp tư duy-hành động của nó.

2. **Bước 2 – Tích hợp MCP: Kết nối agent với công cụ/dịch vụ ngoài:** Khi đã có agent cơ bản, bước tiếp theo là mở rộng năng lực cho nó thông qua MCP. Tìm hiểu khái niệm **MCP server** – đó là một dịch vụ cung cấp một hoặc nhiều công cụ (tools) cho agent truy cập thông qua giao thức MCP [4]. Bạn có thể bắt đầu bằng cách **tạo một MCP server cục bộ**. Ví dụ đơn giản: xây dựng một dịch vụ REST nội bộ (hoặc dùng Node.js) cung cấp API tra cứu sản phẩm (giả lập **kho hàng của merchant**), hoặc API đổi tiền tệ. MCP server này sẽ đăng ký các tool theo chuẩn MCP (có mô tả interface, tham

số, v.v.). Anthropic có sẵn các server mẫu (như server filesystem) và SDK để bạn tham khảo [19] [20]. Sau đó, **tích hợp agent ADK với MCP**: sử dụng **MCP client** để agent có thể gọi được các API từ MCP server như gọi một tool bình thường. Trong tài liệu codelab Google có hướng dẫn tạo MCP server cục bộ và tích hợp nó vào agent ADK làm tool MCP [21] – bạn có thể dựa vào đó. Chẳng hạn, nếu MCP server cung cấp tool "Tra cứu giá sản phẩm", agent của bạn (từ bước 1) có thể gọi tool này để lấy thông tin giá khi người dùng yêu cầu một sản phẩm. Kết thúc bước 2, bạn sẽ có một agent "thông minh" hơn: **vừa có khả năng tư duy ngôn ngữ (LLM), vừa biết truy cập dữ liệu bên ngoài** (qua MCP). Điều này đặt nền móng để agent có thể thực hiện các nhiệm vụ thực tế trong doanh nghiệp.

3. **Bước 3 – Xây dựng hệ thống đa tác nhân với A2A:** Tiếp theo, bạn sẽ mở rộng hệ thống thành **nhiều tác nhân phối hợp** với nhau, ứng dụng giao thức A2A. Thay vì một agent đơn lẻ làm mọi việc, ta phân chia vai trò: ví dụ **Shopping Agent** (đại diện người dùng, sử dụng kết quả từ bước 2) và **Merchant Agent** (đại diện bên bán/nhà cung cấp). Shopping Agent sẽ chịu trách nhiệm tương tác với người dùng (nhận yêu cầu mua hàng, tìm sản phẩm), còn Merchant Agent đóng vai trò như một "dịch vụ" cung cấp thông tin sản phẩm, giá cả và xử lý đơn hàng. Sử dụng ADK, bạn tạo một instance agent thứ hai cho vai trò merchant – agent này có thể đơn giản là một BaseAgent chạy không cần LLM, chỉ cần có logic tra cứu thông tin sản phẩm (có thể gọi trực tiếp vào cơ sở dữ liệu nội bộ hoặc qua MCP server ở bước 2). Bây giờ, triển khai **giao tiếp A2A** giữa hai agent: bạn cấu hình mỗi agent với một **Agent Card** (thẻ thông tin agent) chứa các metadata cần thiết cho A2A (tên, khả năng, endpoint mạng, v.v.) [6]. Dùng SDK A2A (có sẵn cho Python) để **đăng ký Shopping Agent như một A2A server** (lắng nghe các yêu cầu A2A đến) và **Merchant Agent như A2A client** hoặc ngược lại tùy kịch bản [22] [23]. Trong kịch bản này, Shopping Agent sẽ **khởi tạo yêu cầu A2A** gửi sang Merchant Agent khi cần thông tin: ví dụ Shopping Agent hỏi "Giá sản phẩm X là bao nhiêu?" thông qua thông điệp A2A. Merchant Agent nhận được, tra cứu giá (sử dụng MCP tool nếu cần) rồi gửi lại kết quả cho Shopping Agent dưới dạng **Artifact** (đối tượng kết quả chuẩn của A2A). Hãy tham khảo tài liệu hướng dẫn A2A (Quickstart Python) để biết cách thiết lập server và gửi nhận message [22]. Khi tích hợp đúng, bạn sẽ thấy hai agent trao đổi thành công (ví dụ log của merchant agent hiển thị yêu cầu nhận được, log của shopping agent hiển thị phản hồi tương ứng). Kết thúc bước 3, bạn đã xây dựng một **hệ thống đa tác nhân hoàn chỉnh on-premise**: các agent **liên lạc với nhau qua A2A** để chia sẻ nhiệm vụ, trong khi mỗi agent vẫn có **năng lực riêng** (LLM và công cụ qua MCP). Hệ thống này mô phỏng mô hình microservice AI: agent người mua giao tiếp với agent người bán thay vì gọi trực tiếp API lẫn lộn, nhờ đó kiến trúc tổng thể modul hơn và sẵn sàng mở rộng.

4. **Bước 4 – Tích hợp quy trình thanh toán với AP2:** Đây là giai đoạn nâng cao, bổ sung chức năng thanh toán tự động cho hệ thống multi-agent. Bạn sẽ áp dụng giao thức AP2 để đảm bảo giao dịch do agent thực hiện có đầy đủ xác nhận cần thiết. Trước hết, hãy tìm hiểu các **thành phần của AP2**: khái niệm **Mandate** và các vai trò trong quy trình. AP2 định nghĩa ba loại *Mandate* (ủy thác) chính dưới dạng **chứng chỉ số (Verifiable Credentials)**: **Intent Mandate** (đại diện cho ý định/ủy quyền ban đầu của người dùng, ký bởi người dùng), **Cart Mandate** (đại diện cho giỏ hàng cuối cùng được người dùng duyệt, ký bởi user và phía merchant) và **Payment Mandate** (thông tin gửi đến mạng lưới thanh toán, cho biết giao dịch có agent tham gia) [24] [25]. Trong dự án của bạn, chúng ta có thể giả lập hai tình huống: **human-present** (người dùng duyệt mua tại thời điểm thanh toán) và **human-not-present** (người dùng ủy quyền trước, agent tự mua sau).

Mô phỏng kiến trúc AP2: AI Shopping Agent (giữa) kết nối với hệ sinh thái Merchant (bên trái: bao gồm Merchant Agent, dịch vụ ví/thẻ của người dùng) và hệ sinh thái Thanh toán (bên phải: cổng thanh toán, mạng lưới ngân hàng). AP2 đảm bảo mọi bước từ ý định đến thanh toán đều có bằng chứng mật mã kèm theo [26] [27].

Triển khai AP2 vào hệ thống: Trong kịch bản **human-present**, luồng có thể như sau: Người dùng ra lệnh cho Shopping Agent "mua sản phẩm X giá tốt nhất có thể". Shopping Agent tìm được sản phẩm qua Merchant Agent, đưa ra giỏ hàng (cart) với giá cụ thể. Trước khi thanh toán, **người dùng xác nhận giỏ hàng** này trên giao diện (ví dụ bạn in ra chi tiết và yêu cầu người dùng đồng ý). Lúc này, tạo một **Cart Mandate** – một object chứa chi tiết đơn hàng (mã sản phẩm, giá tiền, merchant...) được **merchant ký xác nhận** (đảm bảo "những gì user thấy là những gì sẽ trả") và **người dùng ký chấp thuận** [28] [29]. Bạn có thể sử dụng thư viện từ repo AP2 (các class trong `ap2.types`) để tạo các đối tượng mandate này theo đúng cấu trúc chuẩn. Trong môi trường phát triển, bạn sẽ cần sinh khóa ký số cho các bên: ví dụ tạo cặp public/private key đại diện cho người dùng và merchant (có thể dùng thư viện crypto như `cryptography` của Python), rồi dùng chúng ký lên thông điệp JSON của Mandate. AP2 yêu cầu chữ ký và định dạng tuân theo chuẩn VC (W3C Verifiable Credentials), bạn có thể tham khảo mã mẫu trong **AP2 sample** để thực hiện đúng [30]. Sau khi có Cart Mandate đã ký, Shopping Agent gửi yêu cầu thanh toán kèm mandate này cho **Payment Processor** (giả lập thành một **Payment Agent** hoặc đơn giản là một module kiểm tra). Payment Agent sẽ tạo tiếp **Payment Mandate** (chứa thông tin rằng AI agent đã tham gia giao dịch, tránh rủi ro) và hoàn tất giao dịch (ở bản demo, bạn chỉ cần in ra "Thanh toán thành công" kèm chuỗi bằng chứng). Kết thúc luồng, bạn sẽ có **chuỗi bằng chứng mật mã** liên kết từ ủy quyền ban đầu của user đến giao dịch cuối cùng, có thể kiểm tra độc lập.

Trong kịch bản **human-not-present** (mua tự động khi có điều kiện), bạn làm tương tự nhưng thêm bước tạo **Intent Mandate** từ đầu: ví dụ người dùng *ký trước một Intent Mandate* với điều kiện "mua sản phẩm X nếu giá < 100$" [31]. Shopping Agent giữ mandate này, đến khi Merchant Agent báo giá thỏa mãn thì tự động tiến hành tạo Cart Mandate và thanh toán không cần hỏi lại. Bạn có thể thử mô phỏng tình huống này để hiểu thêm, nhưng ban đầu hãy ưu tiên luồng human-present cơ bản cho chắc chắn.

Lưu ý triển khai: Do trong môi trường on-premise, bạn sẽ giả lập hầu hết các vai trò, nên việc tích hợp có thể phức tạp. Hãy tận dụng **mã nguồn mở của AP2** – Google đã cung cấp **repo mẫu (Apache-2.0)** bao gồm **các loại dữ liệu chuẩn** và **tình huống giả lập** cho cả thanh toán thẻ và crypto [30]. Bạn có thể xem một kịch bản mẫu trong đó (vd. thanh toán thẻ tín dụng) và đối chiếu với hệ thống của mình. Đảm bảo rằng chữ ký số của bạn hoạt động đúng (thử thay đổi nội dung cart xem chữ ký có bị invalid không để kiểm tra). Ngoài ra, khi tích hợp AP2, hệ thống của bạn sẽ có thêm các thành phần: **Credentials Provider** (ví dụ một module giữ phương thức thanh toán của user, ở đây có thể đơn giản coi user cung cấp thông tin thẻ), **Merchant Endpoint/Processor** (agent merchant đóng vai trò này, ký cart và xử lý thanh toán) [26]. Hãy cố gắng giữ cho mô-đun rõ ràng, ví dụ tách biệt phần logic ký xác nhận của merchant và phần logic xử lý thanh toán cuối (có thể gộp vào Payment Agent). Mục tiêu của bước 4 là bạn hiểu được cách **AP2 đảm bảo tính tin cậy**: mọi bên đều có chứng cứ về hành động của agent, tránh tình huống agent mua linh tinh ngoài ý muốn. Kết quả dự án sẽ là một **prototype hệ thống thương mại tự động**: người dùng chỉ việc đưa ra yêu cầu và giới hạn, phần còn lại agent lo liệu, và tất cả đều được kiểm soát an toàn.

Kết luận: Sau khi hoàn thành các bước trên, bạn sẽ có một cái nhìn toàn diện về cách kết hợp ADK, MCP, A2A và AP2. Bạn đã xây dựng một dự án mẫu on-premise trong đó một agent AI biết sử dụng công cụ (MCP), biết phối hợp với agent khác (A2A), và thậm chí thực hiện giao dịch thanh toán có kiểm soát (AP2). Lộ trình này không chỉ giúp bạn nắm vững từng công nghệ riêng lẻ mà quan trọng hơn là hiểu **cách chúng phối hợp trong một hệ sinh thái agent**. Từ đây, bạn có thể thử nghiệm mở rộng dự án (ví dụ tích hợp thêm nhiều agent chuyên biệt, kết nối hệ thống thanh toán thực tế hoặc ví crypto cho AP2, v.v.) và dần hướng tới một sản phẩm thương mại hóa cho công ty. Hãy luôn bám sát tài liệu chính thức và các mẫu mã nguồn có sẵn – đó là cách học hiệu quả và nhanh chóng nhất để làm chủ các công nghệ mới này. Chúc bạn thành công trong việc xây dựng dự án và ứng dụng kiến thức đã học một cách toàn diện!

Tài liệu tham khảo: Các nội dung trên có tham khảo từ tài liệu chính thức của Google và Anthropic về ADK, A2A, MCP, AP2 cũng như bài viết chuyên môn liên quan (Medium, MarkTechPost, Google Codelabs,...). Bạn có thể tìm đọc thêm để đào sâu từng phần nếu cần thiết. Các liên kết cụ thể đã được trích dẫn trong bài để tiện tra cứu. [1] [13]

---

[1] [5] [7] [21] Bắt đầu sử dụng MCP, ADK và A2A | Google Codelabs
https://codelabs.developers.google.com/codelabs/currency-agent?hl=vi

[2] [3] [17] [18] [23] GitHub - google/adk-python: An open-source, code-first Python toolkit for building, evaluating, and deploying sophisticated AI agents with flexibility and control.
https://github.com/google/adk-python

[4] What is the Model Context Protocol (MCP)? - Model Context Protocol
https://modelcontextprotocol.io/docs/getting-started/intro

[6] [8] [9] [13] [14] [22] A2A Protocol Documentation
https://a2a-protocol.org/latest/

[10] [12] [28] [29] Google chính thức ra mắt Agent Payments Protocol - Chuyên trang Sài Gòn Đầu Tư Tài Chính - Báo Sài Gòn Giải Phóng
https://baomoi.com/google-chinh-thuc-ra-mat-agent-payments-protocol-c53261810.epi

[11] [24] [25] [26] [27] [30] [31] Google AI Introduces Agent Payments Protocol (AP2): An Open Protocol for Interoperable AI Agent Checkout Across Merchants and Wallets - MarkTechPost
https://www.marktechpost.com/2025/09/16/google-ai-introduces-agent-payments-protocol-ap2-an-open-protocol-for-interoperable-ai-agent-checkout-across-merchants-and-wallets/

[15] [16] A2A, MCP, and ADK — Clarifying Their Roles in the AI Ecosystem | by #TheGenAIGirl — code, community, and GenAI. | Google Cloud - Community | Medium
https://medium.com/google-cloud/a2a-mcp-and-adk-clarifying-their-roles-in-the-ai-ecosystem-44727700a0c6

[19] [20] Connect to local MCP servers - Model Context Protocol
https://modelcontextprotocol.io/docs/develop/connect-local-servers